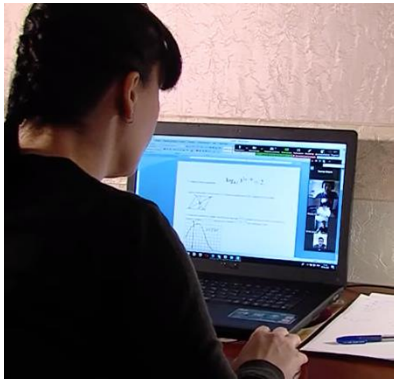
What is a Computer?

a *computer* is a device that receives, stores, and processes information

different types of computers have different characteristics

- *supercomputers:* powerful but expensive; used for complex computations (e.g., weather forecasting, engineering design and modeling)
- *desktop computers:* less powerful but affordable; used for a variety of user applications (e.g., email, Web browsing, document processing)
- *laptop computers:* similar functionality to desktops, but mobile
- *palmtop computers:* portable, but limited applications and screen size
- smartphones: portable, integrated with phone, camera



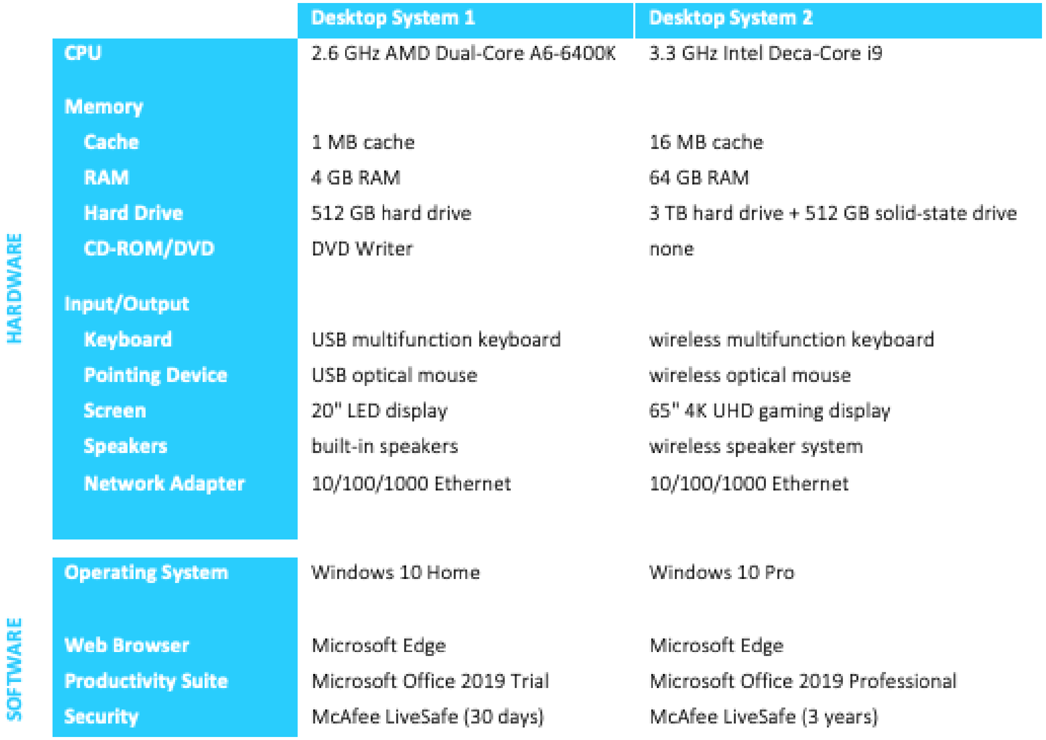
Desktop Specifications

purchasing a computer can be confusing

- sales materials contain highly technical information and computer jargon

the following specs describe two computer systems for sale in January 2020

- Desktop 1 is a low-end system, inexpensive but with limited features
- Desktop 2 is a high-end system, uses the latest technology so expensive

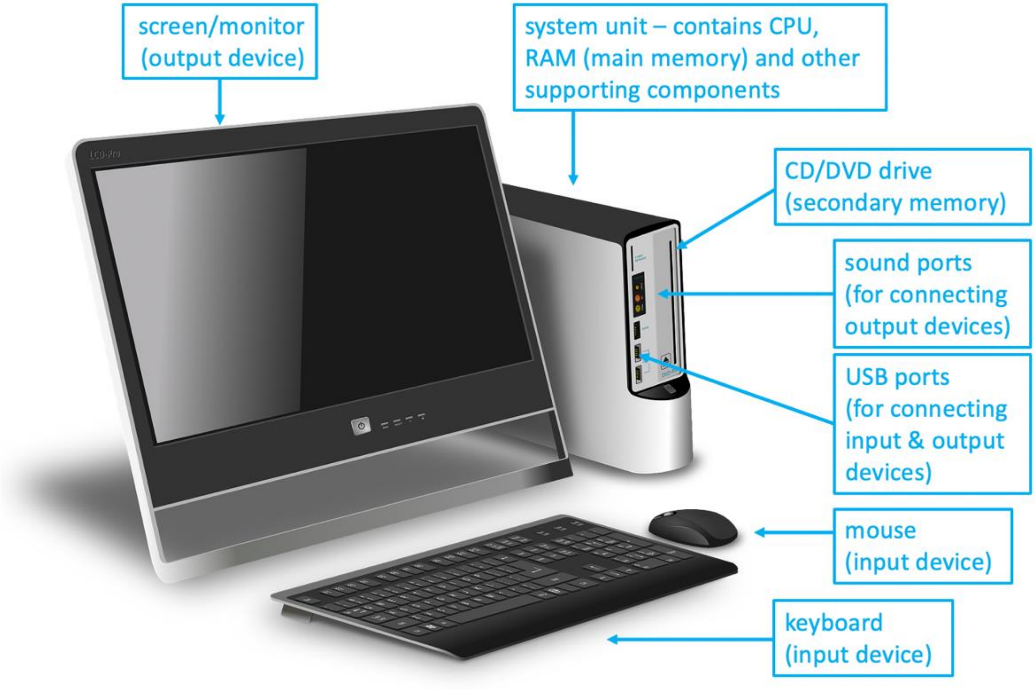


		Desktop System 1	Desktop System 2
HARDWARE	CPU	2.6 GHz AMD Dual-Core A6-6400K	3.3 GHz Intel Deca-Core i9
	Memory		
	Cache	1 MB cache	16 MB cache
	RAM	4 GB RAM	64 GB RAM
	Hard Drive	512 GB hard drive	3 TB hard drive + 512 GB solid-state drive
	CD-ROM/DVD	DVD Writer	none
	Input/Output		
	Keyboard	USB multifunction keyboard	wireless multifunction keyboard
	Pointing Device	USB optical mouse	wireless optical mouse
	Screen	20" LED display	65" 4K UHD gaming display
	Speakers	built-in speakers	wireless speaker system
	Network Adapter	10/100/1000 Ethernet	10/100/1000 Ethernet
SOFTWARE	Operating System	Windows 10 Home	Windows 10 Pro
	Web Browser	Microsoft Edge	Microsoft Edge
	Productivity Suite	Microsoft Office 2019 Trial	Microsoft Office 2019 Professional
	Security	McAfee LiveSafe (30 days)	McAfee LiveSafe (3 years)

Hardware vs. Software

the term *hardware* refers to the physical components of a computer system

- e.g., monitor, keyboard, mouse, hard drive


screen/monitor (output device)
system unit – contains CPU, RAM (main memory) and other supporting components
CD/DVD drive (secondary memory)
sound ports (for connecting output devices)
USB ports (for connecting input & output devices)
mouse (input device)
keyboard (input device)

the term *software* refers to the programs that execute on the computer

- e.g., word processing program, Web browser

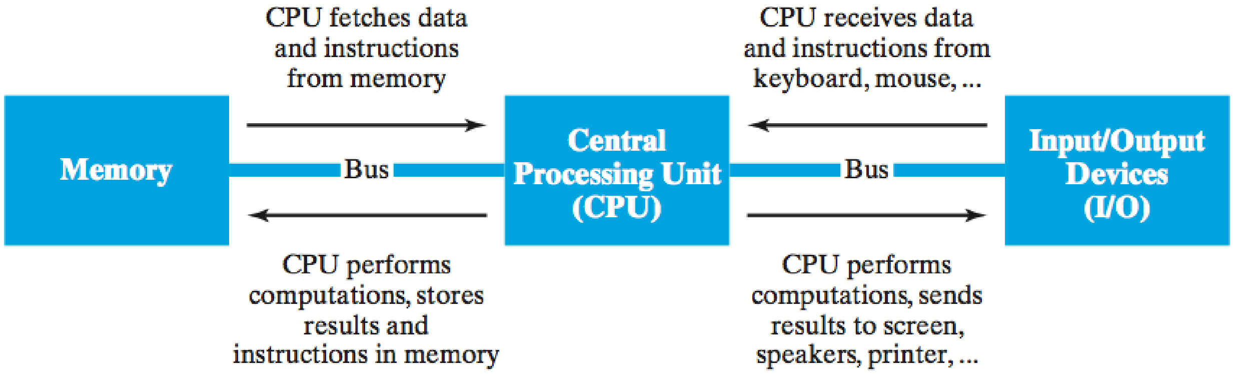
von Neumann Architecture

although specific components may vary, virtually all modern computers have the same underlying structure

- known as the *von Neumann architecture*
- named after computer pioneer, John von Neumann, who popularized the design in the early 1950's

the von Neumann architecture identifies 3 essential components

1. *Input/Output Devices (I/O)* allow the user to interact with the computer
2. *Memory* stores information to be processed as well as programs (instructions specifying the steps necessary to complete specific tasks)
3. *Central Processing Unit (CPU)* carries out the instructions to process information

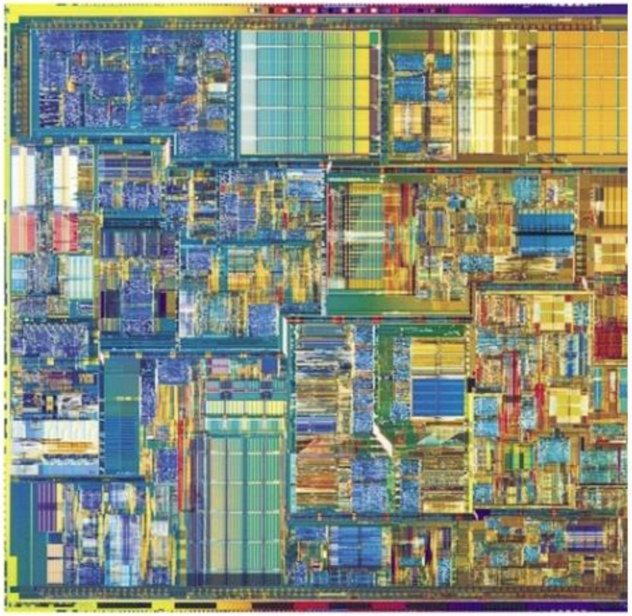

CPU fetches data and instructions from memory
CPU receives data and instructions from keyboard, mouse, ...
Memory
Bus
Central Processing Unit (CPU)
Bus
Input/Output Devices (I/O)
CPU performs computations, stores results and instructions in memory
CPU performs computations, sends results to screen, speakers, printer, ...

Central Processing Unit (CPU)

the CPU is the "brains" of the computer, responsible for controlling its inner workings

- made of *circuitry* – electronic components wired together to control the flow of electrical signals
- the circuitry is embedded in a small silicon chip, 1-2 inches square
- despite its small size, the CPU is the most complex part of a computer
  (CPU circuitry can have 100's of millions of individual components)

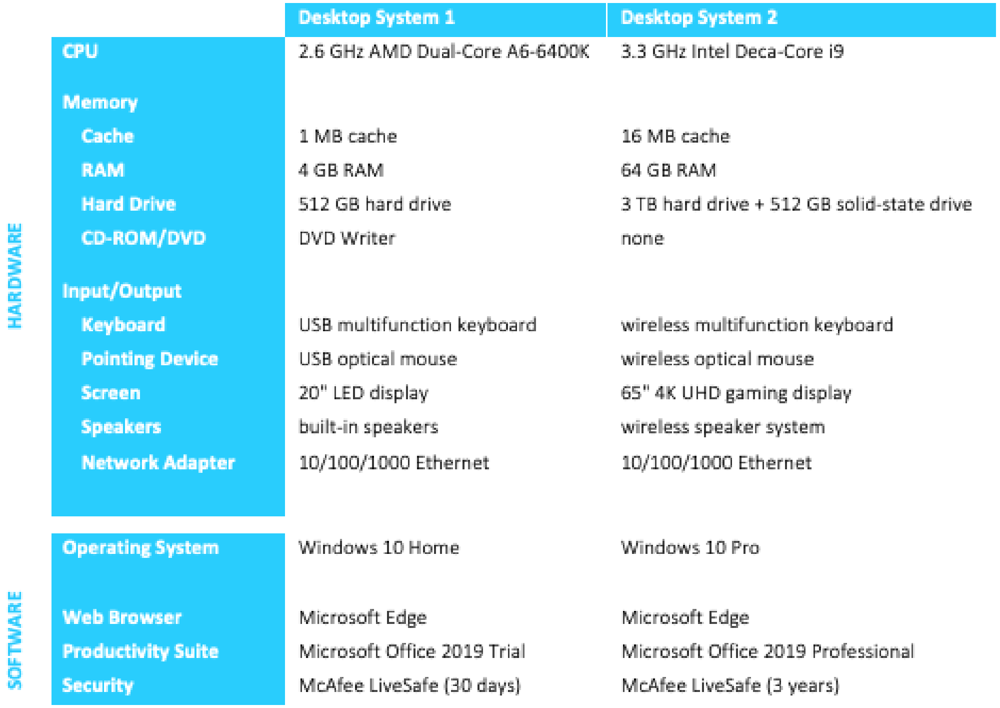
- commercial examples: AMD Ryzen 5, Intel Core i5, and Intel Core i7



CPU (cont.)

the CPU works by repeatedly fetching a program instruction from memory and executing that instruction

- individual instructions are very simple (e.g., add two numbers, or copy this data)
  - but they vary across CPUs, higher end can do more in a single instruction
- complex behavior results from incredible speed
  - a 2.6 GHz AMD A6 processor can execute 2.6 billion instructions per second
  - a 3.3 GHz Intel i9 processor can execute 3.3 billion instructions per second



		Desktop System 1	Desktop System 2
HARDWARE	CPU	2.6 GHz AMD Dual-Core A6-6400K	3.3 GHz Intel Deca-Core i9
	Memory		
	Cache	1 MB cache	16 MB cache
	RAM	4 GB RAM	64 GB RAM
	Hard Drive	512 GB hard drive	3 TB hard drive + 512 GB solid-state drive
	CD-ROM/DVD	DVD Writer	none
	Input/Output		
	Keyboard	USB multifunction keyboard	wireless multifunction keyboard
	Pointing Device	USB optical mouse	wireless optical mouse
	Screen	20" LED display	65" 4K UHD gaming display
	Speakers	built-in speakers	wireless speaker system
	Network Adapter	10/100/1000 Ethernet	10/100/1000 Ethernet
SOFTWARE	Operating System	Windows 10 Home	Windows 10 Pro
	Web Browser	Microsoft Edge	Microsoft Edge
	Productivity Suite	Microsoft Office 2019 Trial	Microsoft Office 2019 Professional
	Security	McAfee LiveSafe (30 days)	McAfee LiveSafe (3 years)

a dual-core processor contains the circuitry of 2 processors, packaged on a single chip

- in theory, can execute 2 instructions simultaneously

a deca-core processor contains the circuitry of 10 processors, packaged on a single chip

- in theory, can execute 10 instructions simultaneously

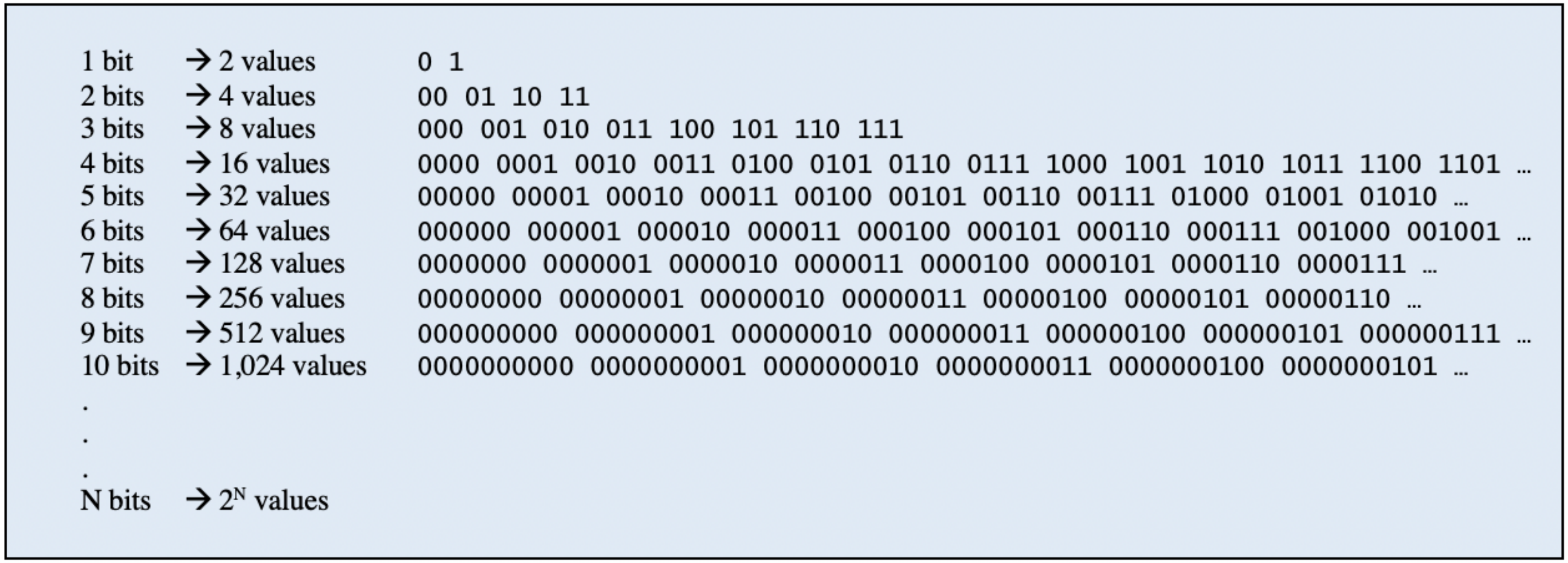
Memory

memory is the part of the computer that stores data and programs

modern computers are *digital* devices, meaning they store and process information as *binary digits (bits)*

- bits are commonly represented as either 0 or 1
- bits are the building block of digital memory

  by grouping bits together, large ranges of values can be represented

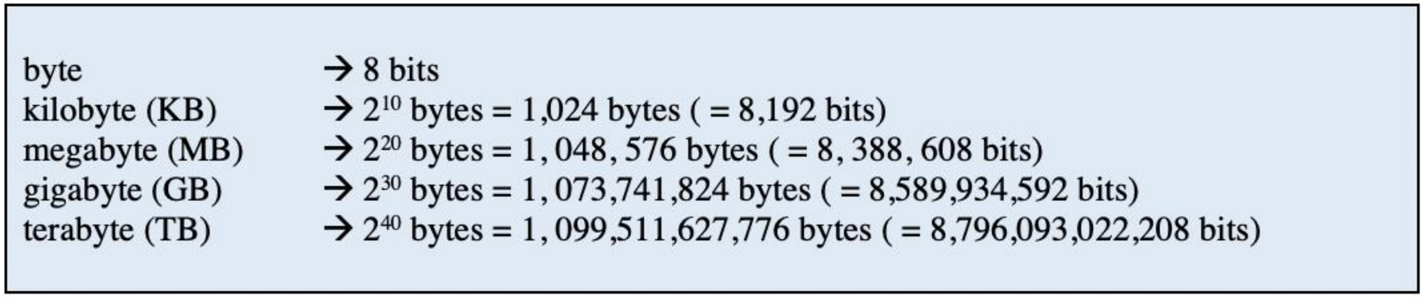


1 bit	→ 2 values	0 1
2 bits	→ 4 values	00 01 10 11
3 bits	→ 8 values	000 001 010 011 100 101 110 111
4 bits	→ 16 values	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 …
5 bits	→ 32 values	00000 00001 00010 00011 00100 00101 00110 00111 01000 01001 01010 …
6 bits	→ 64 values	000000 000001 000010 000011 000100 000101 000110 000111 001000 001001 …
7 bits	→ 128 values	0000000 0000001 0000010 0000011 0000100 0000101 0000110 0000111 …
8 bits	→ 256 values	00000000 00000001 00000010 00000011 00000100 00000101 00000110 …
9 bits	→ 512 values	000000000 000000001 000000010 000000011 000000100 000000101 000000111 …
10 bits	→ 1,024 values	0000000000 0000000001 0000000010 0000000011 0000000100 0000000101 …
.		
.		
.		
N bits	→ 2^N values	

Memory (cont.)

memory capacity is usually specified in bytes

- a *byte* is a collection of 8 bits – so can represent a range of 2^8 = 256 values
- large collections of bytes can be specified using prefixes



byte	→ 8 bits
kilobyte (KB)	→ 2^{10} bytes = 1,024 bytes (= 8,192 bits)
megabyte (MB)	→ 2^{20} bytes = 1, 048, 576 bytes (= 8, 388, 608 bits)
gigabyte (GB)	→ 2^{30} bytes = 1, 073,741,824 bytes (= 8,589,934,592 bits)
terabyte (TB)	→ 2^{40} bytes = 1, 099,511,627,776 bytes (= 8,796,093,022,208 bits)

since a byte is sufficient to represent a single character, can think of memory in terms of text

- a kilobyte can store a few paragraphs (roughly 1 thousand characters)
- a megabyte can store a book (roughly 1 million characters)
- a gigabyte can store a small library (roughly 1 billion characters)
- a terabyte can store a book repository (roughly 1 trillion characters)

Memory (cont.)

modern computers use a combination of memory types, each with its own performance and cost characteristics

main memory (or *primary memory*) is fast and expensive

- data is stored as electric signals in circuitry, used to store active data
- memory is volatile – data is lost when the computer is turned off
- examples: Random Access Memory (RAM), cache

secondary memory is slower but cheaper

- use different technologies (magnetic signals on hard disk, reflective spots on CD)
- memory is permanent – useful for storing long-term data
- examples: hard disk, flash drive, compact disk (CD)

RAM chips

hard drive

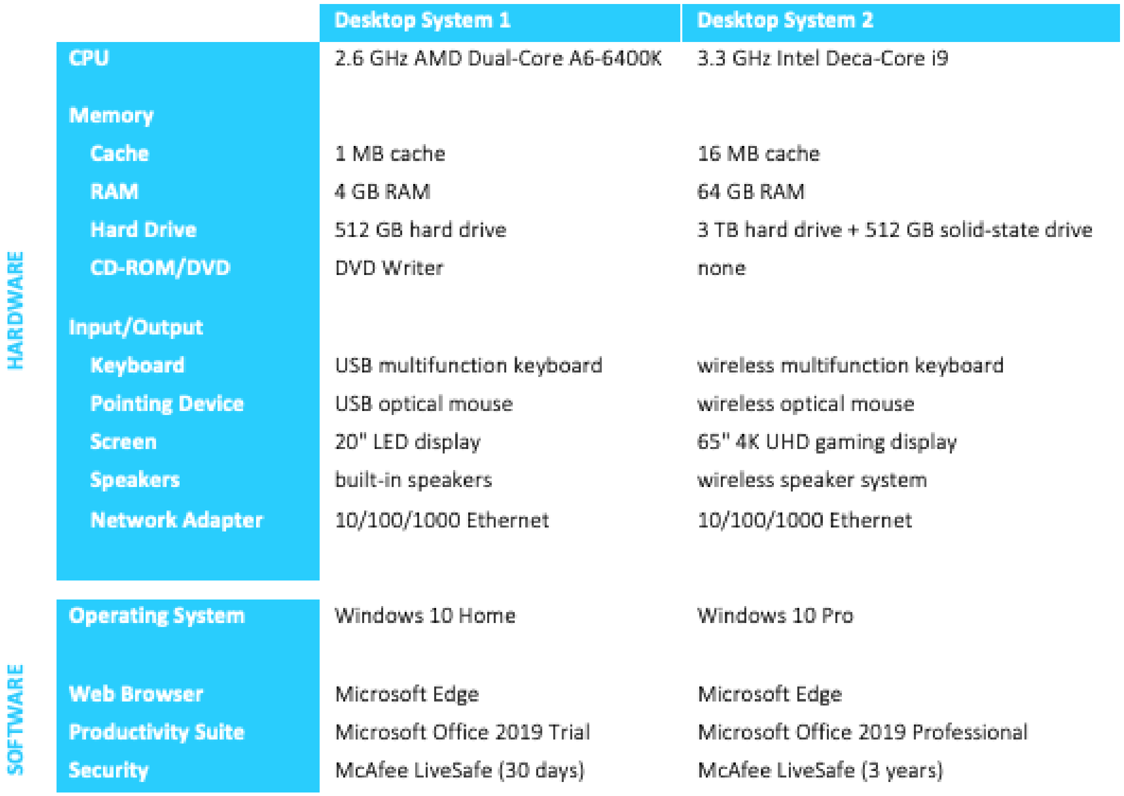
flash drive

CD/DVD

Memory (cont.)

higher-end computers tend to have

- more main memory to allow for quick access to more data and programs
- more secondary memory to allow for storing more long-term data



		Desktop System 1	Desktop System 2
HARDWARE	CPU	2.6 GHz AMD Dual-Core A6-6400K	3.3 GHz Intel Deca-Core i9
	Memory		
	Cache	1 MB cache	16 MB cache
	RAM	4 GB RAM	64 GB RAM
	Hard Drive	512 GB hard drive	3 TB hard drive + 512 GB solid-state drive
	CD-ROM/DVD	DVD Writer	none
	Input/Output		
	Keyboard	USB multifunction keyboard	wireless multifunction keyboard
	Pointing Device	USB optical mouse	wireless optical mouse
	Screen	20" LED display	65" 4K UHD gaming display
	Speakers	built-in speakers	wireless speaker system
	Network Adapter	10/100/1000 Ethernet	10/100/1000 Ethernet
SOFTWARE	Operating System	Windows 10 Home	Windows 10 Pro
	Web Browser	Microsoft Edge	Microsoft Edge
	Productivity Suite	Microsoft Office 2019 Trial	Microsoft Office 2019 Professional
	Security	McAfee LiveSafe (30 days)	McAfee LiveSafe (3 years)

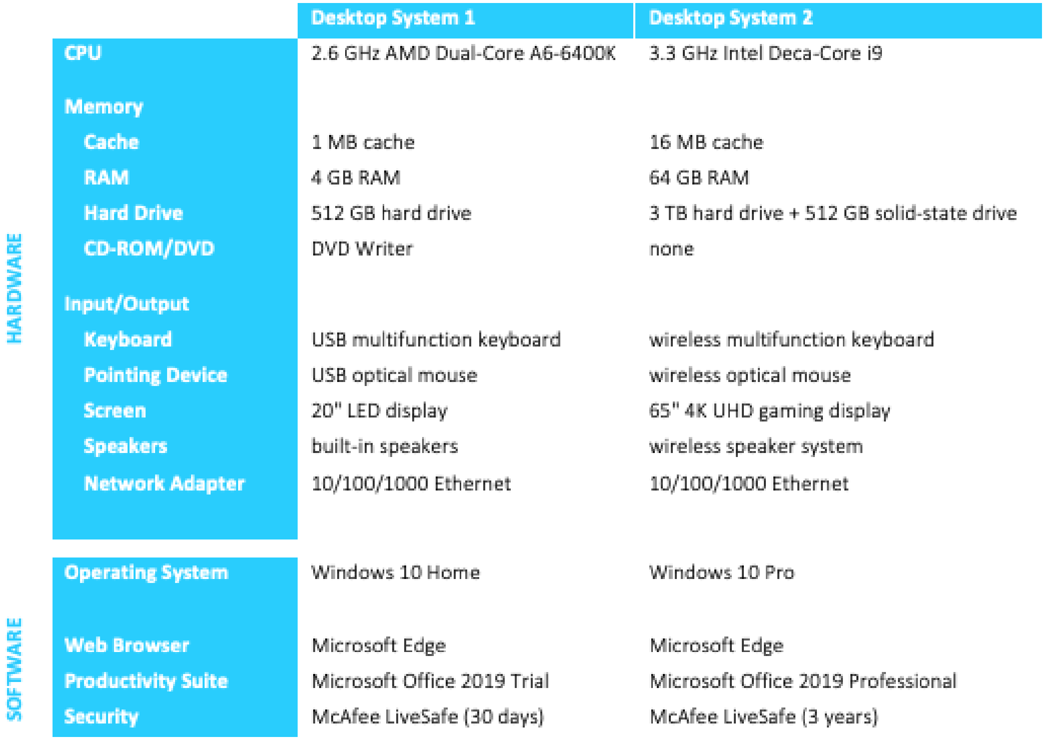
Input/Output (I/O)

input devices allow the computer to receive data and instructions from external sources

- examples: keyboard, mouse, track pad, touch screen, microphone, scanner

output devices allow the computer to display or broadcast its results

- examples: monitor, speaker, printer



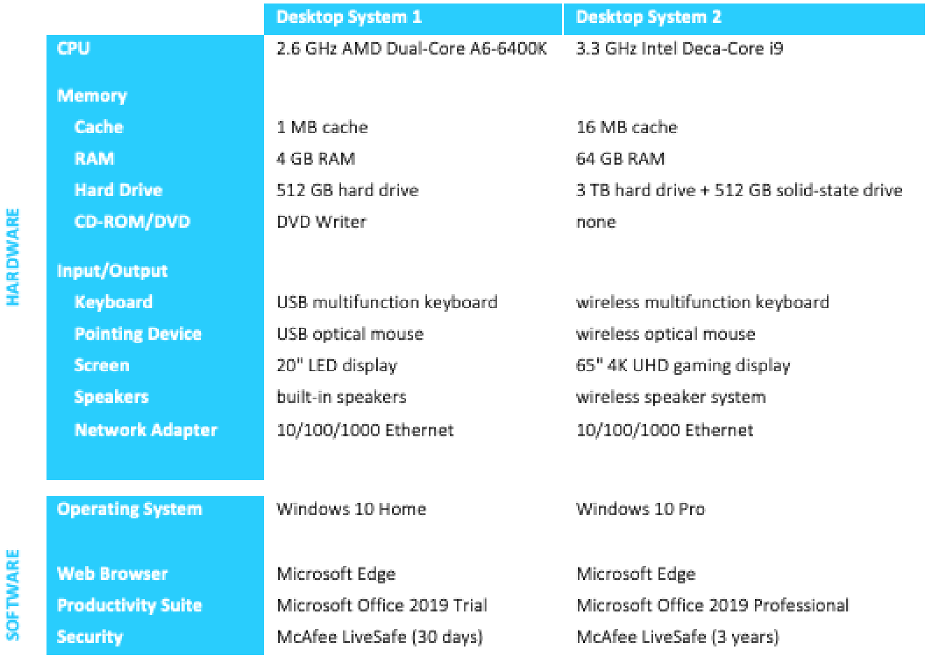
		Desktop System 1	Desktop System 2
HARDWARE	CPU	2.6 GHz AMD Dual-Core A6-6400K	3.3 GHz Intel Deca-Core i9
	Memory		
	Cache	1 MB cache	16 MB cache
	RAM	4 GB RAM	64 GB RAM
	Hard Drive	512 GB hard drive	3 TB hard drive + 512 GB solid-state drive
	CD-ROM/DVD	DVD Writer	none
	Input/Output		
	Keyboard	USB multifunction keyboard	wireless multifunction keyboard
	Pointing Device	USB optical mouse	wireless optical mouse
	Screen	20" LED display	65" 4K UHD gaming display
	Speakers	built-in speakers	wireless speaker system
	Network Adapter	10/100/1000 Ethernet	10/100/1000 Ethernet
SOFTWARE	Operating System	Windows 10 Home	Windows 10 Pro
	Web Browser	Microsoft Edge	Microsoft Edge
	Productivity Suite	Microsoft Office 2019 Trial	Microsoft Office 2019 Professional
	Security	McAfee LiveSafe (30 days)	McAfee LiveSafe (3 years)

Software

recall: *hardware* refers to the physical components of computers
software refers to the programs that execute on the hardware

a software program is a sequence of instructions for the computer (more specifically, for the CPU) to carry out in order to complete some task

- e.g., word processing (Microsoft Word, Corel WordPerfect)
- e.g., image processing (Adobe Photoshop, Flash)
- e.g., Web browsing (Microsoft Edge, Mozilla Firefox, Google Chrome, Safari)

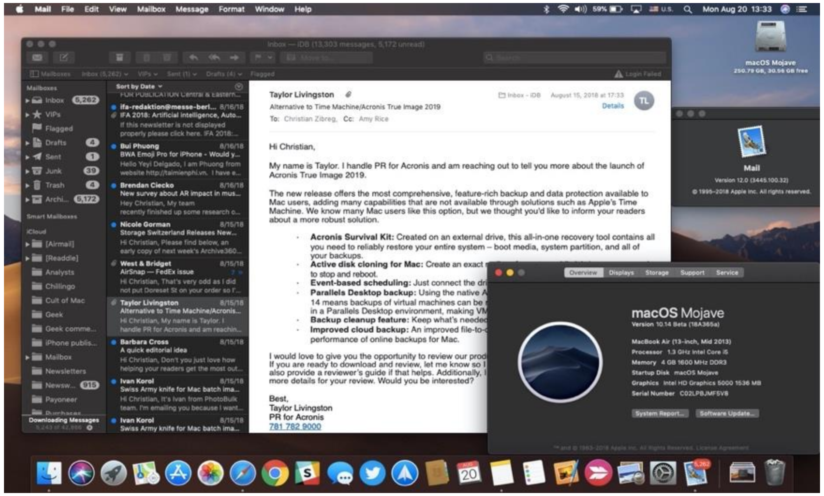


		Desktop System 1	Desktop System 2
HARDWARE	CPU	2.6 GHz AMD Dual-Core A6-6400K	3.3 GHz Intel Deca-Core i9
	Memory		
	Cache	1 MB cache	16 MB cache
	RAM	4 GB RAM	64 GB RAM
	Hard Drive	512 GB hard drive	3 TB hard drive + 512 GB solid-state drive
	CD-ROM/DVD	DVD Writer	none
	Input/Output		
	Keyboard	USB multifunction keyboard	wireless multifunction keyboard
	Pointing Device	USB optical mouse	wireless optical mouse
	Screen	20" LED display	65" 4K UHD gaming display
	Speakers	built-in speakers	wireless speaker system
	Network Adapter	10/100/1000 Ethernet	10/100/1000 Ethernet
SOFTWARE	Operating System	Windows 10 Home	Windows 10 Pro
	Web Browser	Microsoft Edge	Microsoft Edge
	Productivity Suite	Microsoft Office 2019 Trial	Microsoft Office 2019 Professional
	Security	McAfee LiveSafe (30 days)	McAfee LiveSafe (3 years)

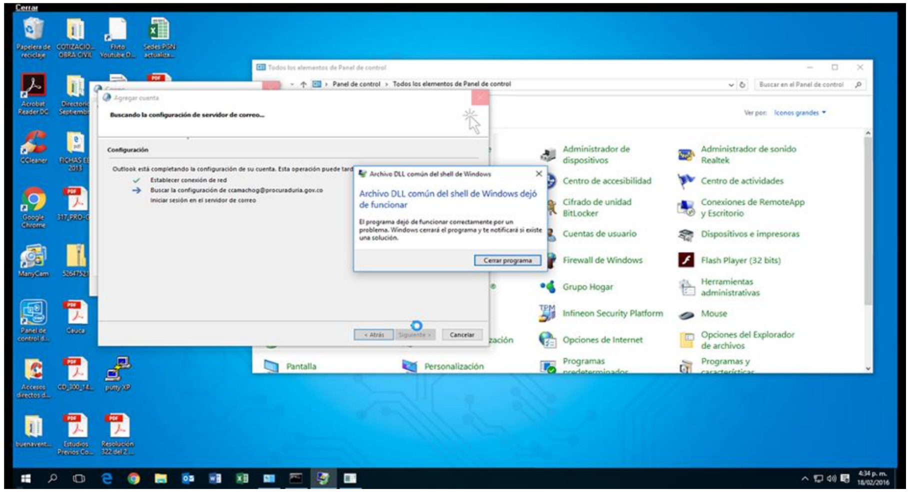
Operating Systems

the *Operating System (OS)* is a collection of programs that controls how the CPU, memory, and I/O devices work together

- *kernel*: manages the CPU's operations, controls how data and instructions are loaded and executed by the CPU, coordinates other hardware components
- *file system*: organizes and manages files and directories
- *graphical user interface (GUI)*: provides intuitive, visual elements for interacting with the computer
  - GUI's utilize windows, icons, menus, and pointers



Mail File Edit View Mailbox Message Format Window Help
macOS Mojave
Overview Displays Storage Support Service
System Report... Software Update...

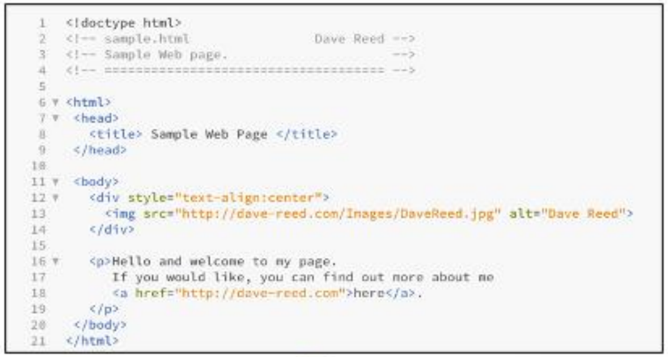


Panel de control
Todos los elementos de Panel de control
Archivo DLL común del shell de Windows dejó de funcionar
Cerrar programa

HTML & Web Pages

a Web page is a text document that contains additional formatting information in the HyperText Markup Language (HTML)

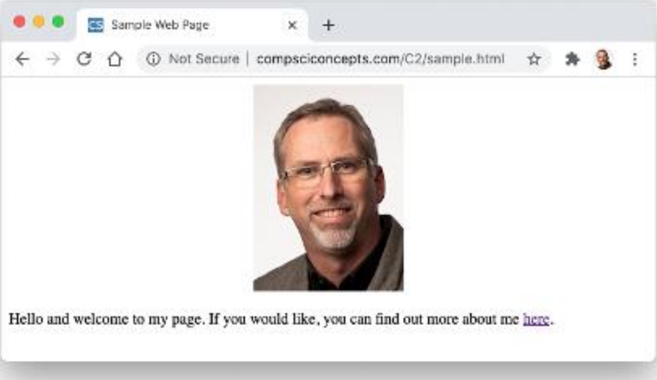
- HTML specifies formatting within a page using *tags*
- in its simplest form, a tag is a word or symbol surrounded by brackets (<>)



```
<!doctype html>
<!-- sample.html                Dave Reed -->
<!-- Sample Web page.                     -->
<!-- ==================================== -->

<html>
 <head>
   <title> Sample Web Page </title>
 </head>

 <body>
   <div style="text-align:center">
     <img src="http://dave-reed.com/Images/DaveReed.jpg" alt="Dave Reed">
   </div>

   <p>Hello and welcome to my page.
      If you would like, you can find out more about me
      <a href="http://dave-reed.com">here</a>.
   </p>
 </body>
</html>
```

A Web page is a text document that contains HTML formatting.

A Web browser (e.g., Chrome) is a program that interprets the HTML and displays the page.

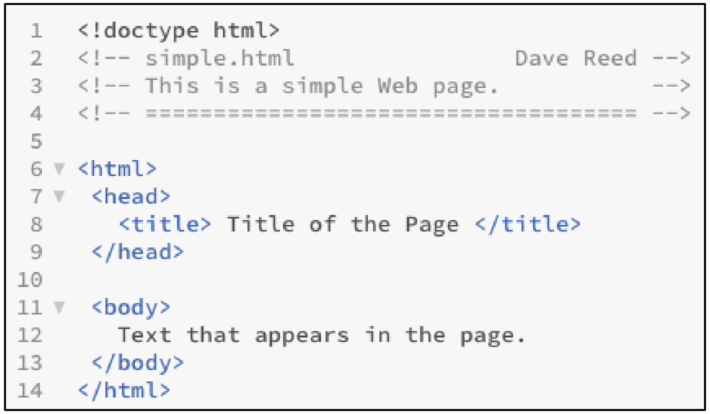


Hello and welcome to my page. If you would like, you can find out more about me here.

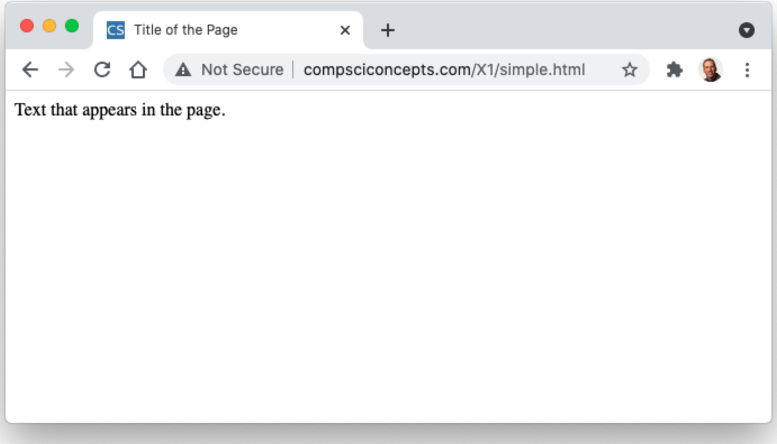
HTML Tags

required tags in a Web page:

- <!doctype html> tells the browser this is a Web page
- <html> and </html> enclose the entire HTML document
- the head section (enclosed between <head> and </head>) contains information that the browser uses to control the look of the page
  - the head can contain a title for the browser window, enclosed between <title> and </title>
- the body section (enclosed between <body> and </body>) contains the text that will appear in the page



```
<!doctype html>
<!-- simple.html                Dave Reed -->
<!-- This is a simple Web page.           -->
<!-- ==================================== -->

<html>
 <head>
   <title> Title of the Page </title>
 </head>

 <body>
   Text that appears in the page.
 </body>
</html>
```



Text that appears in the page.

HTML Elements

tags and the text they enclose form an *HTML element*

```
<title> Title of the Page </title>
```

is a `title` element

```
<head>
  <title> Title of the Page </title>
</head>
```

is a head element (which contains a nested `title` element)

most HTML elements have opening and closing tags, but not all

```
<!-- simple.html      Dave Reed -->
```

is a comment element

- a comment is ignored by the browser (it does not appear in the rendered page)
- comments are used by the page developer to document page features

Text Layout & Formatting

extra white space (spaces, tabs and blank lines) is ignored by the browser

- this allows the browser to adjust the text to the window size

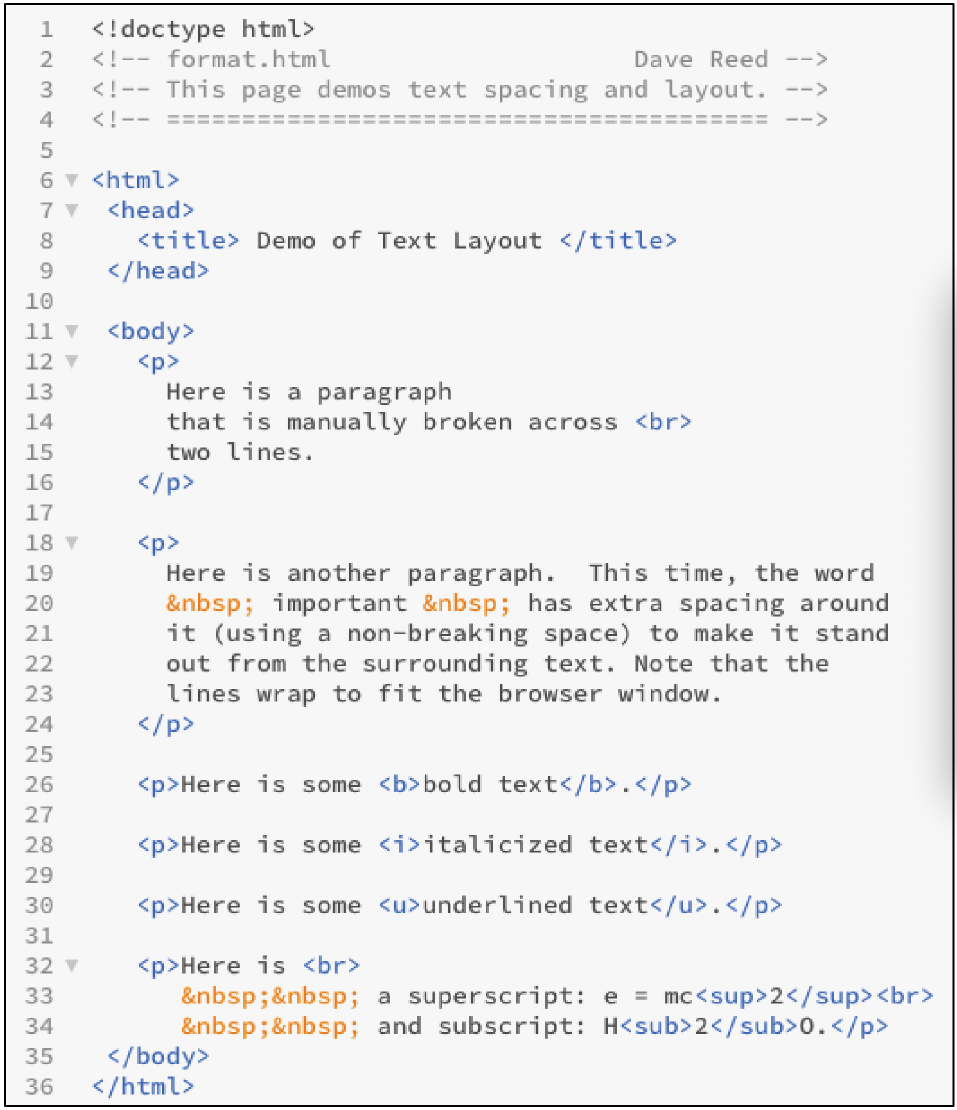
you can control some of the text layout using HTML elements

- a paragraph element (`<p>…</p>`) specifies text surrounded by blank lines
- a break element (`<br>`) causes text to be displayed on a new line
- the `&nbsp;` special symbol forces a space to appear in the text

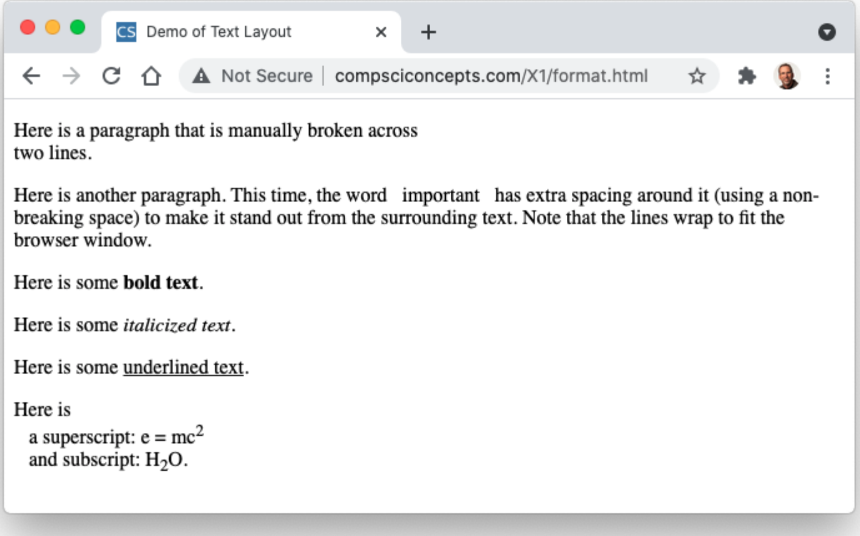
you can specify some text formatting using other HTML elements

- a b element (`<b>…</b>`) specifies **bold text**
- an `i` element (`<i>…</i>`) specifies *italicized text*
- a u element (`<u>…</u>`) specifies underlined text
- a sup element (`<sup>…</sup>`) specifies $^{\text{superscripted text}}$
- a sub element (`<sub>…</sub>`) specifies $_{\text{superscripted text}}$

Layout & Formatting Example



```
<!doctype html>
<!-- format.html                    Dave Reed -->
<!-- This page demos text spacing and layout. -->
<!-- ======================================== -->

<html>
 <head>
   <title> Demo of Text Layout </title>
 </head>

 <body>
   <p>
     Here is a paragraph
     that is manually broken across <br>
     two lines.
   </p>

   <p>
     Here is another paragraph.  This time, the word
     &nbsp; important &nbsp; has extra spacing around
     it (using a non-breaking space) to make it stand
     out from the surrounding text. Note that the
     lines wrap to fit the browser window.
   </p>

   <p>Here is some <b>bold text</b>.</p>

   <p>Here is some <i>italicized text</i>.</p>

   <p>Here is some <u>underlined text</u>.</p>

   <p>Here is <br>
      &nbsp;&nbsp; a superscript: e = mc<sup>2</sup><br>
      &nbsp;&nbsp; and subscript: H<sub>2</sub>O.</p>
 </body>
</html>
```



Here is a paragraph that is manually broken across
two lines.

Here is another paragraph. This time, the word   important   has extra spacing around it (using a non-breaking space) to make it stand out from the surrounding text. Note that the lines wrap to fit the browser window.

Here is some **bold text**.

Here is some *italicized text*.

Here is some underlined text.

Here is
  a superscript: $e = mc^2$
  and subscript: H_2O.

Sections

in a large document, it is useful to divide the text into sections and then provide each with a heading describing the content that follows

- <h1> … </h1> enclose a **Top-level Heading**
- <h2> … </h2> enclose a **Sub-heading**

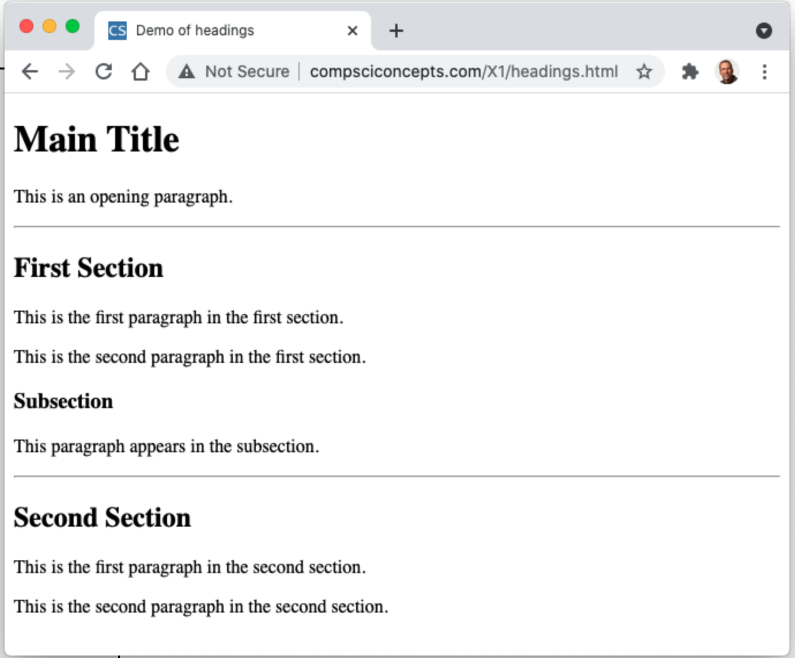
.
.
.

- <h6> … </h6> enclose a **Sub-sub-sub-sub-sub-heading**

the horizontal-rule element <hr> draws a dividing line in the page

Section Example

```
<!doctype html>
<!-- headings.html                         Dave Reed -->
<!-- This page demos headings and horizontal lines.  -->
<!-- =============================================== -->

<html>
 <head>
   <title> Demo of headings </title>
 </head>

 <body>
   <h1>Main Title</h1>
     <p>This is an opening paragraph.</p>

   <hr>

   <h2>First Section</h2>
     <p>This is the first paragraph in the first section.</p>
     <p>This is the second paragraph in the first section.</p>
     <h3>Subsection</h3>
       <p>This paragraph appears in the subsection.</p>

   <hr>

   <h2>Second Section</h2>
     <p>This is the first paragraph in the second section.</p>
     <p>This is the second paragraph in the second section.</p>
 </body>
</html>
```



Main Title

This is an opening paragraph.

First Section

This is the first paragraph in the first section.

This is the second paragraph in the first section.

Subsection

This paragraph appears in the subsection.

Second Section

This is the first paragraph in the second section.

This is the second paragraph in the second section.

Styling Elements

Web browsers rely on user preferences when displaying a page

- each browser has a set of default (language, font, text size, color scheme), which can be reset by the user
- can override some of these defaults by adding STYLE attributes

an *attribute* is qualifier that can be added to an element in its opening tag

- the `style` attribute can be used to set style properties for an element

```
style="PROPERTY:VALUE"
```

e.g., can change the text color for an element by setting the `color` property

```
<p style="color:red"> Here is some red text. </p>
```

BODY Styling

when a `style` property is assigned to the body element, it applies to all elements embedded in the page

```
<body style="color:darkblue">
  ENTIRE PAGE APPEARS IN DARK BLUE TEXT
</body>
```

the `background-color` property can also be set for the entire page

```
<body style="background-color:lightgray">
  ENTIRE PAGE APPEARS WITH LIGHT GRAY BACKGROUND
</body>
```

can set multiple properties in the same `style` attribute

```
<body style="background-color:gray; color:white">
  ENTIRE PAGE APPEARS WITH GRAY BACKGROUND, WHITE TEXT
</body>
```

SPAN & DIV

in addition to p, the span and div elements are useful for grouping text

- span specifies a short span of text embedded in a paragraph
- div specifies a page division which groups multiple elements together

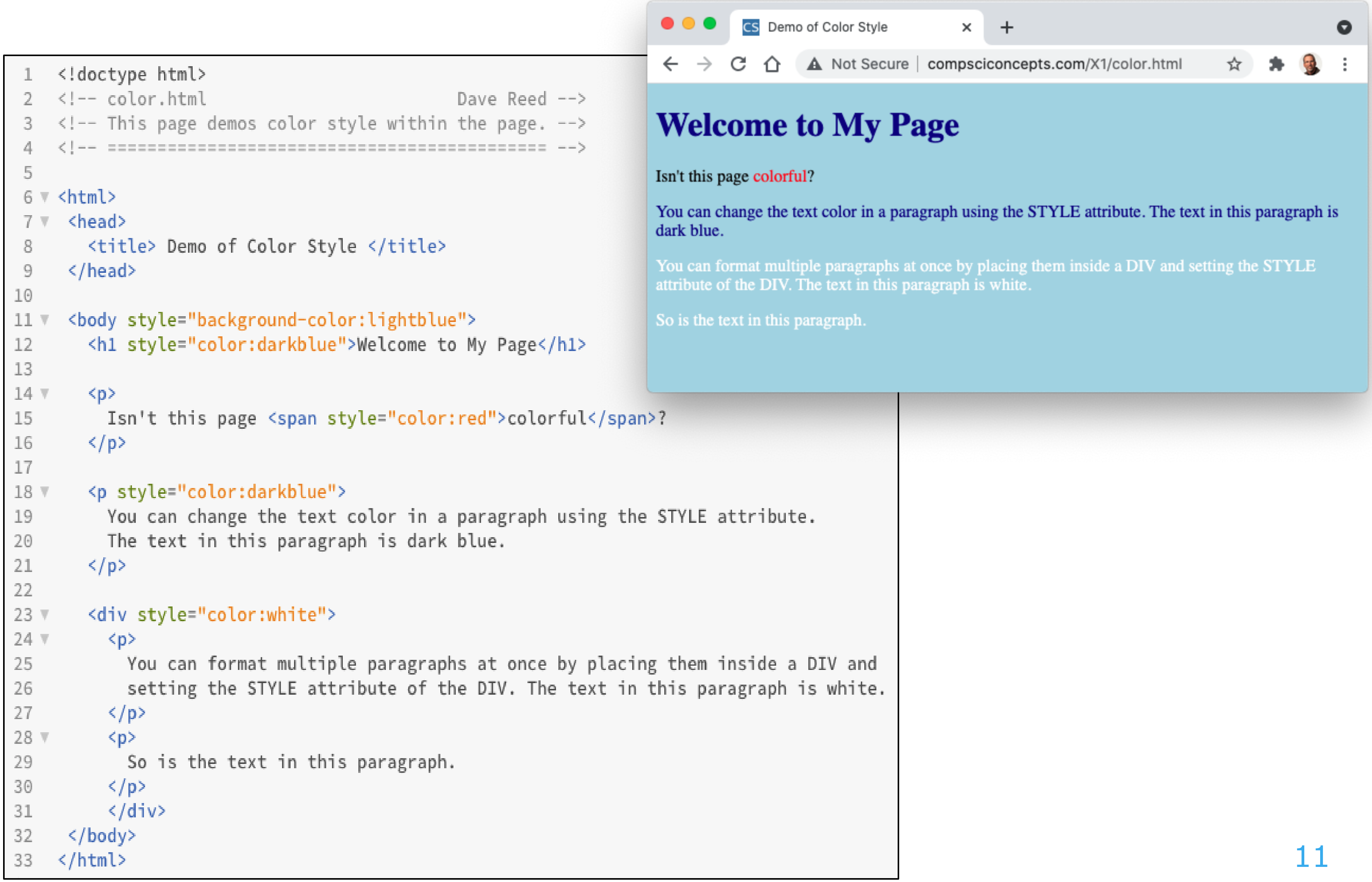
span can be used to embedded colored words or phrases

```
<p>Isn't this page <span style="color:red">colorful</span>?</p>
```

div can be used to group paragraphs and color them as one

```
<div style="color:white">
  <p>
    You can format multiple paragraphs at once by placing them
    inside a DIV and setting the STYLE attribute of the DIV.
  </p>
  <p>Both of these paragraphs will have white text.</p>
</div>
```

Color Styling Example



```
<!doctype html>
<!-- color.html                            Dave Reed -->
<!-- This page demos color style within the page. -->
<!-- ============================================ -->

<html>
 <head>
   <title> Demo of Color Style </title>
 </head>

 <body style="background-color:lightblue">
   <h1 style="color:darkblue">Welcome to My Page</h1>

   <p>
     Isn't this page <span style="color:red">colorful</span>?
   </p>

   <p style="color:darkblue">
     You can change the text color in a paragraph using the STYLE attribute.
     The text in this paragraph is dark blue.
   </p>

   <div style="color:white">
     <p>
       You can format multiple paragraphs at once by placing them inside a DIV and
       setting the STYLE attribute of the DIV. The text in this paragraph is white.
     </p>
     <p>
       So is the text in this paragraph.
     </p>
     </div>
 </body>
</html>
```

Welcome to My Page

Isn't this page colorful?

You can change the text color in a paragraph using the STYLE attribute. The text in this paragraph is dark blue.

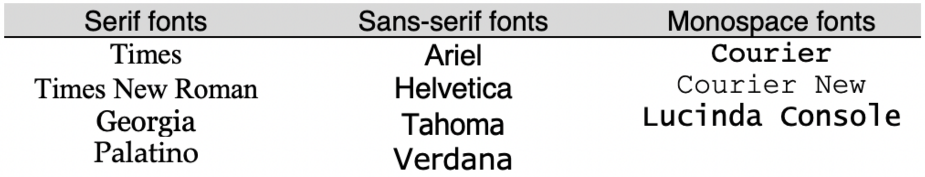
You can format multiple paragraphs at once by placing them inside a DIV and setting the STYLE attribute of the DIV. The text in this paragraph is white.

So is the text in this paragraph.

Font Styling

the font-family property can override the default font typeface

- must specify the font name and its family (as a backup)



Serif fonts	Sans-serif fonts	Monospace fonts
Times	Ariel	Courier
Times New Roman	Helvetica	Courier New
Georgia	Tahoma	Lucinda Console
Palatino	Verdana	

```
<p style="font-family:Helvetica, sans-serif">
  This text appears in Helvetica.
</p>

<p style="font-family:Times, serif">This text appears in Times.</p>
```

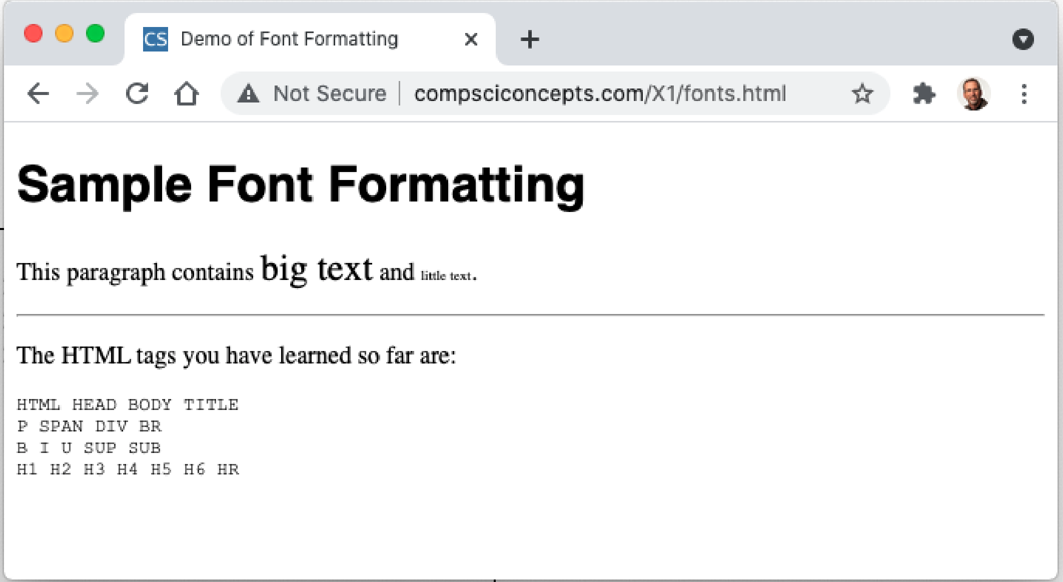
the font-size property can override the default size of the font

- can be absolute (in pixels) or relative to the current size (as percentage)

```
<span style="font-size:20px">This text appears 20 pixels tall.</span>

<p style="font-size:150%">This text is 50% larger than normal.</p>
```

Font Styling Example

```
<!doctype html>
<!-- fonts.html                 Dave Reed --
<!-- This page demos font formatting.     --
<!-- ==================================== --

<html>
 <head>
   <title> Demo of Font Formatting </title>
 </head>

 <body>
   <h1  style="font-family:Helvetica, sans-serif">Sample Font Formatting</h1>
   <p>This paragraph contains <span style="font-size:150%">big text</span>
      and <span style="font-size:50%">little text</span>.</p>

   <hr>

   <p>The HTML tags you have learned so far are:</p>
   <p style="font-family:Courier New, monospace; font-size:75%">
      HTML HEAD BODY TITLE<br>
      P SPAN DIV BR<br>
      B I U SUP SUB<br>
      H1 H2 H3 H4 H5 H6 HR
   </p>
 </body>
</html>
```


CS Demo of Font Formatting ×
+
Not Secure | compsciconcepts.com/X1/fonts.html

Sample Font Formatting

This paragraph contains big text and little text.

The HTML tags you have learned so far are:

```
HTML HEAD BODY TITLE
P SPAN DIV BR
B I U SUP SUB
H1 H2 H3 H4 H5 H6 HR
```

Alignment Styling

the `text-align` property can set the alignment of text elements

```
<h2 style="text-align:center">Centered Heading</h2>

<h2 style="text-align:right">Right-Justified Heading</h2>

<h2 style="text-align:justify">Left- and Right-Justified Heading</h2>
```

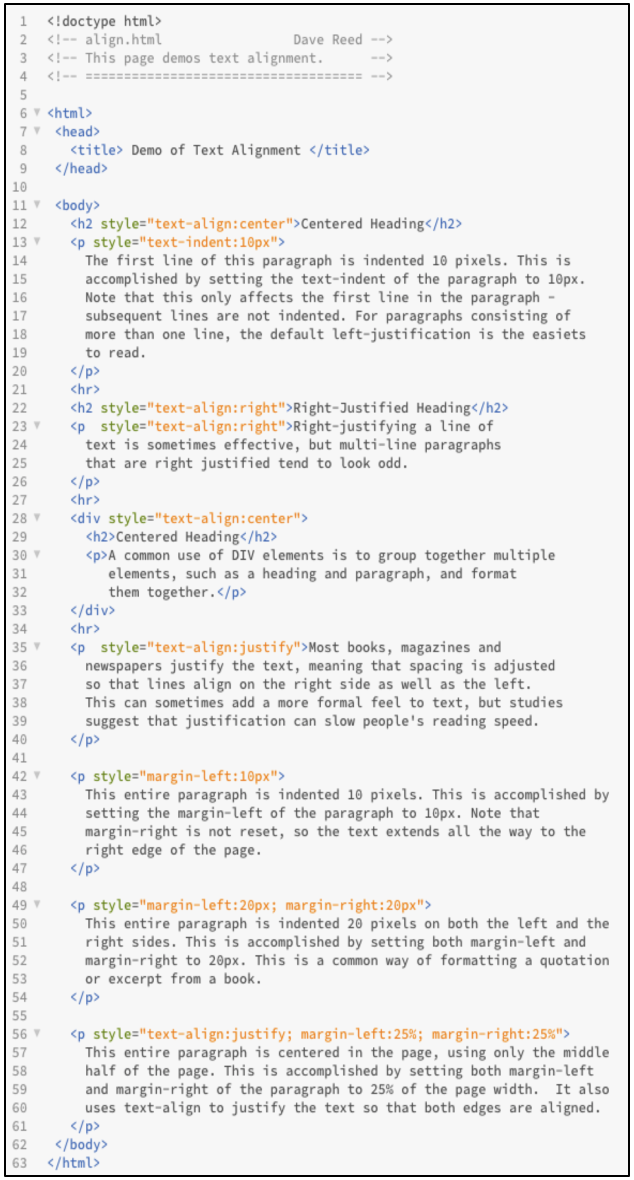
the `text-indent` property indents the first line of a paragraph

```
<p style="text-indent:10px">A paragraph with the 1st line indented...</p>
```

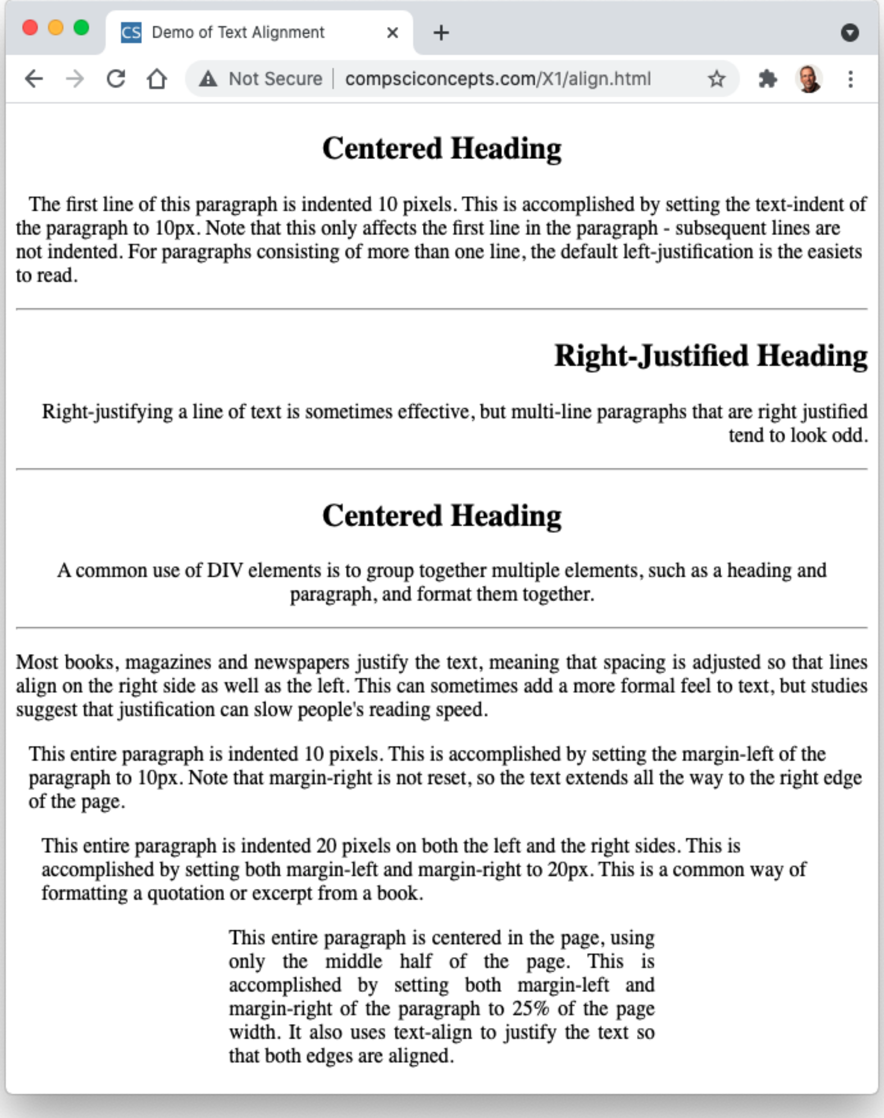
the `margin-left` and `margin-right` properties indent the entire paragraph

```
<p style="margin-left:10px">A paragraph with all lines indented...</p>
```

Alignment Styling Example



```
<!doctype html>
<!-- align.html                 Dave Reed -->
<!-- This page demos text alignment.      -->
<!-- ==================================== -->

<html>
 <head>
   <title> Demo of Text Alignment </title>
 </head>

 <body>
   <h2 style="text-align:center">Centered Heading</h2>
   <p style="text-indent:10px">
     The first line of this paragraph is indented 10 pixels. This is
     accomplished by setting the text-indent of the paragraph to 10px.
     Note that this only affects the first line in the paragraph -
     subsequent lines are not indented. For paragraphs consisting of
     more than one line, the default left-justification is the easiets
     to read.
   </p>
   <hr>
   <h2 style="text-align:right">Right-Justified Heading</h2>
   <p  style="text-align:right">Right-justifying a line of
     text is sometimes effective, but multi-line paragraphs
     that are right justified tend to look odd.
   </p>
   <hr>
   <div style="text-align:center">
     <h2>Centered Heading</h2>
     <p>A common use of DIV elements is to group together multiple
        elements, such as a heading and paragraph, and format
        them together.</p>
   </div>
   <hr>
   <p  style="text-align:justify">Most books, magazines and
     newspapers justify the text, meaning that spacing is adjusted
     so that lines align on the right side as well as the left.
     This can sometimes add a more formal feel to text, but studies
     suggest that justification can slow people's reading speed.
   </p>

   <p style="margin-left:10px">
     This entire paragraph is indented 10 pixels. This is accomplished by
     setting the margin-left of the paragraph to 10px. Note that
     margin-right is not reset, so the text extends all the way to the
     right edge of the page.
   </p>

   <p style="margin-left:20px; margin-right:20px">
     This entire paragraph is indented 20 pixels on both the left and the
     right sides. This is accomplished by setting both margin-left and
     margin-right to 20px. This is a common way of formatting a quotation
     or excerpt from a book.
   </p>

   <p style="text-align:justify; margin-left:25%; margin-right:25%">
     This entire paragraph is centered in the page, using only the middle
     half of the page. This is accomplished by setting both margin-left
     and margin-right of the paragraph to 25% of the page width.  It also
     uses text-align to justify the text so that both edges are aligned.
   </p>
 </body>
</html>
```



Centered Heading

The first line of this paragraph is indented 10 pixels. This is accomplished by setting the text-indent of the paragraph to 10px. Note that this only affects the first line in the paragraph - subsequent lines are not indented. For paragraphs consisting of more than one line, the default left-justification is the easiets to read.

Right-Justified Heading

Right-justifying a line of text is sometimes effective, but multi-line paragraphs that are right justified tend to look odd.

Centered Heading

A common use of DIV elements is to group together multiple elements, such as a heading and paragraph, and format them together.

Most books, magazines and newspapers justify the text, meaning that spacing is adjusted so that lines align on the right side as well as the left. This can sometimes add a more formal feel to text, but studies suggest that justification can slow people's reading speed.

This entire paragraph is indented 10 pixels. This is accomplished by setting the margin-left of the paragraph to 10px. Note that margin-right is not reset, so the text extends all the way to the right edge of the page.

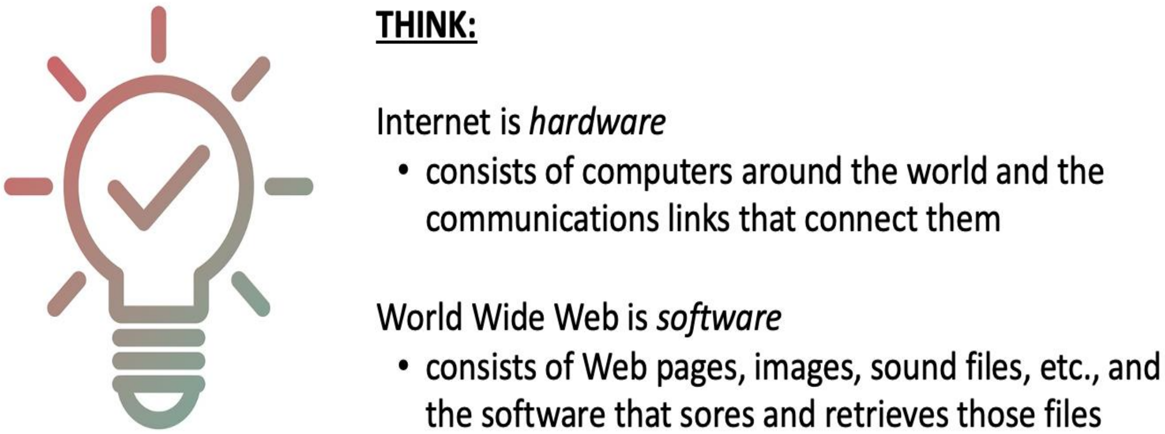
This entire paragraph is indented 20 pixels on both the left and the right sides. This is accomplished by setting both margin-left and margin-right to 20px. This is a common way of formatting a quotation or excerpt from a book.

This entire paragraph is centered in the page, using only the middle half of the page. This is accomplished by setting both margin-left and margin-right of the paragraph to 25% of the page width. It also uses text-align to justify the text so that both edges are aligned.

Web ≠ Internet

people often confuse the Web and the Internet – they are not the same!

- Internet was created in 1969; World Wide Web was created in 1990



THINK:

Internet is *hardware*

- consists of computers around the world and the communications links that connect them

World Wide Web is *software*

- consists of Web pages, images, sound files, etc., and the software that sores and retrieves those files

the Internet could exist without the Web

- and did, in fact, for many years (applications included email and news groups)

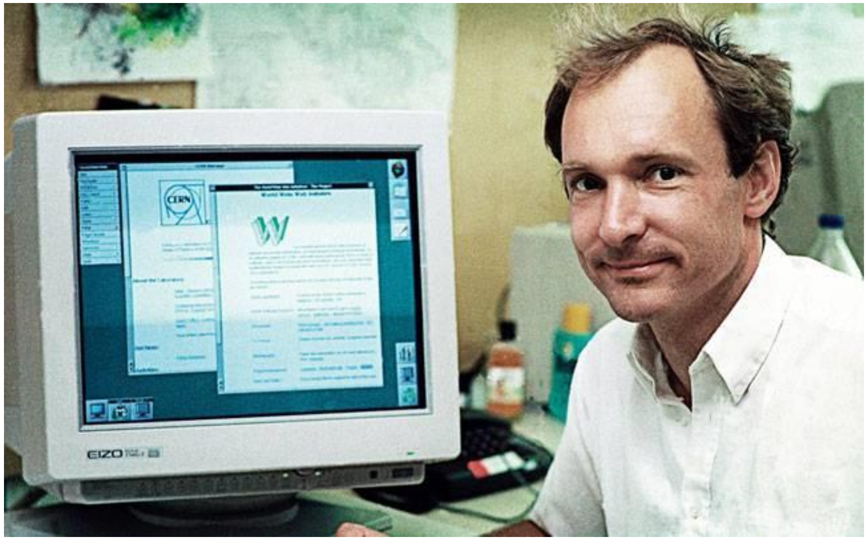
the Web couldn't exist without the Internet

- the Internet is the hardware that stores and executes the Web software

History of the Web

World Wide Web was invented by Tim Berners-Lee

- researcher from 1984-1994 at the European Laboratory for Particle Physics (CERN)
- founded and serves as director of the World Wide Web Consortium (W3C)
- knighted by Queen Elizabeth in 2004



CERN researchers were spread across Europe, but needed to collaborate

- in 1989, Berners-Lee devised a system that would allow them to freely exchange data, regardless of location or computer type

his design integrated two key ideas

1. hypertext (documents with interlinked text and media)
   - Web pages can contain images and links to other pages
2. the distributed nature of the Internet
   - pages can be stored on machines all across the Internet
   - logical connections between pages are independent of physical locations

Web Timeline

1990: Berners-Lee produced working prototypes of a Web server and browser

1991: Berners-Lee made his software available for free over the Internet

1993: Marc Andreesen and Eric Bina at NCSA wrote the first graphical browser: Mosaic

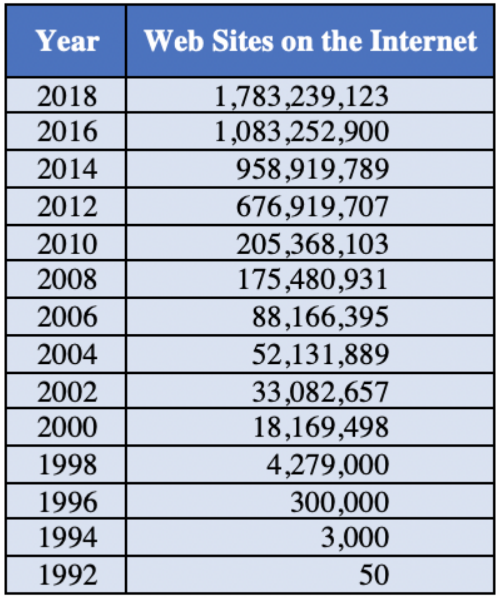
- Mosaic integrated text, image & links, made browsing more intuitive

1994: Andreesen founded Netscape, which marketed the Netscape Navigator

1995: Microsoft released Internet Explorer → the browser wars begin!

1999: Internet Explorer becomes the most popular browser (~90% of market in 2002)

2021: Google Chrome has ~63% of market, then Safari at 19%, Mozilla Firefox at 4%



Year	Web Sites on the Internet
2018	1,783,239,123
2016	1,083,252,900
2014	958,919,789
2012	676,919,707
2010	205,368,103
2008	175,480,931
2006	88,166,395
2004	52,131,889
2002	33,082,657
2000	18,169,498
1998	4,279,000
1996	300,000
1994	3,000
1992	50

in 2019, Google claimed to have indexed more than 130 trillion pages

others estimate the number of Web pages could be in the hundreds of quadrillions

Search Engines

as the Web grew, it became difficult to find resources

- in general, you needed to somehow know the address of a page to access it

manually generated index sites appeared in the early 1990s

- provided lists of popular Web sites, organized by topic or alphabetically
- these were not scalable as the Web exploded in size

the first Web search engines appeared in the mid 1990s

- used software called Web crawlers, or spiders, to surf the Web, indexing pages
- enabled users to search those indexed pages via search words or phrases

unfortunately, the quality of early searches was not very good

- the search for a word/phrase might return unrelated or unreliable pages

the Google search engine began in 1996 as a research project by Stanford grad students Larry Page and Sergey Brin

- their goal was to create an easy-to-use search engine that produced high-quality results
- founded Google Inc. in 1998

Google

at the heart of Google's performance is the PageRank algorithm

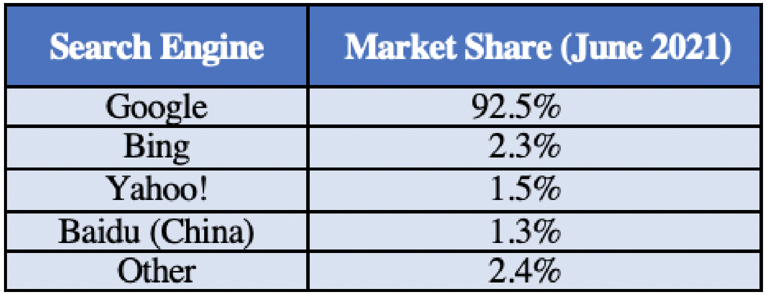
- ranks pages based on their perceived value and trustworthiness
- if a page is linked to by many other pages, that suggests that many people find its contents valuable and trustworthy
- moreover, the more valued/trusted those linking pages are, the more impact their links will have

Brin & Page also revolutionized how browsers made money

- they sold targeted adds and charged based on clickthrough
- e.g., a shoe store could purchase adds for when a user entered "shoe" or "footwear" as search terms, and would be charged based on how often users clicked on the ads

Brin & Page donated the patent for the PageRank algorithm to Stanford

- licensed its use back for $336 million in stock



Search Engine	Market Share (June 2021)
Google	92.5%
Bing	2.3%
Yahoo!	1.5%
Baidu (China)	1.3%
Other	2.4%

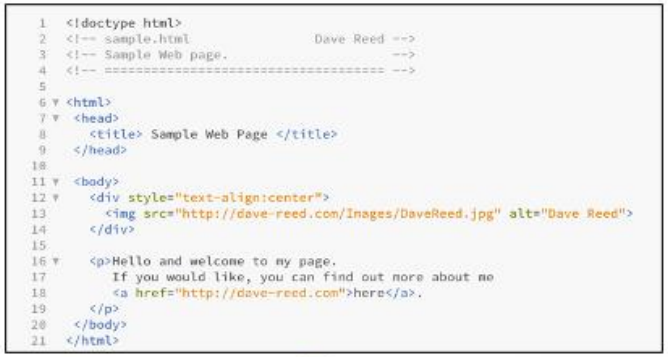
Google dominates the search market

- performed 5.6 billion searches in 2020
- that's 63,000 searches per second!

Viewing a Web Page

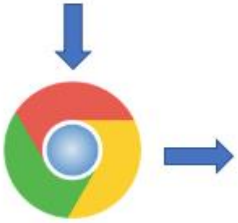
a *Web page* is a text document that contains additional formatting information in a language called HTML (HyperText Markup Language)

a *Web browser* is a program that accesses a Web page, interprets its content, and displays the page

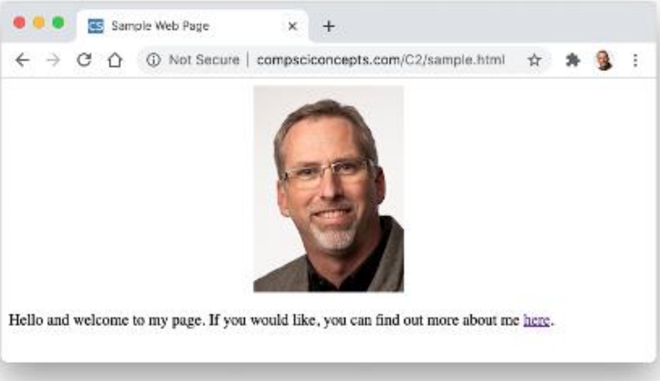


```
<!doctype html>
<!-- sample.html                    Dave Reed -->
<!-- Sample Web page.                         -->
<!-- ==================================== -->

<html>
 <head>
   <title> Sample Web Page </title>
 </head>

 <body>
   <div style="text-align:center">
     <img src="http://dave-reed.com/Images/DaveReed.jpg" alt="Dave Reed">
   </div>

   <p>Hello and welcome to my page.
      If you would like, you can find out more about me
      <a href="http://dave-reed.com">here</a>.
   </p>
 </body>
</html>
```

A Web page is a text document that contains HTML formatting.



A Web browser (e.g., Chrome) is a program that interprets the HTML and displays the page.



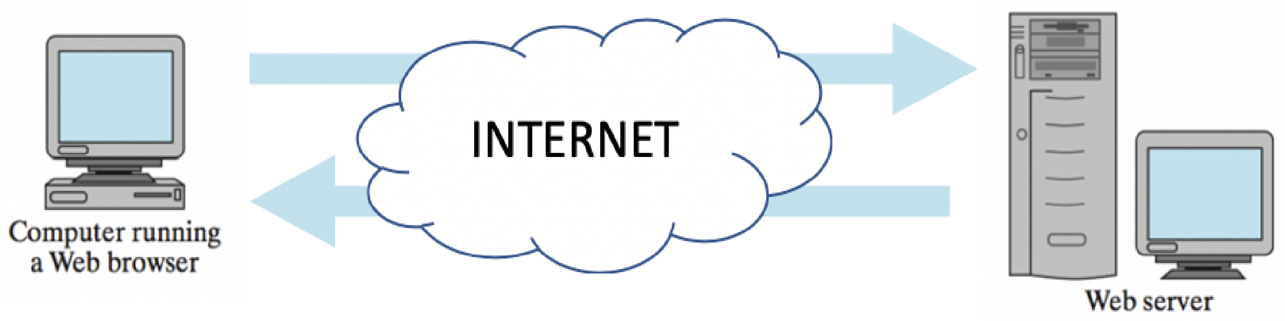
Hello and welcome to my page. If you would like, you can find out more about me here.

Web Server

a *Web server* is an Internet-enabled computer that executes software for providing access to certain Web document

- it stores Web pages and files (images, videos, …) and sends them to browsers who request them

1. When the user clicks on a link, the browser sends a request to the server for that page.

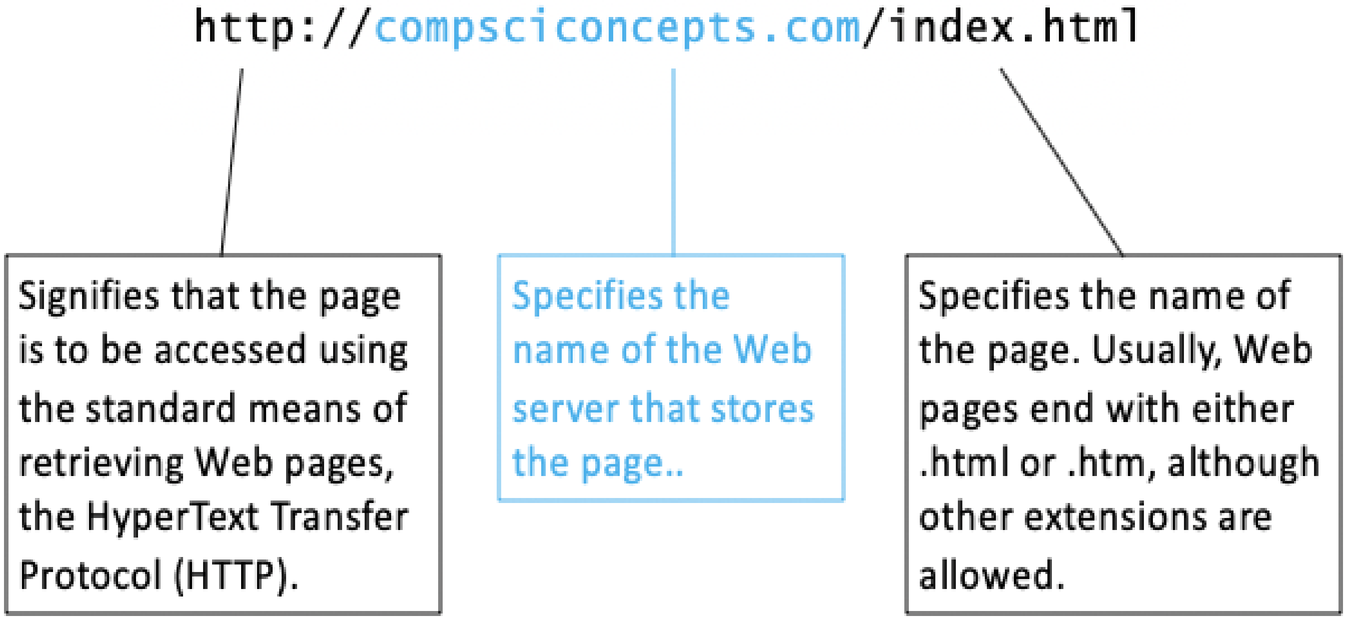


2. The server locates the page and sends it to the browser for display.

Web Addresses

Web pages require uniform names to locate and identify them uniquely

- each page is assigned a *Uniform Resource Locator (URL)*
- URL's are commonly referred to as *Web addresses*
- the different parts of the Web address provide information for locating the page



http://compsciconcepts.com/index.html

Signifies that the page is to be accessed using the standard means of retrieving Web pages, the HyperText Transfer Protocol (HTTP).

Specifies the name of the Web server that stores the page..

Specifies the name of the page. Usually, Web pages end with either .html or .htm, although other extensions are allowed.

Viewing Local Web Pages

a Web browser can be used to view pages stored on the same computer

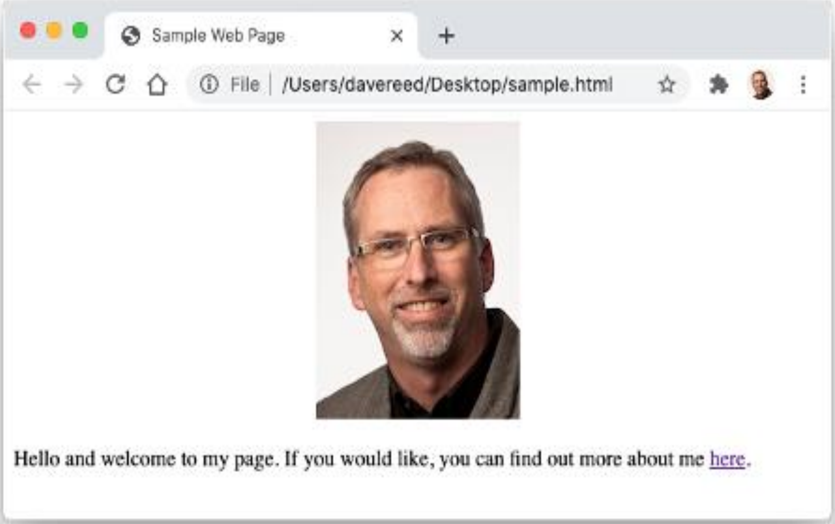
- can go through the File menu to select the local page, or
- can enter the File location in the address box (without the `http` prefix)

this feature is handy when developing Web pages

- can create a Web page and view it in the browser before uploading to a server

Note: "File" in the Address Box identifies the Web page as a locally stored file.

In this case, the file sample.html is stored on the Desktop of davereed's computer.



Hello and welcome to my page. If you would like, you can find out more about me here.

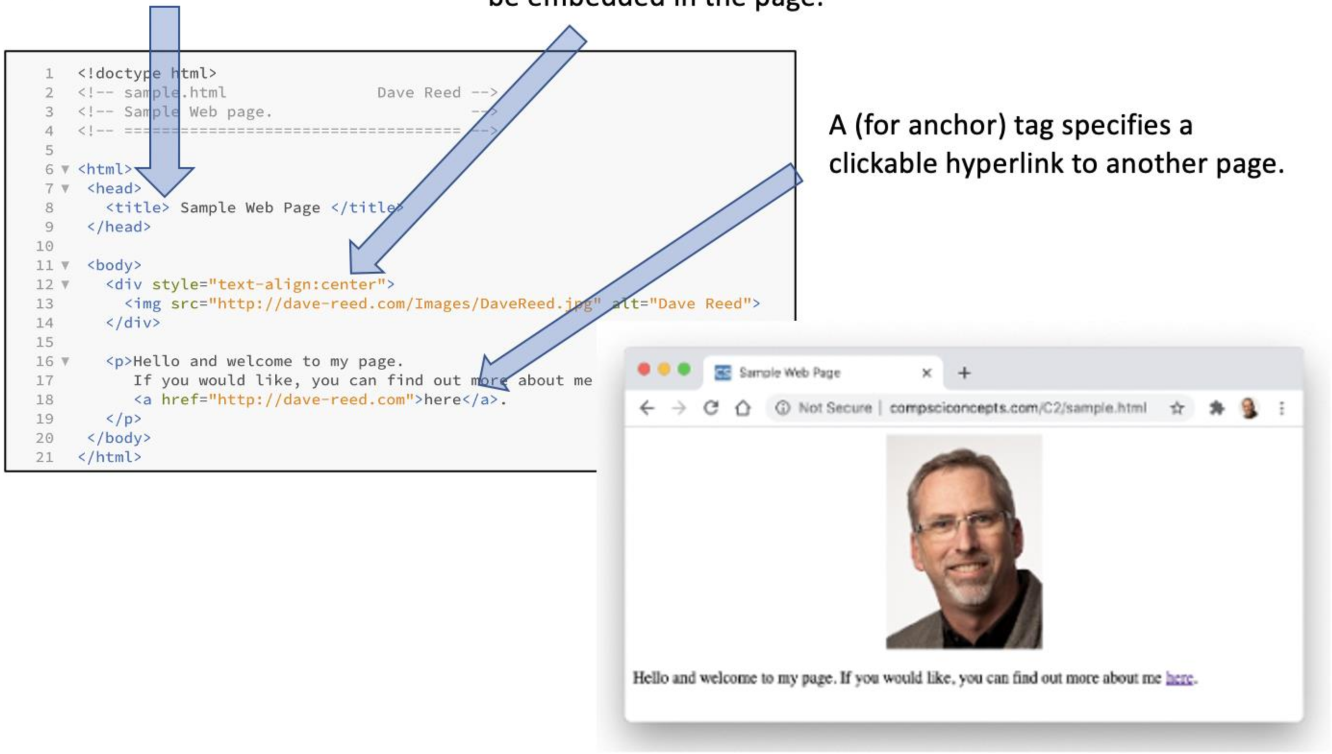
Web Protocols: HTML

HyperText Markup Language (HTML) utilizes tags to markup page contents

- these tags tell the browser how to display the contents
- HTML5 is the current standard, supported by all browsers

TITLE tags specify the title that appears in the browser tab.

IMG tag specifies an image to be embedded in the page.



```
<!doctype html>
<!-- sample.html                 Dave Reed -->
<!-- Sample Web page.                      -->
<!-- ===================================== -->

<html>
 <head>
   <title> Sample Web Page </title>
 </head>

 <body>
   <div style="text-align:center">
     <img src="http://dave-reed.com/Images/DaveReed.jpg" alt="Dave Reed">
   </div>

   <p>Hello and welcome to my page.
      If you would like, you can find out more about me
      <a href="http://dave-reed.com">here</a>.
   </p>
 </body>
</html>
```

A (for anchor) tag specifies a clickable hyperlink to another page.

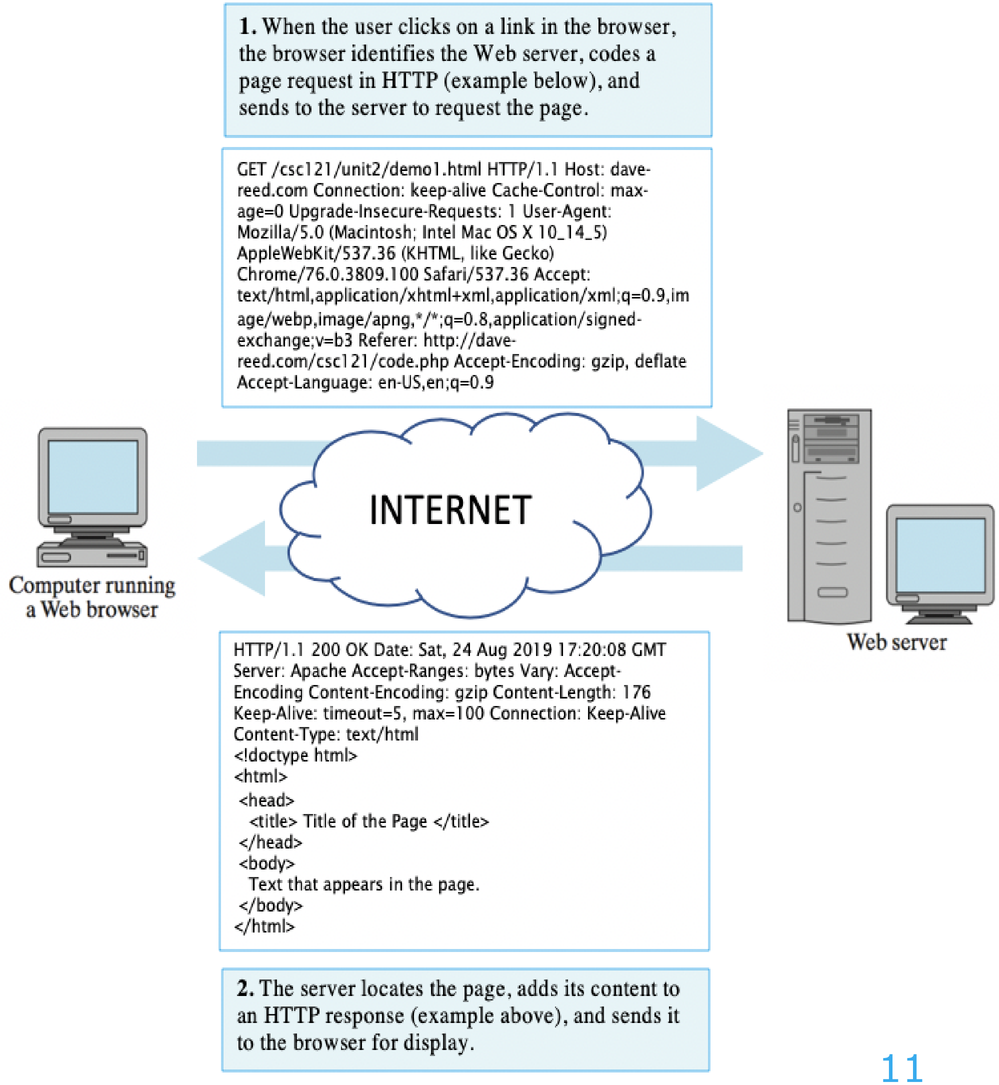
Hello and welcome to my page. If you would like, you can find out more about me here.

Web Protocols: HTTP

HyperText Transfer Protocol (HTTP) defines how messages between browsers and servers are formatted

- the prefix `http://` in a URL specifies that the HTTP protocol is to be used in communicating with the server

- the prefix `https://` is similarly used for secure (encrypted) HTTP communications



1. When the user clicks on a link in the browser, the browser identifies the Web server, codes a page request in HTTP (example below), and sends to the server to request the page.

```
GET /csc121/unit2/demo1.html HTTP/1.1 Host: dave-
reed.com Connection: keep-alive Cache-Control: max-
age=0 Upgrade-Insecure-Requests: 1 User-Agent:
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/76.0.3809.100 Safari/537.36 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,im
age/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3 Referer: http://dave-
reed.com/csc121/code.php Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

```
HTTP/1.1 200 OK Date: Sat, 24 Aug 2019 17:20:08 GMT
Server: Apache Accept-Ranges: bytes Vary: Accept-
Encoding Content-Encoding: gzip Content-Length: 176
Keep-Alive: timeout=5, max=100 Connection: Keep-Alive
Content-Type: text/html
<!doctype html>
<html>
 <head>
   <title> Title of the Page </title>
 </head>
 <body>
   Text that appears in the page.
 </body>
</html>
```

2. The server locates the page, adds its content to an HTTP response (example above), and sends it to the browser for display.

Browser caching

for efficiency reasons, browsers will sometimes *cache* pages/images

- the browser reserves space (a cache folder) on the user's computer
- to avoid redundant downloads, the browser will store a copy of a page/image in that reserved space (along with a timestamp)
- the next time the page/image is requested, browser will send a timestamp of the cached copy
    - the server compares that timestamp with one of the stored document
        - if cached copy is newer, then response says to use it
        - if server copy is newer, then response includes the new version

in general, browsers are not allowed to access/modify user files

- this is for safety – you don't want to visit a Web site and risk having your files copied or damaged
- caching is a loophole (can only access/modify files in cache folder)

NOTE: caching still requires the browser to contact the server

- but only have to download the page if it has changed since last cached

Cookies

another loophole is cookies

a cookie is a small file that can be stored and accessed by a Web server

- similar to caching, browsers reserve space (a cookie folder) on the user's computer
- when the user visits a site, the Web server is allowed to store a small amount of data in a cookie file (e.g., date & time of visit, items purchased)
- when the user returns to that site, the Web server can retrieve any cookies it previously stored
- NOTE: only the site that stored the cookie is able to retrieve it

cookies can improve the user's experience, but can also be intrusive

- most browsers enable the user to control the use of cookies

Static vs. Dynamic Pages

recall: a Web page uses HTML tags to identify page content and formatting information

HTML can produce only *static pages*

- static pages look the same and behave in the same manner each time they are loaded into a browser

in 1995, researchers at Netscape developed JavaScript, a language for creating *dynamic pages*

- Web pages with JavaScript can change their appearance:
  - over time (e.g., a different image each time that a page is loaded), or
  - in response to a user's actions (e.g., typing, mouse clicks, and other input methods)

Programming Languages

JavaScript is a *programming language*

- a *programming language* is a language for specifying instructions that a computer can execute
- each *statement* in a programming language specifies a particular action that the computer is to carry out

  (e.g., changing an image or opening a window when a button is clicked)

some programming languages are general-purpose

- popular languages include C++, Java, J#

JavaScript was defined for a specific purpose: *adding dynamic content to Web pages*

- can associate JavaScript statements with certain HTML elements so that they react to actions taken by the user (e.g., a button click)

ID Attributes

in order for an element to behave dynamically, it must have an ID attribute

- ID is assigned a unique identifier by which that element can be accessed and changed

```
<img id="familyImg" src="Images/beach.jpg" alt="My Family">
```

an identifier should start with a lowercase letter, consist of letters and digits

e.g., `familyImg`   `mysteryImg`   `outputSpan`   `num1Box`

once an element has an ID, it can be accessed and altered using dynamic attributes known as *event handlers*

- the ONMOUSEOVER attribute specifies the action(s) to take place when the mouse is moved over the element
- the ONMOUSEOUT attribute specifies specifies the action(s) to take place when the mouse is moved off the element
- the actions are encoded as statements in the JavaScript language

Event Handler Attributes

for example, can have an image that reacts to mouse movements:

```
<img src="ADDRESS_OF_IMAGE" alt="DESCRIPTIVE_TEXT"
     onmouseover="CODE_TO_EXECUTE_WHEN_MOUSE_GOES_OVER_IMAGE"
     onmouseout="CODE_TO_EXECUTE_WHEN_MOUSE_LEAVES_IMAGE">
```

the simplest type of action is changing the value of an element's attribute

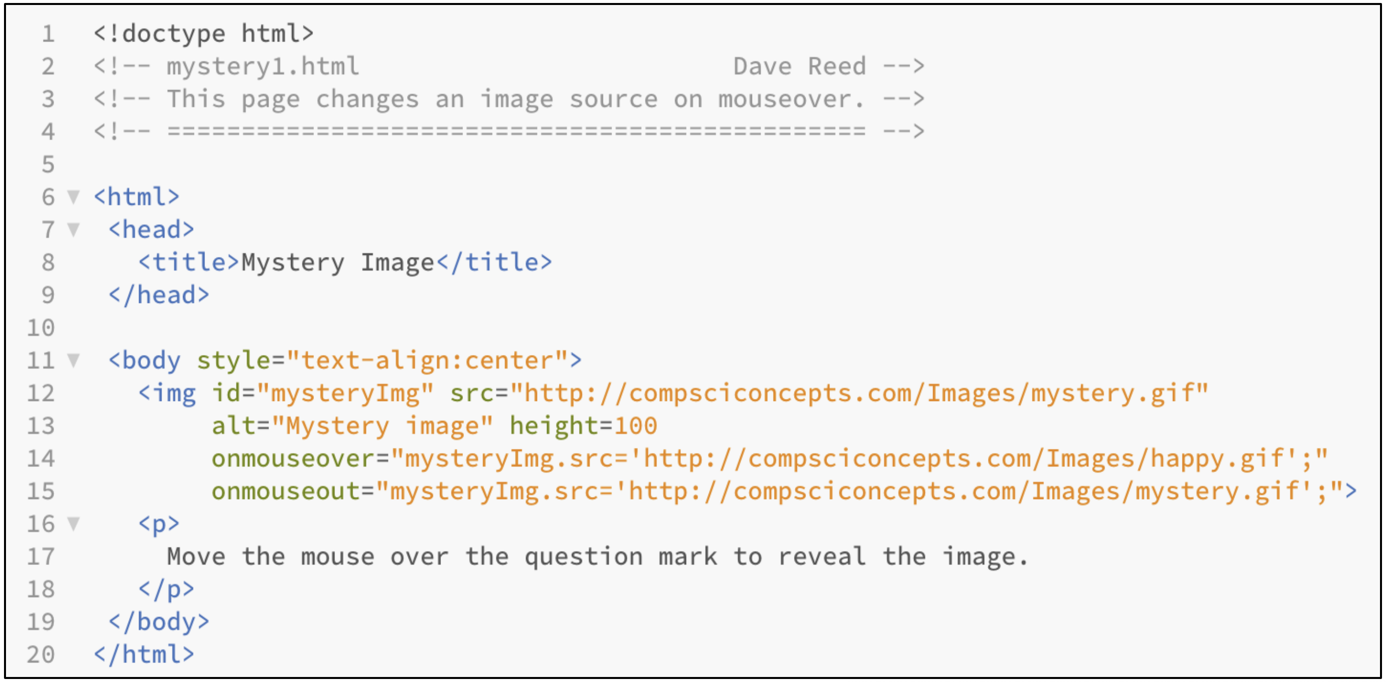
- this is accomplished via a JavaScript *assignment statement*

```
ELEMENT_ID.ATTRIBUTE_NAME = NEW_ATTRIBUTE_VALUE;
```

for example, the following JavaScript assignment will change the SRC attribute of the element with ID `mysteryImg`

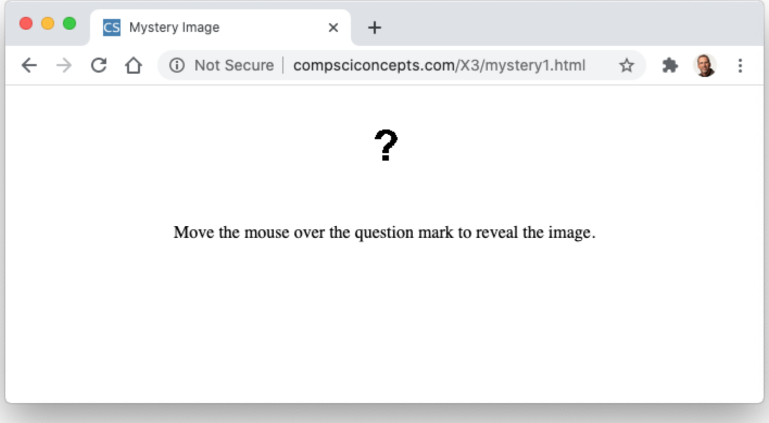
```
mysteryImg.src='http://compsciconcepts.com/Images/happy.gif';
```

Mystery Image Page



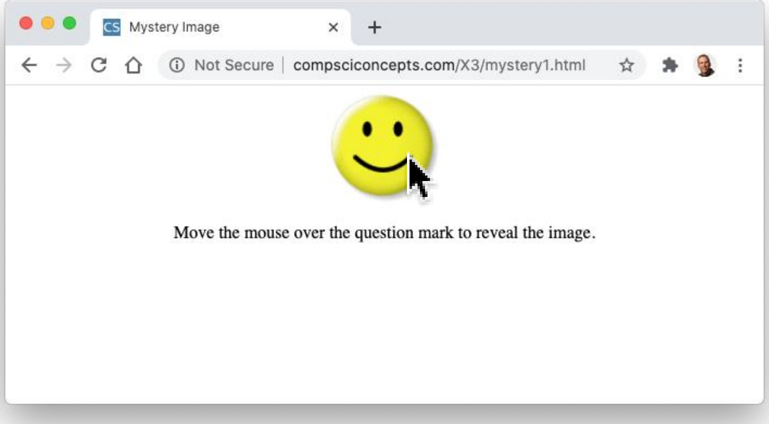
```
<!doctype html>
<!-- mystery1.html                         Dave Reed -->
<!-- This page changes an image source on mouseover. -->
<!-- =============================================== -->

<html>
 <head>
   <title>Mystery Image</title>
 </head>

 <body style="text-align:center">
   <img id="mysteryImg" src="http://compsciconcepts.com/Images/mystery.gif"
        alt="Mystery image" height=100
        onmouseover="mysteryImg.src='http://compsciconcepts.com/Images/happy.gif';"
        onmouseout="mysteryImg.src='http://compsciconcepts.com/Images/mystery.gif';">
   <p>
     Move the mouse over the question mark to reveal the image.
   </p>
 </body>
</html>
```

- initially, the image displays a '?'
- when mouse moves over, SRC attribute is assigned to happy face
- when mouse leaves, SRC attribute is assigned back to '?'



?

Move the mouse over the question mark to reveal the image.



Move the mouse over the question mark to reveal the image.

Strings & Syntax Errors

a *string literal* (or just *string*) is a sequence of characters enclosed in quotes

- to avoid confusion, we will always use double quotes for HTML strings; single quotes for JavaScript strings

```
<img id="mysteryImg" src="mystery.gif" alt="Mystery image"
     onmouseover="mysteryImg.src='happy.gif';"
     onmouseout="mysteryImg.src='mystery.gif';">
```

syntax errors are errors in the format of HTML or JavaScript statements

- for example, misspelling an element ID in a JavaScript assignment:

```
mysteryimg.src='http://compsciconcepts.com/Images/happy.gif';
```

unlike HTML syntax errors (which are largely ignored by the browser, JavaScript syntax errors often just fail

- browsers produce error messages that help to identify JavaScript errors
- in Google Chrome, error messages appear in the JavaScript Console:

  `View menu → Developer → JavaScript Console`
- when a page fails to behave as expected, CHECK THE ERROR MESSAGES!

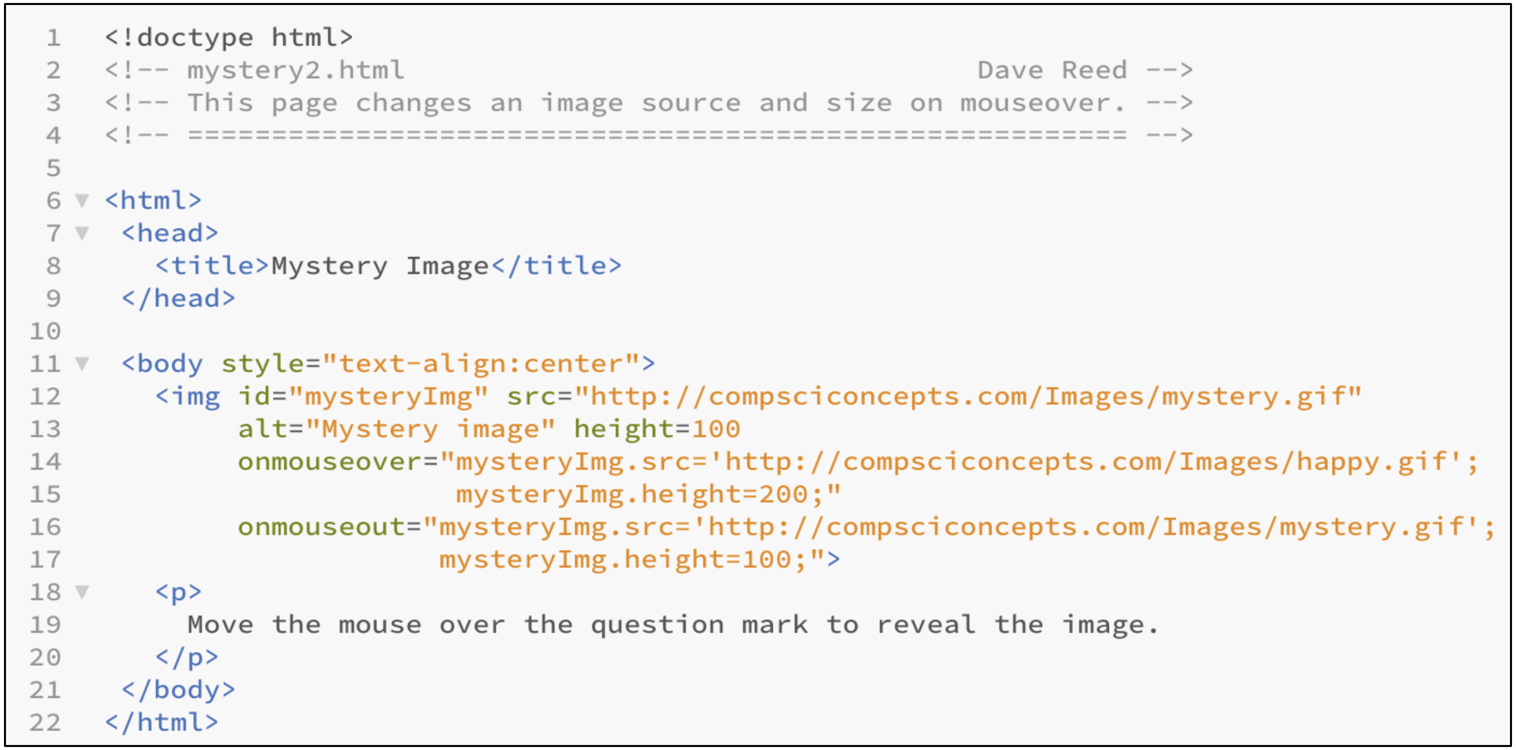
Multiple Actions

JavaScript assignments can similarly change other element attributes

e.g., can change an images height: `mysteryImg.height = 200;`

if desired, an event handler can perform multiple actions (separated by ;)

- in this example, both the SRC and HEIGHT change on mouse events



```
<!doctype html>
<!-- mystery2.html                                  Dave Reed -->
<!-- This page changes an image source and size on mouseover. -->
<!-- ======================================================== -->

<html>
 <head>
   <title>Mystery Image</title>
 </head>

 <body style="text-align:center">
   <img id="mysteryImg" src="http://compsciconcepts.com/Images/mystery.gif"
        alt="Mystery image" height=100
        onmouseover="mysteryImg.src='http://compsciconcepts.com/Images/happy.gif';
                     mysteryImg.height=200;"
        onmouseout="mysteryImg.src='http://compsciconcepts.com/Images/mystery.gif';
                    mysteryImg.height=100;">
   <p>
     Move the mouse over the question mark to reveal the image.
   </p>
 </body>
</html>
```

Interaction via Buttons

a button is an HTML element that appears as a labeled rectangle or oval

- usually associated with the ONCLICK event handler attribute, which specifies the action to take place when the button is clicked

general form:

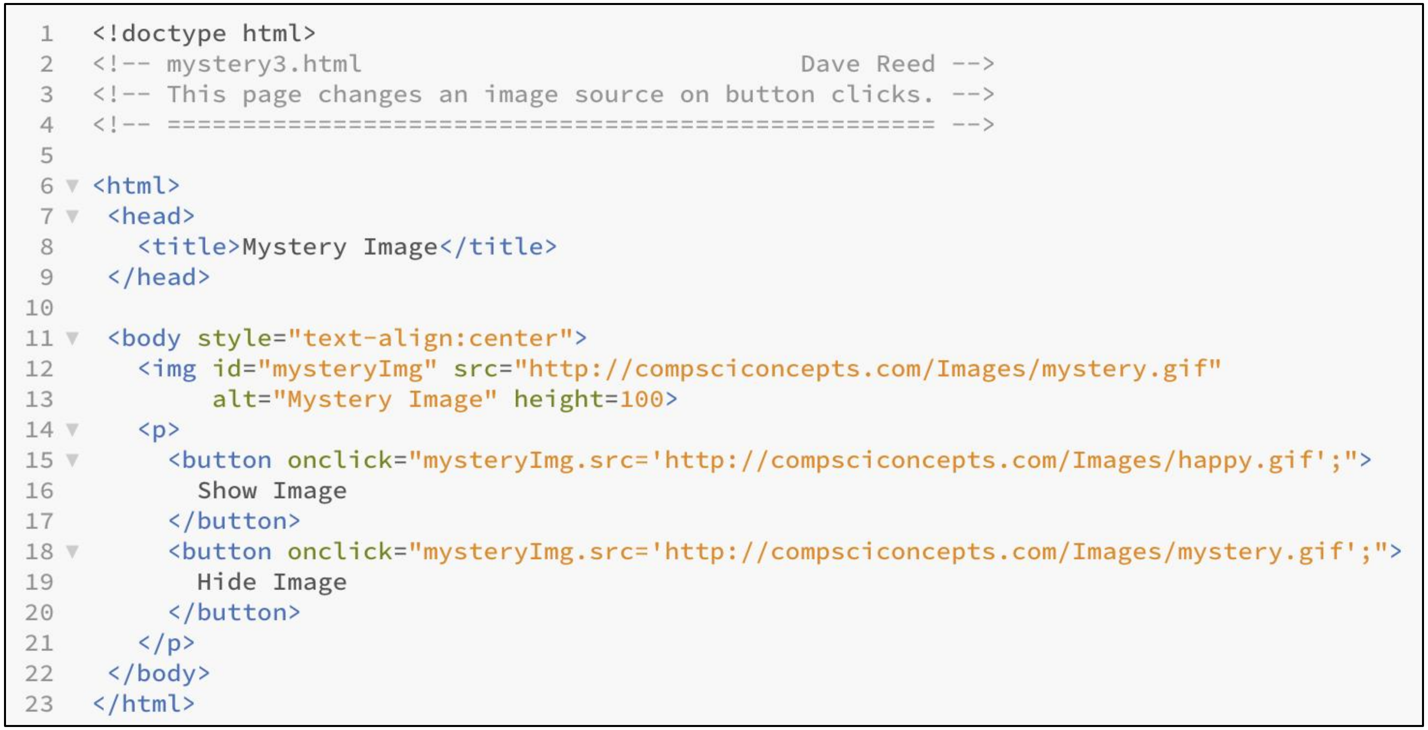
```
<button onclick="CODE_TO_BE_EXECUTED_WHEN_MOUSE_CLICKS_ON_BUTTON">
  BUTTON_LABEL
</button>
```

typically, buttons are used to initiate actions on other elements

- e.g., click on a button to change the `src` or `height`/`width` of an `img`

```
<button onclick="mysteryImg.height=200;">Expand the Image</button>
```

Button Example

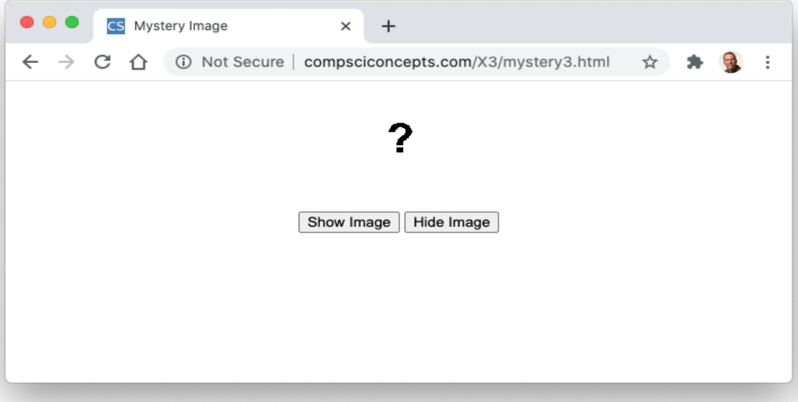


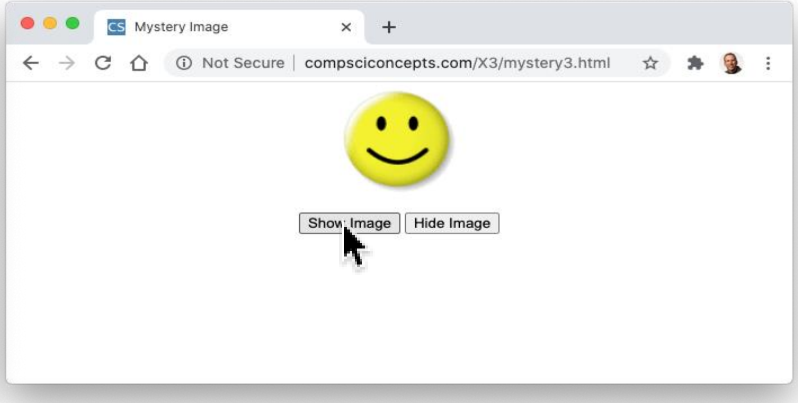
```
<!doctype html>
<!-- mystery3.html                             Dave Reed -->
<!-- This page changes an image source on button clicks. -->
<!-- =================================================== -->

<html>
 <head>
   <title>Mystery Image</title>
 </head>

 <body style="text-align:center">
   <img id="mysteryImg" src="http://compsciconcepts.com/Images/mystery.gif"
        alt="Mystery Image" height=100>
   <p>
     <button onclick="mysteryImg.src='http://compsciconcepts.com/Images/happy.gif';">
       Show Image
     </button>
     <button onclick="mysteryImg.src='http://compsciconcepts.com/Images/mystery.gif';">
       Hide Image
     </button>
   </p>
 </body>
</html>
```

image initially displays a question mark

when Show Image button is clicked, the image changes to 🙂

when Hide Image button is clicked, it changes back to ?





Dynamic Text

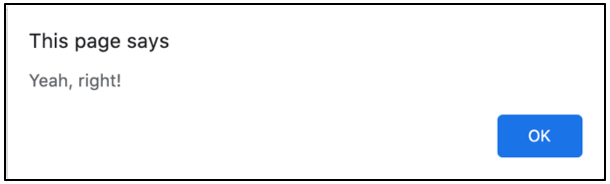
to display text within a page, there are 2 main options

1. `alert` statement: will display a simple text message in a separate alert window
2. `innerHTML` attribute: can display text directly in the page

general form of an alert statement: `alert('MESSAGE');`

- when executed, it opens a separate window displaying that message

```
<button onclick="alert('Yeah, right.');">
  Click for free money!
</button>
```



This page says

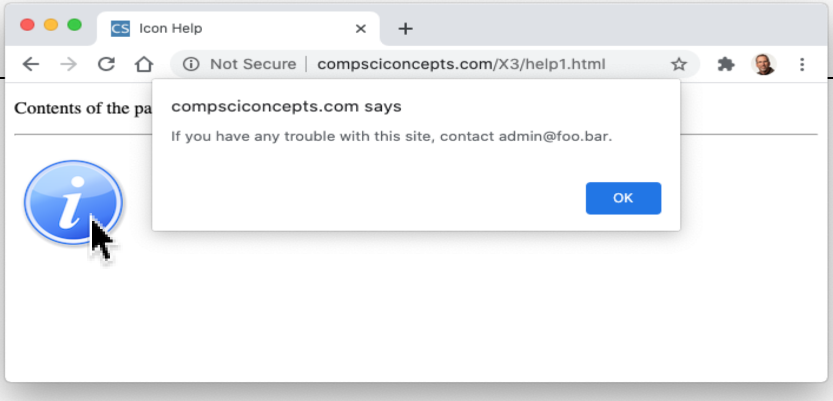
Yeah, right!

- `alert` statements are useful when you want to display a short (1-line) message
- the messages are limited, in that they can't include any HTML tags
- can be annoying to the user since the pop-up window must be manually closed

note: if a message contains an apostrophe, must use backslash (escape character) to distinguish it from the ending quote: `alert('I\'m happy you are here.');`

Help Page

```html
<!doctype html>
<!-- help1.html                                       Dave Reed -->
<!-- Web page that displays a help message in an alert window. -->
<!-- ========================================================= -->

<html>
 <head>
   <title>Icon Help</title>
 </head>

 <body>
   <p>
     Contents of the page.
   </p>
   <hr>
   <p>
     <img src="http://compsciconcepts.com/Images/information.png"
          alt="Information icon"
          onclick="alert('If you have any trouble with this site, ' +
                         'contact admin@foo.bar.');">
   </p>
 </body>
</html>
```



compsciconcepts.com says

If you have any trouble with this site, contact admin@foo.bar.

when the mouse clicks on the image, an alert window is opened, displaying the message

note: the user must click OK to close the window

innerHTML

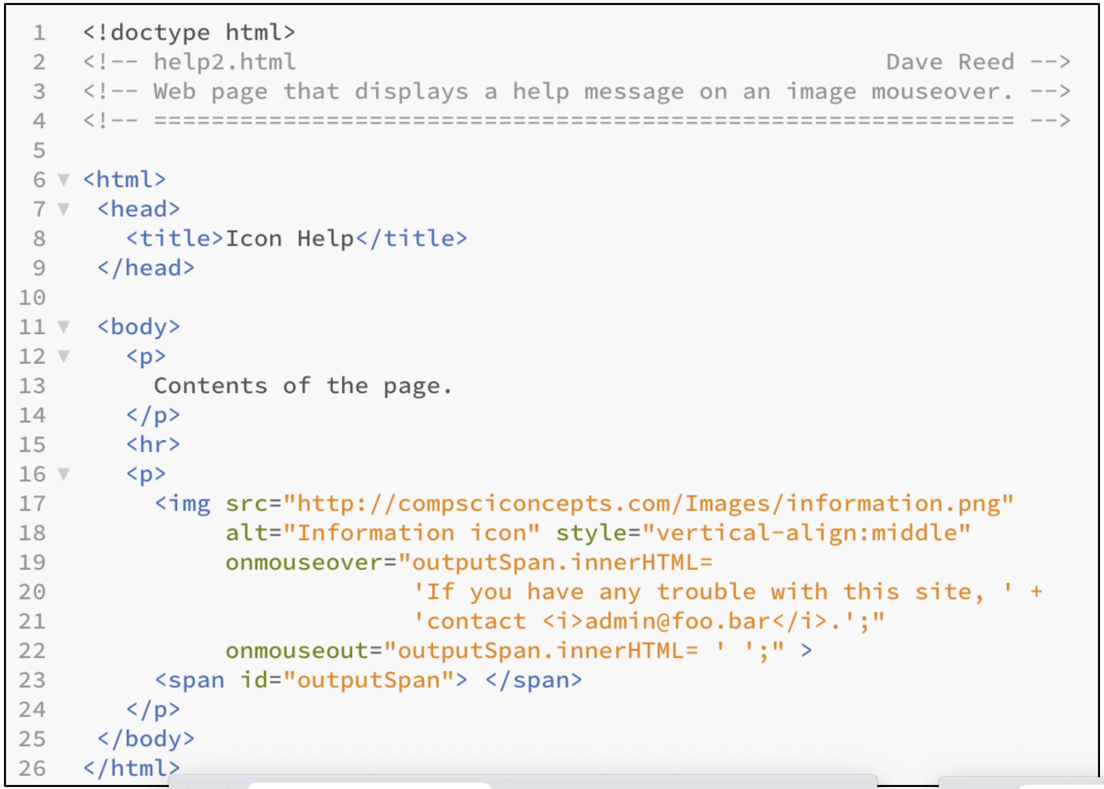
better yet, embed the text directly in the page

text-based elements (p, span, div) have an innerHTML attribute

- can be used to access or change the text within that element
- be careful: the capitalization must be exact

```
outputSpan.innerHTML = 'Hello';


outputDiv.innerHTML =
  '<p>You can write long messages that are embedded directly ' +
  'in the page. You can even add <i>HTML formatting</i> to the ' +
  'text. </p> <p>The contents of this dynamic DIV element is ' +
  'being assigned multiple paragraphs.</p>';


outputP.innerHTML = outputP.innerHTML + '!';
```

Help Page

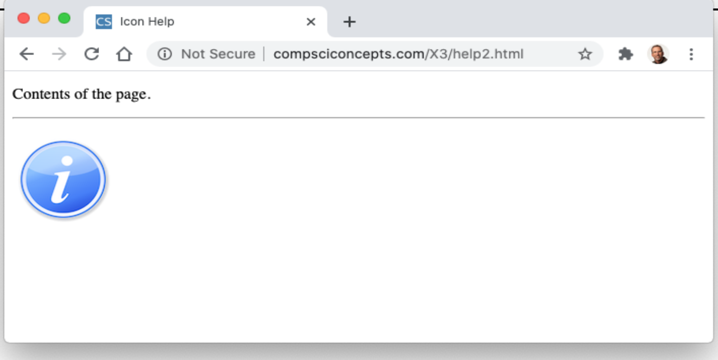


```
<!doctype html>
<!-- help2.html                                         Dave Reed -->
<!-- Web page that displays a help message on an image mouseover. -->
<!-- ============================================================ -->

<html>
 <head>
   <title>Icon Help</title>
 </head>

 <body>
   <p>
     Contents of the page.
   </p>
   <hr>
   <p>
     <img src="http://compsciconcepts.com/Images/information.png"
          alt="Information icon" style="vertical-align:middle"
          onmouseover="outputSpan.innerHTML=
                       'If you have any trouble with this site, ' +
                       'contact <i>admin@foo.bar</i>.';"
          onmouseout="outputSpan.innerHTML= ' ';" >
     <span id="outputSpan"> </span>
   </p>
 </body>
</html>
```

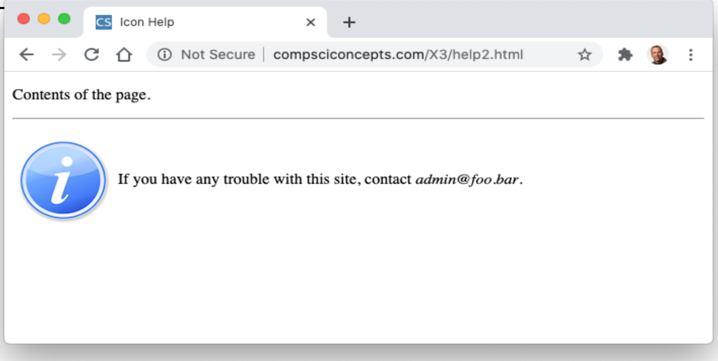
initially, `outputSpan` is blank

when the mouse moves over the image, a text message is assigned to `outputSpan.innerHTML` (showing the message)

when the mouse moves off the image, an empty string is assigned back to `outputSpan.innerHTML` (erasing the message)



Contents of the page.



Contents of the page.

If you have any trouble with this site, contact *admin@foo.bar*.

Common Errors

two types of errors are common when displaying complex messages

BROKEN STRING: can't start a string on one line and continue on the next

```
alert('This example is illegal because the
          string is broken across lines');

alert('This example is OK because the message ' +
          'is broken into two distinct strings');
```

DISCONNECTED STRING: if message is broken into pieces, must have + between the pieces to connect them

```
alert('This example is illegal because '
          'there is not a plus connecting the pieces');

alert('This example is OK because the ' +
          'pieces are properly connected');
```

error messages in the JavaScript Console make identifying these types of mistakes much easier

Dynamic Style

it is also possible to change the style attribute of an element

- must specify the property to be changed: style.PROPERTY
- if the property contains a hyphen, instead use capitalization
  e.g., background-color → backgroundColor

```
<p id="colorP"
   onmouseover="colorP.style.color='red';"
   onmouseout="colorP.style.color='black';">
    This text will turn red when the mouse moves over it.
</p>
```

```
<p>This is a really
   <span id="colorSpan
         onmouseover="colorSpan.style.backgroundColor='yellow';"
         onmouseout="colorSpan.style.backgroundColor='white';">
     important</span> point to note.
</p>
```

Machine Learning

Machine Learning (ML) is driving many powerful applications

- ML utilizes algorithms to process (potentially large) data sets to improve problem solving
- historically, has been seen as a subdiscipline of Artificial Intelligence

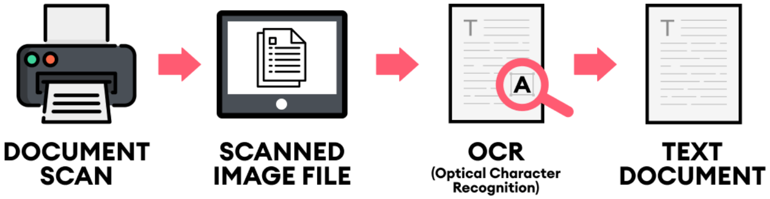
- the new discipline Data Science utilizes ML to process (potentially large) data sets to discover patterns
- this process is known as Data Mining

Machine Learning relies heavily on *supervised learning*

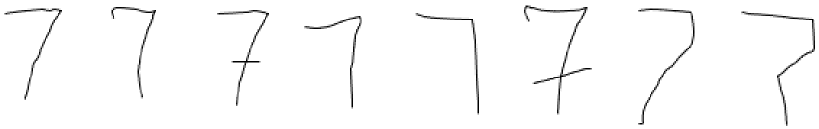
- algorithms are trained by providing sample inputs and classifications
- the algorithms learn to recognize patterns and classify new inputs

Example: OCR

Optical Character Recognition is used by devices (e.g., phones, tablets, scanners) to extract text from images



- when scanning printed text, each font has slight variations
- handwriting can have extreme variations



- programming every variation would be impossible
- instead, provide numerous examples and have the program learn to identify
  - essentially, the program deduces which features are essential to each character
  - uses those features to classify new examples

Example: facial recognition

similarly, facial recognition software identifies facial features (e.g., distance between eyes, ear size)

- can then match those features against new images to identify people
- since many features persist regardless of lighting or perspective, can be effective under varying conditions

- many smartphones utilize facial recognition for security
- has been used by law enforcement, but false matches and biases have caused many to rethink

Example: self-driving cars

autonomous vehicles utilize cameras, radar, lidar (light detection) and even GPS sense their surroundings

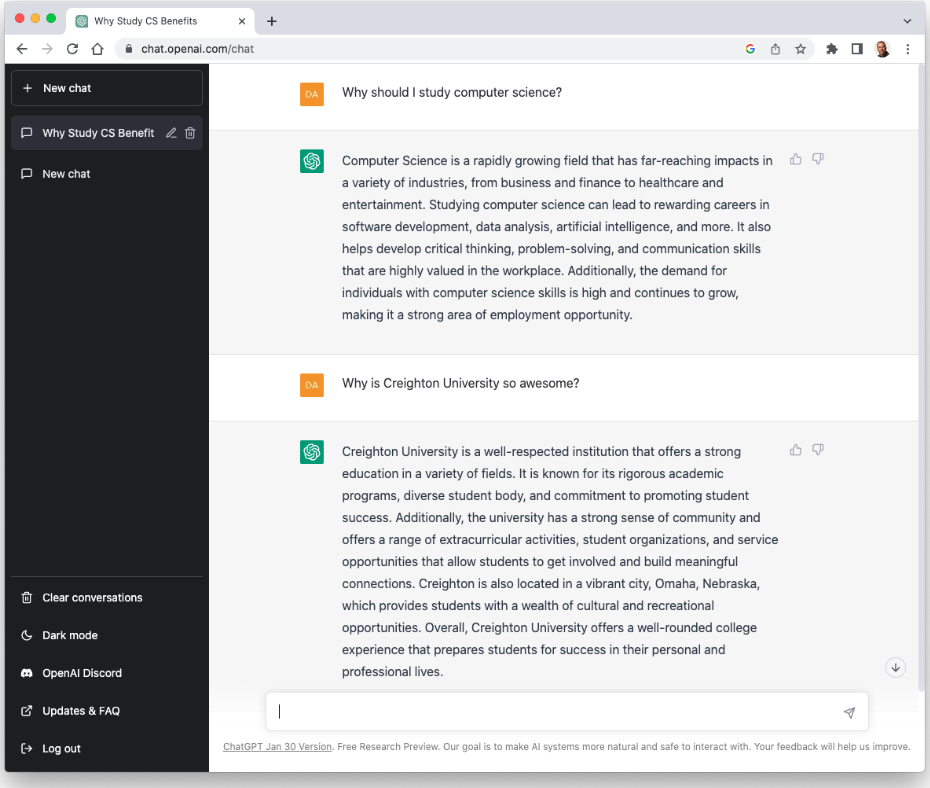
- they are programmed to react under specific circumstance, but learn to apply rules to new situations
- they can improve over time as new circumstances are experienced

- July-Oct 2022, 605 reported crashes by *advanced driver assistance* vehicles

Example: chatbots

ChatGPT (Chat Generative Pre-trained Transformer) was released in Nov 2022 by OpenAI

- was trained by providing recorded conversations and also utilized human trainers to provide feedback
- in addition to carry on on a conversation, can write code, music, poems, …



DA Why should I study computer science?

Computer Science is a rapidly growing field that has far-reaching impacts in a variety of industries, from business and finance to healthcare and entertainment. Studying computer science can lead to rewarding careers in software development, data analysis, artificial intelligence, and more. It also helps develop critical thinking, problem-solving, and communication skills that are highly valued in the workplace. Additionally, the demand for individuals with computer science skills is high and continues to grow, making it a strong area of employment opportunity.

DA Why is Creighton University so awesome?

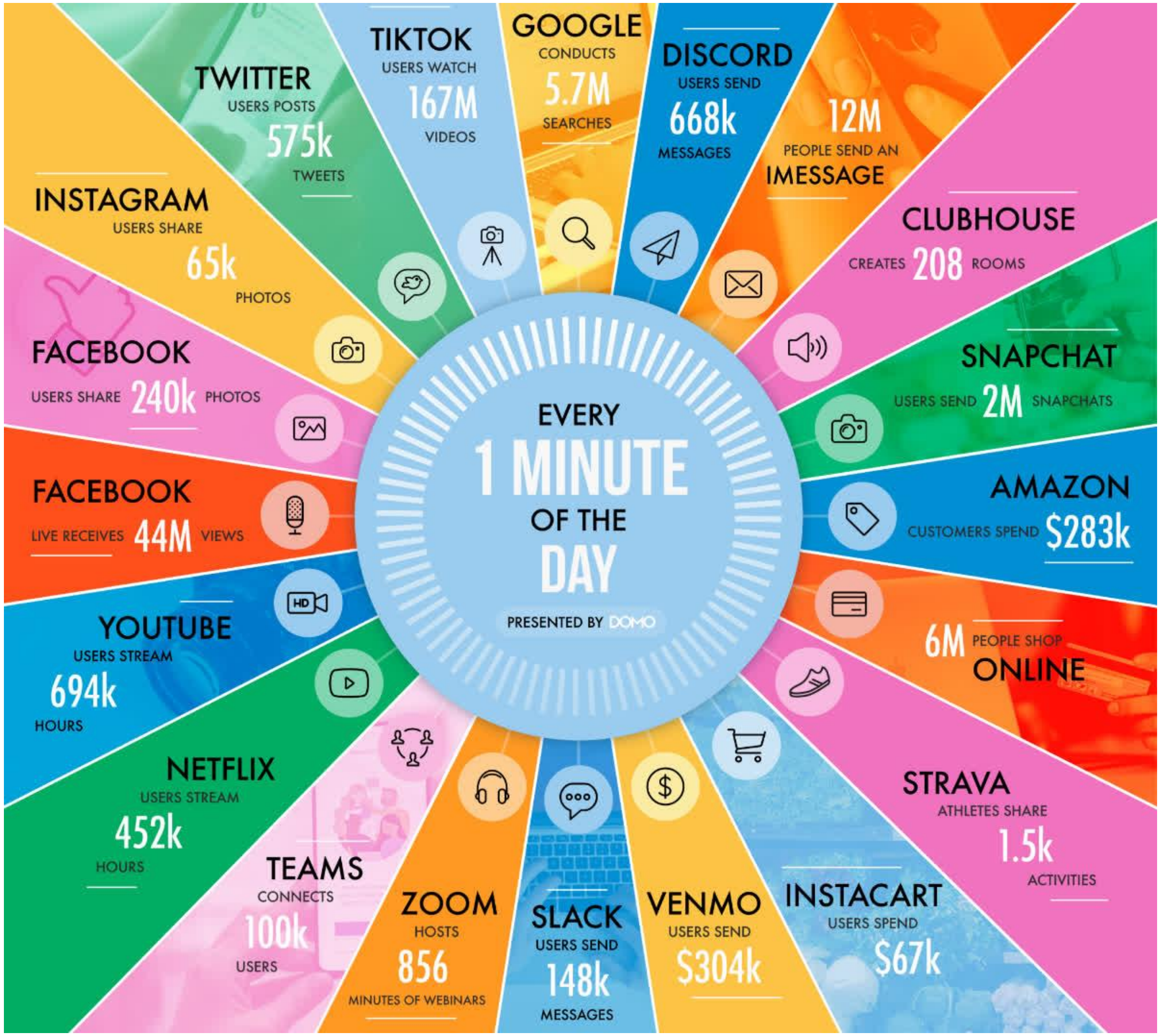
Creighton University is a well-respected institution that offers a strong education in a variety of fields. It is known for its rigorous academic programs, diverse student body, and commitment to promoting student success. Additionally, the university has a strong sense of community and offers a range of extracurricular activities, student organizations, and service opportunities that allow students to get involved and build meaningful connections. Creighton is also located in a vibrant city, Omaha, Nebraska, which provides students with a wealth of cultural and recreational opportunities. Overall, Creighton University offers a well-rounded college experience that prepares students for success in their personal and professional lives.

ChatGPT Jan 30 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

how will tools like this change

- education
- writing
- software development
- tech support
- therapy

Example: data mining



it was estimated that 1.1 trillion MB of data was generated every day in 2021

Data Science is a discipline that focuses on extracting patterns and trends from data

- relies on Machine Learning to process the massive amounts of data and learn what features are significant

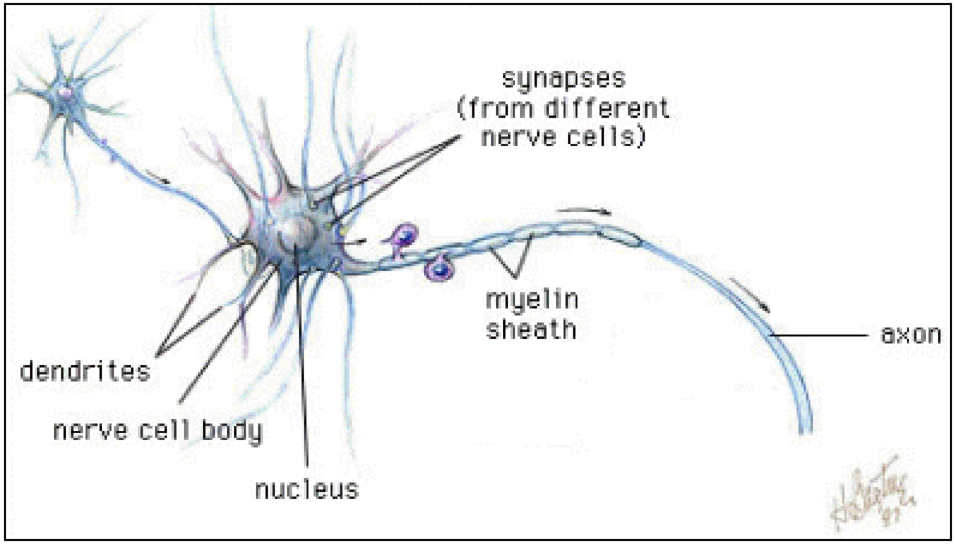
Neural Networks

at the core of most Machine Learning algorithms are *neural networks*

- a neural network mimics the structure of the human brain
- utilizes supervised learning to develop proficiency in recognizing patterns

neural nets predate modern computers

- first invented by McCulloch and Pitts in 1943



general brain architecture:

- many (relatively) slow neurons, interconnected
- dendrites serve as input devices (receive electrical impulses from other neurons)
- cell body "sums" inputs from the dendrites (possibly inhibiting or exciting)
- if sum exceeds some threshold, the neuron fires an output impulse along axon

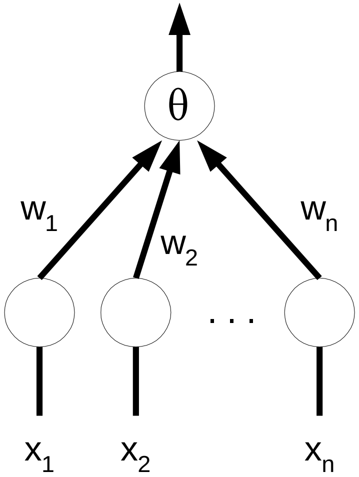
Artificial Neurons

neural networks are based on the brain metaphor

- large number of simple, neuron-like processing elements
- large number of weighted connections between neurons
  note: the weights encode information, not symbols!
- parallel, distributed control
- emphasis on learning

McCulloch & Pitts (1943) described an artificial neuron

- inputs are binary: 0 (no input signal) or 1 (input signal)
- each input has a weight associated with it
- the activation function multiplies each input value by its weight
- if the sum of the weighted inputs >= θ,
  then the neuron fires (outputs 1), else doesn't fire (outputs 0)



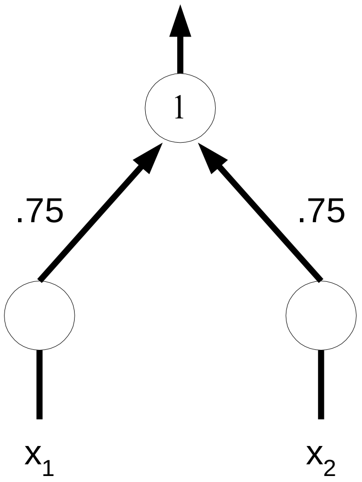
if $\Sigma w_i x_i >= \theta$, output = 1

if $\Sigma w_i x_i < \theta$, output = 0

Computation via Neurons

can view an (artificial or not) neuron as a computational element

- *accepts* or *classifies* an input if the output fires



INPUT: $x_1 = 1, x_2 = 1$

$.75*1 + .75*1 = 1.5 >= 1$ ➔ OUTPUT: 1

INPUT: $x_1 = 1, x_2 = 0$

$.75*1 + .75*0 = .75 < 1$ ➔ OUTPUT: 0

INPUT: $x_1 = 0, x_2 = 1$

$.75*0 + .75*1 = .75 < 1$ ➔ OUTPUT: 0

INPUT: $x_1 = 0, x_2 = 0$

$.75*0 + .75*0 = 0 < 1$ ➔ OUTPUT: 0

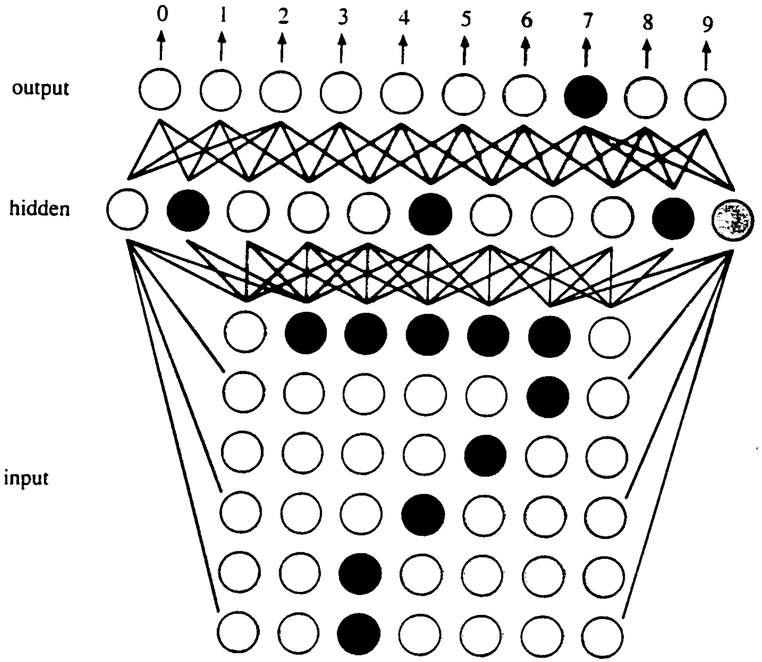
this neuron *computes* the AND function

Learning Algorithm

Rosenblatt (1958) devised a learning algorithm for artificial neurons

- start with a training set (example inputs & corresponding desired outputs)
- train the network to recognize the examples in the training set (by adjusting the weights on the connections)
- once trained, the network can be applied to new examples

this basic algorithm has been expanded to handle complex applications



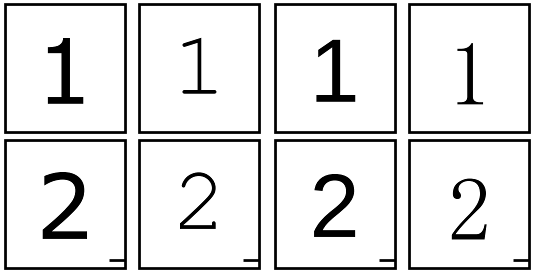
e.g., OCR

- inputs are pixels in a scanned image
- a "hidden" layer identifies relevant features (e.g., horizontal line near top, diagonal in middle)
- the combination of features discovered by the hidden layer identifies the character

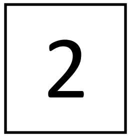
Generalization problem

suppose a network is trained to recognize digits:

- training set for 1:
- training set for 2:



when the network is asked to identify



it comes back with 1. WHY?

there is always a danger that the network will focus on specific (maybe unintentional) features as opposed to general patterns

to avoid networks that are too specific, *cross-validation* can be used

1. split training set into training & validation data
2. after each generation, test the net on the validation data
3. continue until performance on the validation data diminishes

Neural Net Applications

Aerospace: Aircraft component fault detectors and simulations, aircraft control systems

Automotive: Improved guidance systems, virtual sensors, warranty activity analyzers

Electronics: Chip failure analysis, circuit chip layouts, machine vision, non-linear modeling, process control

Manufacturing: Machine diagnosis, product design and analysis, visual quality inspection systems

Robotics: Forklift robots, manipulator controllers, trajectory control, and vision systems

Telecommunications: Network monitoring, speech recognition, customer payment processing systems

Banking: Credit card attrition, credit and loan application evaluation, fraud and risk evaluation

Business Analytics: Customer behavior modeling, fraud propensity, market research

Financial: Corporate financial analysis, currency price prediction, loan advising, portfolio trading

Securities: Automatic bond rating, market analysis, and stock trading advisory systems

source: https://www.smartsheet.com/neural-network-applications

more than 9,000 companies (including credit cards) use FICO Falcon

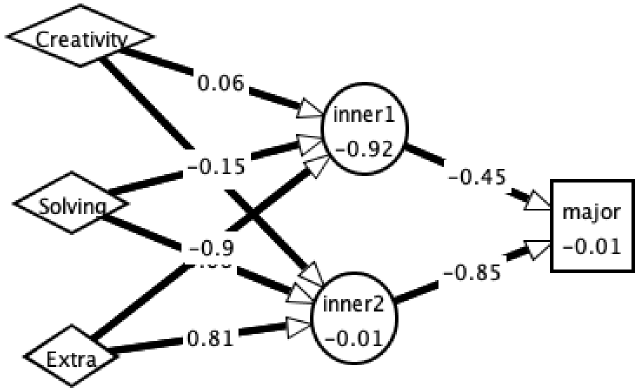
- uses neural nets to model customer behavior, identify fraud
- claims improvement in credit card fraud detection of 30-70%

NN example

suppose we wanted to provide guidance on major selection

- hypothesis: certain personality traits or life skills *suggest* certain majors
- conduct a survey of students, asking them to self-assess their traits/skills on a scale of 0 to 1.0, along with their major
- build a network of artificial neurons, with three inputs (corresponding to survey skills) and a single output (corresponding to major)

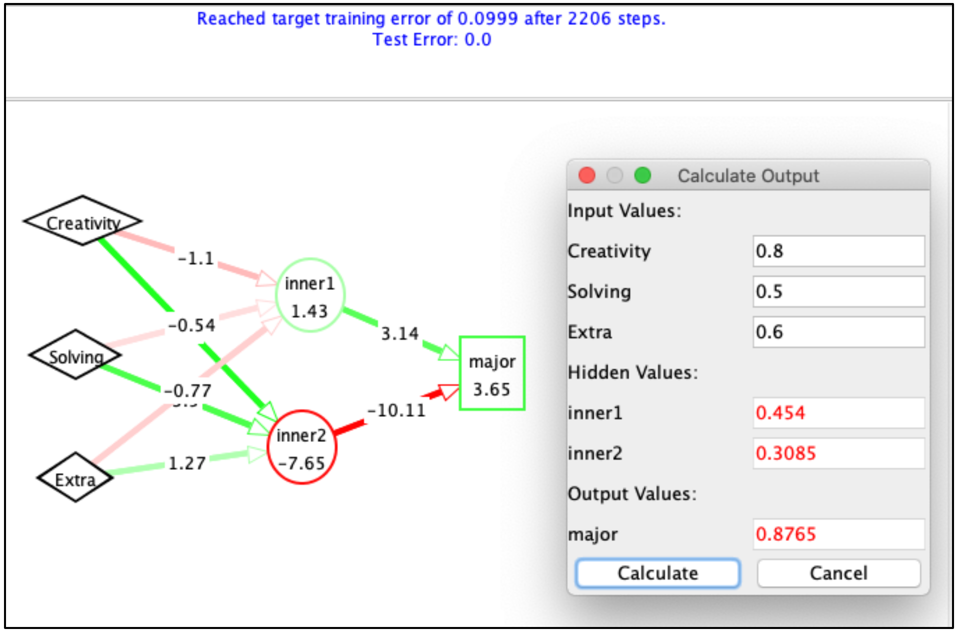
	How creative are you?	**Good at problem solving?**	**How extraverted are you?**	**Major**
stu1	0.85	0.75	0.9	GDE (0)
stu2	0.9	0.7	1.0	GDE (0)
stu3	0.8	0.9	0.6	GDE (0)
stu4	0.2	0.9	0.2	CSC (1)
stu5	0.6	0.8	0.4	CSC (1)
stu6	0.8	0.8	0.8	CSC (1)



NN example

- feed those inputs and outputs to the network and train it to recognize
- once trained, it can be applied to new patterns to classify them (based on best fit)

	How creative are you?	Good at problem solving?	How extraverted are you?	Major
stu1	0.85	0.75	0.9	GDE (0)
stu2	0.9	0.7	1.0	GDE (0)
stu3	0.8	0.9	0.6	GDE (0)
stu4	0.2	0.9	0.2	CSC (1)
stu5	0.6	0.8	0.4	CSC (1)
stu6	0.8	0.8	0.8	CSC (1)



Reached target training error of 0.0999 after 2206 steps.
Test Error: 0.0

Input Values:

Creativity	0.8
Solving	0.5
Extra	0.6

Hidden Values:

inner1	0.454
inner2	0.3085

Output Values:

major	0.8765

note: this is a very small, non-scientific example

- data scientists typically deal with massive amounts of data (e.g., hundreds or thousands of survey responses)
- utilize analytical methods to ensure statistical significance

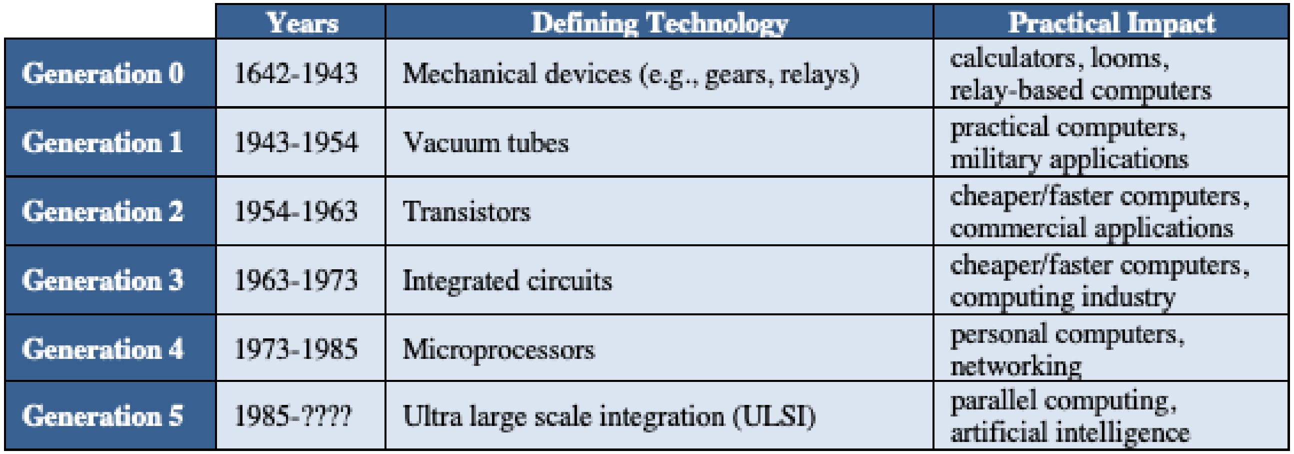
History of computing

calculating devices have been around for millennia (e.g., abacus ~3,000 B.C.)

modern "computing technology" traces its roots to the 16-17th centuries

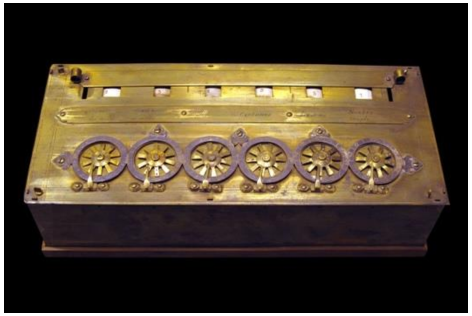
- as part of the "Scientific Revolution", people like Kepler, Galileo, and Newton viewed the natural world as mechanistic and understandable
- this led to technological advances & innovation

from simple mechanical calculating devices to powerful modern computers, computing technology has evolved through technological breakthroughs



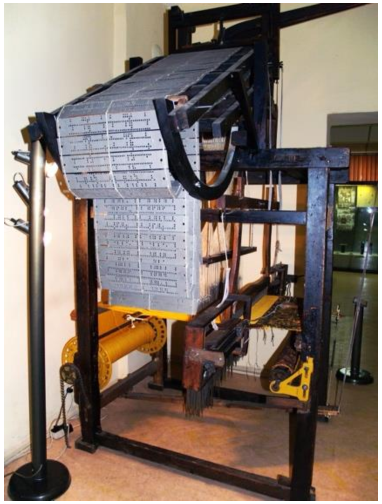
	Years	Defining Technology	Practical Impact
Generation 0	1642-1943	Mechanical devices (e.g., gears, relays)	calculators, looms, relay-based computers
Generation 1	1943-1954	Vacuum tubes	practical computers, military applications
Generation 2	1954-1963	Transistors	cheaper/faster computers, commercial applications
Generation 3	1963-1973	Integrated circuits	cheaper/faster computers, computing industry
Generation 4	1973-1985	Microprocessors	personal computers, networking
Generation 5	1985-????	Ultra large scale integration (ULSI)	parallel computing, artificial intelligence

Generation 0: Mechanical Computers



1642 – Pascal built a mechanical calculating machine

- used mechanical gears, a hand-crank, dials and knobs
- other similar machines followed



1805 – the first programmable device was Jacquard's loom

- the loom wove tapestries with elaborate, programmable patterns
- a pattern was represented by metal punch-cards, fed into the loom
- using the loom, it became possible to mass-produce tapestries, and even reprogram it to produce different patterns simply by changing the cards

mid 1800's – Babbage designed his "analytical engine"

- its design expanded upon mechanical calculators, but was programmable via punch-cards (similar to Jacquard's loom)
- Babbage's vision described the general layout of modern computers
- Ada Lovelace developed instructions for the never-quite-finished Analytical Engine – is considered the world's first programmer

Generation 0 (cont.)

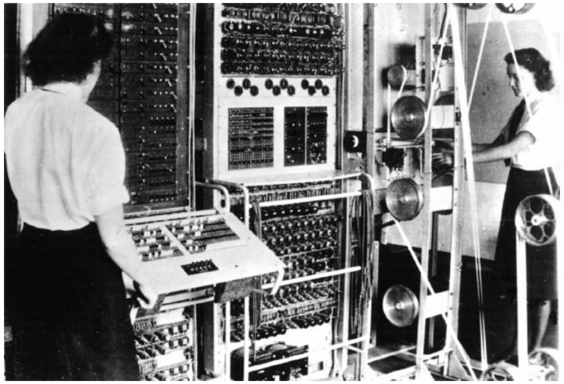
1930's – several engineers independently built "computers" using electromagnetic relays

- an electromagnetic relay is physical switch, which can be opened/closed via electrical current
- relays were used extensively in early telephone exchanges
- Zuse (Nazi Germany) – his machines were destroyed in WWII
- Atanasoff (Iowa State) – built a partially-working machine with his grad student
- Stibitz (Bell Labs) – built the MARK I computer that followed the designs of Babbage
  - limited capabilities by modern standards: could store only 72 numbers, required 1/10 sec to add, 6 sec to multiply
  - still, 100 times faster than previous technology

Generation 1: Vacuum Tubes

mid 1940's – vacuum tubes replaced relays

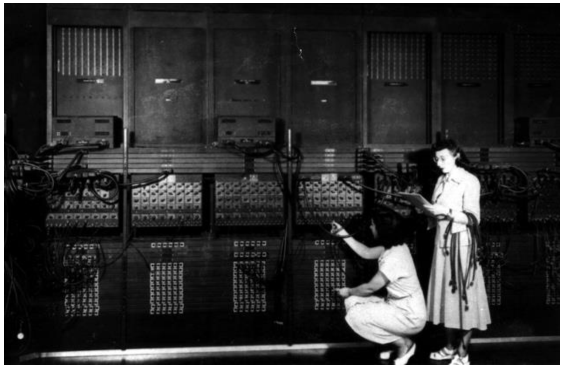
- a vacuum tube is a light bulb containing a partial vacuum to speed electron flow
- vacuum tubes could control the flow of electricity faster than relays since they had no moving parts
- invented by Lee de Forest in 1906



1940's – hybrid computers using vacuum tubes and relays were built

COLOSSUS (1943)

- first "electronic computer", built by the British govt. (based on designs by Alan Turing)
- used to decode Nazi communications during the war
- the computer was top-secret, so did not influence other researchers



ENIAC (1946)

- first publicly-acknowledged "electronic computer", built by Eckert & Mauchly (UPenn)
- 18,000 vacuum tubes and 1,500 relays
- weighed 30 tons, consumed 140 kwatts
- "programmed" by women CS pioneers

Generation 1 (cont.)

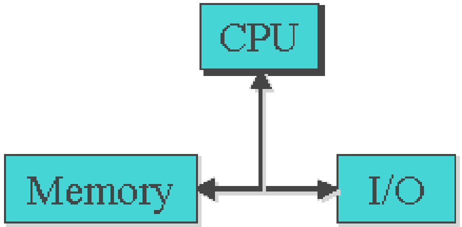
COLOSSUS and ENIAC were not general purpose computers

- could enter input using dials & knobs, paper tape
- but to perform a different computation, needed to reconfigure

von Neumann popularized the idea of a "stored program" computer

- Memory stores both data and programs
- Central Processing Unit (CPU) executes by loading program instructions from memory and executing them in sequence
- Input/Output devices allow for interaction with the user

virtually all modern machines follow this
von Neumann Architecture
(note: same basic design as Babbage)



programming was still difficult and tedious

- each machine had its own machine language, 0's & 1's corresponding to the settings of physical components
- in 1950's, assembly languages replaced 0's & 1's with mnemonic names
  e.g., `ADD` instead of `00101110`

Generation 2: Transistors

mid 1950's – transistors began to replace tubes

- a transistor is a piece of silicon whose conductivity can be turned on and off using an electric current
- they performed the same switching function of vacuum tubes, but were smaller, faster, more reliable, and cheaper to mass produce
- invented by Bardeen, Brattain, & Shockley in 1948 (earning them the 1956 Nobel Prize in physics)

some historians claim the transistor was the most important invention of the 20th century

as the cost of computers dropped, high-level languages were designed to make programming more natural (& efficient)

- FORTRAN (1957, Backus at IBM)

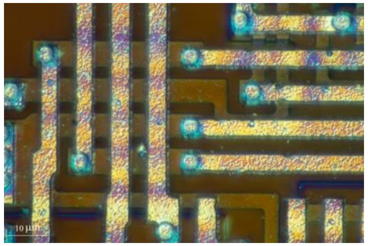
```
PROGRAM add
READ *, a,b
s = a + b
PRINT *, ' The sum is ', s
STOP
END
```

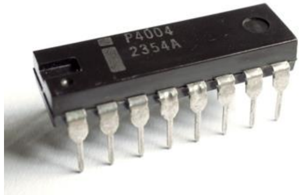
- LISP (1959, McCarthy at MIT)
- COBOL (1960, Hopper at DOD)

Generation 3: Integrated Circuits

mid 1960's - integrated circuits (IC) were produced

- Noyce and Kilby independently developed techniques for packaging transistors and circuitry on a silicon chip (Kilby won the 2000 Nobel Prize in physics)
- was made possible by miniaturization & improved manufacturing
- allowed for mass-producing useful circuitry





1960's saw the rise of computing for business

recall: an *operating system* is a collection of programs that manage peripheral devices and other resources

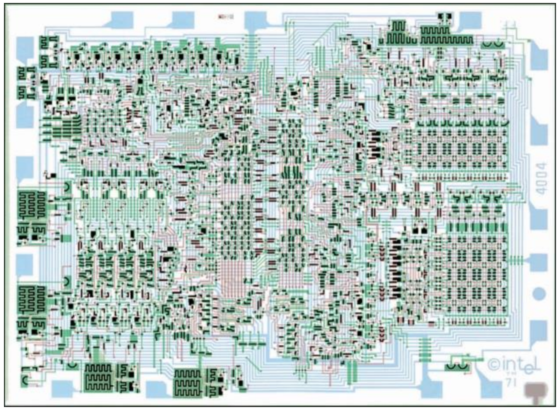
- in the 60's, operating systems enabled time-sharing, where users share a computer by swapping jobs in and out
- specialized programming languages were developed, e.g., Pascal (1971, Wirth), C (1972, Ritchie)

U.S. space program was a driving force behind innovation

- computers began to replace humans for complex calculations (e.g., Katherine Johnson)
- Margaret Hamilton at MIT led team that developed Apollo Guidance system

Generation 4: Microprocessors

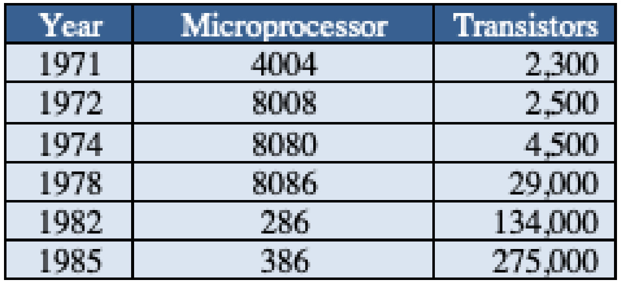
1971 – Intel marketed the first *microprocessor*, the 4004, a chip with all the circuitry for a calculator



- by the late 1970's, manufacturing advances allowed for the very large scale integration (VLSI) of hundreds of thousands of transistors w/ circuitry on a chip
- this "very large scale integration" resulted in mass-produced microprocessors and other useful IC's
- since computers could be constructed by simply connecting powerful IC's and peripheral devices, they were easier to make and more affordable

- Moore's Law (more of an observation, really)–

  the number of transistors that could fit on a chip doubled every 1-2 years



Year	Microprocessor	Transistors
1971	4004	2,300
1972	8008	2,500
1974	8080	4,500
1978	8086	29,000
1982	286	134,000
1985	386	275,000

Generation 4: Microprocessors

with microprocessors came personal computing

- 1975 - Bill Gates & Paul Allen founded Microsoft

  Gates wrote a BASIC interpreter for the first PC (Altair)

- 1977 - Steve Wozniak & Steve Jobs founded Apple

  went from Jobs' garage to $120 million in sales by 1980

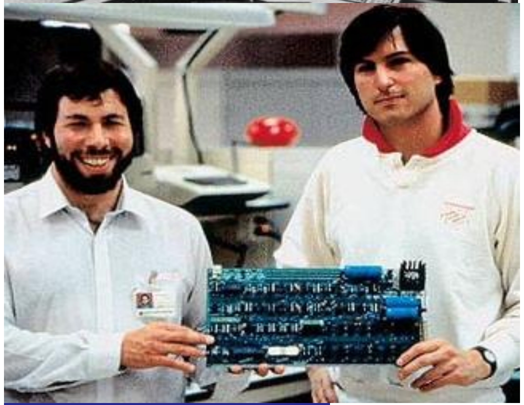
- 1980 - IBM introduced PC

  Microsoft licensed the DOS operating system to IBM

- 1984 - Apple countered with Macintosh

  introduced the modern GUI-based OS (which was mostly developed at Xerox)

- 1985 - Microsoft countered with Windows



in the 1980's

- demand grew for networking computers together

  1982: 235 computers connected to ARPANet

  1989: 300,000 computers connected to Internet

- object-oriented programming represented a new approach to program design which views a program as a collection of interacting software objects that model real-world entities

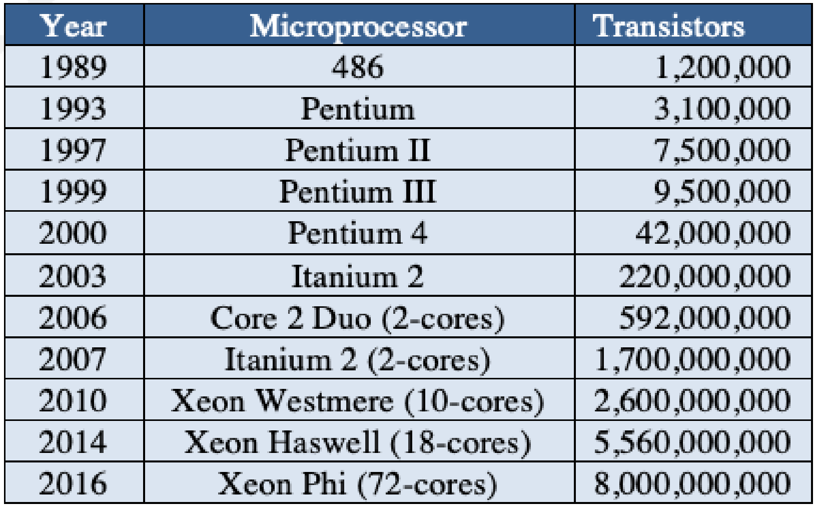
Generation 5: ULSI

the latest generation of computers is still hotly debated

- no new switching technologies, ultra large scale integration (ULSI) has changed how computers are used

1n 1989, the Intel 486 contained 1.2 million transistors

- manufacturing improvements are more difficult to achieve as components get smaller and smaller (Moore's Law in jeopardy?)



Year	Microprocessor	Transistors
1989	486	1,200,000
1993	Pentium	3,100,000
1997	Pentium II	7,500,000
1999	Pentium III	9,500,000
2000	Pentium 4	42,000,000
2003	Itanium 2	220,000,000
2006	Core 2 Duo (2-cores)	592,000,000
2007	Itanium 2 (2-cores)	1,700,000,000
2010	Xeon Westmere (10-cores)	2,600,000,000
2014	Xeon Haswell (18-cores)	5,560,000,000
2016	Xeon Phi (72-cores)	8,000,000,000

workarounds

- multi-core processors increase the chip size by adding duplicate circuitry so that it can execute operations simultaneously
- parallel processing computers have multiple independent processors that can share the load (e.g., a Web server)

Generation 5: ULSI (cont.)

Wi-fi and wireless broadband have made computing mobile and pervasive

- wi-fi utilizes radio waves over short distances to connect computers and devices
- typical speed & range: 100-200 Mbits/sec, 150-300 feet

1. user enters commands on computer/device
2. command is translated into radio signal, broadcast to wi-fi router
3. router carries out the command via Internet connection
4. response is translated into radio signal, broadcast back to computer/device

- a longer range alternative to wi-fi is cellular networking
- 4G (15-25 Mbits/sec) & now 5G (50 Mbits/sec – 1 Gbits/sec), nationwide coverage

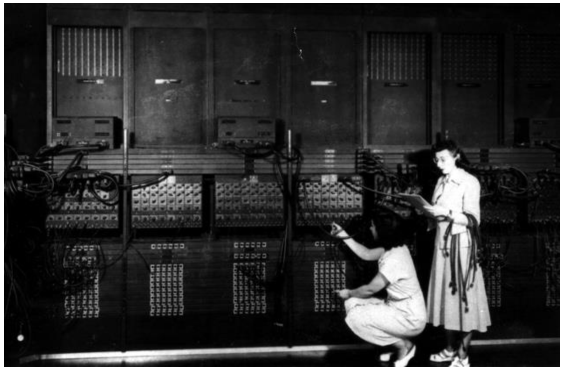
Artificial Intelligence (AI) applications dominate the news

- Apple's Siri (2011) and Amazon's Alexa (2014) can recognize voice commands and control smart home devices
- facial recognition software is used by law enforcement and businesses
- credit card companies model purchasing patterns to identify fraud
- retailers like Amazon use your history to predict future purchases
- Self-driving cars from Uber and Tesla use video processing and AI techniques to control vehicles on open roads

Speed matters

ENIAC (1946)

- could perform 385 operations per second



75 years later…

iPhone 13 (2021)

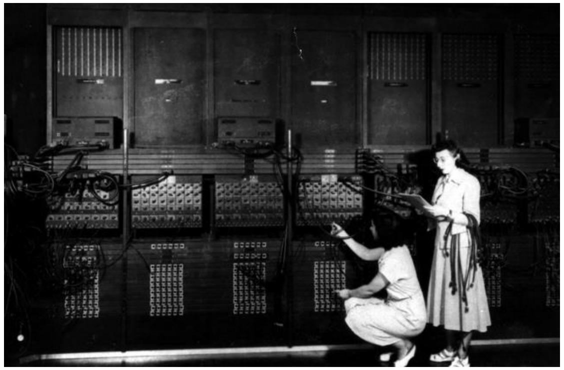
- can perform 15.8 trillion operations per second
- 41 billion times faster!

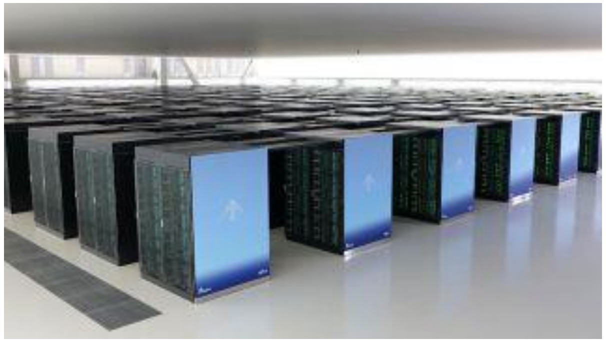
it would take the ENIAC 1,301 years to do what your iPhone can do in 1 second

Speed REALLY matters

ENIAC (1946)

- could perform 385 operations per second





Fugaku supercomputer (2021)

- can perform 442 quadrillion operations per second
- 1.1 quadrillion times faster!!!

it would take the ENIAC 36.4 million years to do what the Fugaku can do in 1 second

Computing entrepreneurs

Richest People in the World (Forbes, 3/5/21)

1. Jeff Bezos	$177.0 billion	Age: 57
2. Elon Musk	$151.0 billion	Age: 50
3. Bernard Arnault	$150.0 billion	Age: 72
4. Bill Gates	$124.0 billion	Age: 65
5. Mark Zuckerberg	$97.0 billion	Age: 36
6. Warren Buffet	$96.0 billion	Age: 90
7. Larry Ellison	$93.0 billion	Age: 76
8. Larry Page	$91.5 billion	Age: 47
9. Sergey Brin	$89.0 billion	Age: 47
10. Mukesh Ambani	$84.5 billion	Age: 63
…		
14. Steve Ballmer	$68.7 billion	Age: 64
15. Ma Huatang	$65.8 billion	Age: 49

Text Boxes

HTML event handlers enable the user to interact with the page

e.g., move the mouse over an image to change it

e.g., click on a button to display a text message in a page division

for greater control, the user must be able to enter information into the page

e.g., enter words to complete a fill-in-the-blank story

e.g., enter grades to calculate a course average

a *text box* is an HTML element that is embedded in the page

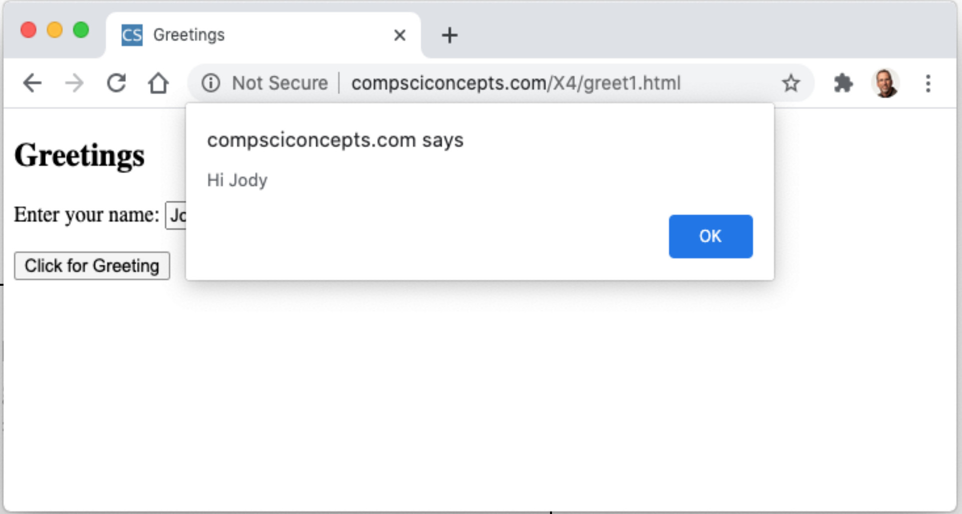
```
<input type="text" id="BOX_ID" size=NUM_CHARS value="INITIAL_CONTENTS">
```

- the user can enter text directly in the box
- a JavaScript statement can then access the contents of the text box by accessing its VALUE attribute

```
BOX_ID.value
```

Greetings Page (v.1)

note: userBox.value does not have quotes around it
what would happen if it did?



compsciconcepts.com says

Hi Jody

Greetings

Enter your name: Jc

Click for Greeting

```
<!doctype html>
<!-- greet1.html
<!-- Web page that displays a personalized
<!-- ====================================

<html>
 <head>
   <title> Greetings </title>
 </head>

 <body>
   <h2>Greetings</h2>
   <p>
     Enter your name: <input type="text" id="userBox" size=12 value="Jody">
   </p>
   <button onclick="alert('Hi ' + userBox.value);">
     Click for Greeting
   </button>
 </body>
</html>
```

"Jody" is the default value

when button is clicked, an alert window displays 'Hi Jody'

the user can enter his/her name in the text box

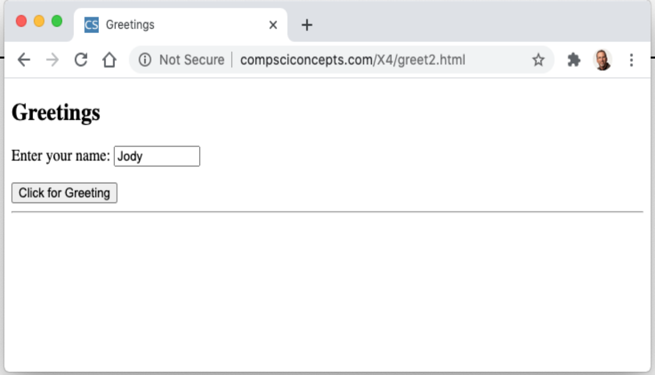
Greetings Page (v.2)

```html
<!doctype html>
<!-- greet2.html                           Dave Reed -->
<!-- Web page that displays a personalized greeting. -->
<!-- =============================================== -->

<html>
 <head>
   <title> Greetings </title>
 </head>

 <body>
   <h2>Greetings</h2>
   <p>
     Enter your name: <input type="text" id="userBox" size=12 value="Jody">
   </p>
   <button onclick="outputP.innerHTML = 'Hi ' + userBox.value;">
     Click for Greeting
   </button>
   <hr>
   <p id="outputP"> </p>
 </body>
</html>
```

alternatively, could display the greeting in a dynamic paragraph

initially, `outputP` is empty

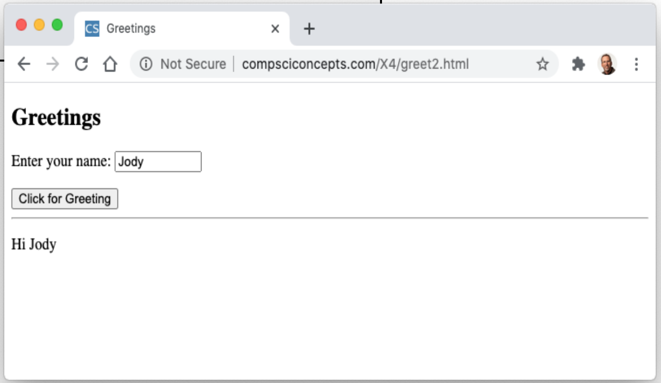
when button is clicked, the greeting is assigned to its `INNERHTML` attribute



Greetings

Enter your name: Jody

Click for Greeting



Greetings

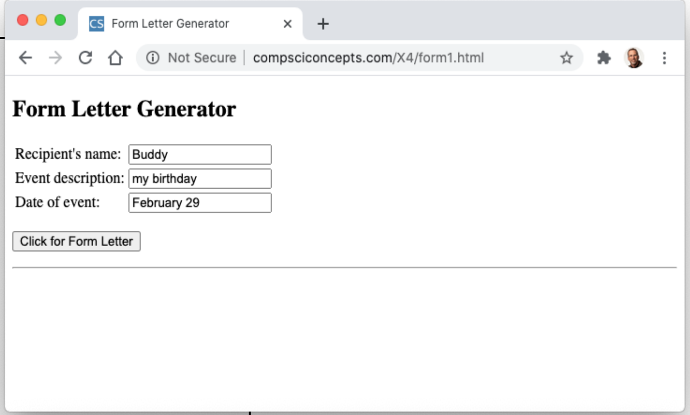
Enter your name: Jody

Click for Greeting

Hi Jody

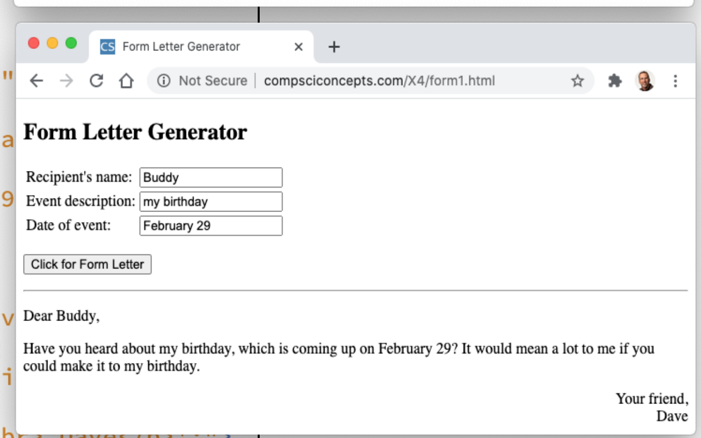
Form Letter Example

```
<!doctype html>
<!-- form1.html                                          Dave Reed -->
<!-- Web page that generates a form letter based on user inputs. -->
<!-- ========================================================== -->

<html>
 <head>
   <title> Form Letter Generator </title>
 </head>

 <body>
   <h2>Form Letter Generator</h2>
   <table>
     <tr><td>Recipient's name: </td>
         <td><input type="text" id="recipientBox" size=20 value="Buddy"
     <tr><td>Event description: </td>
         <td><input type="text" id="eventBox" size=20 value="my birthda
     <tr><td>Date of event: </td>
         <td><input type="text" id="dateBox" size=20 value="February 29
   </table>
   <p>
     <button onclick="outputDiv.innerHTML=
                         '<p>Dear ' + recipientBox.value + ',</p> <p>Hav
                         'about ' + eventBox.value + ', which is coming
                         dateBox.value + '?  It would mean a lot to me i
                         'make it to ' + eventBox.value +
                         '. <p style=\'text-align:right\'>Your friend,<br> Dave</p>';">
       Click for Form Letter
     </button>
   </p>
   <hr>
   <div id="outputDiv"> </div>
 </body>
</html>
```



Form Letter Generator

Recipient's name:	Buddy
Event description:	my birthday
Date of event:	February 29



Form Letter Generator

Recipient's name:	Buddy
Event description:	my birthday
Date of event:	February 29

Dear Buddy,

Have you heard about my birthday, which is coming up on February 29? It would mean a lot to me if you could make it to my birthday.

Your friend,
Dave

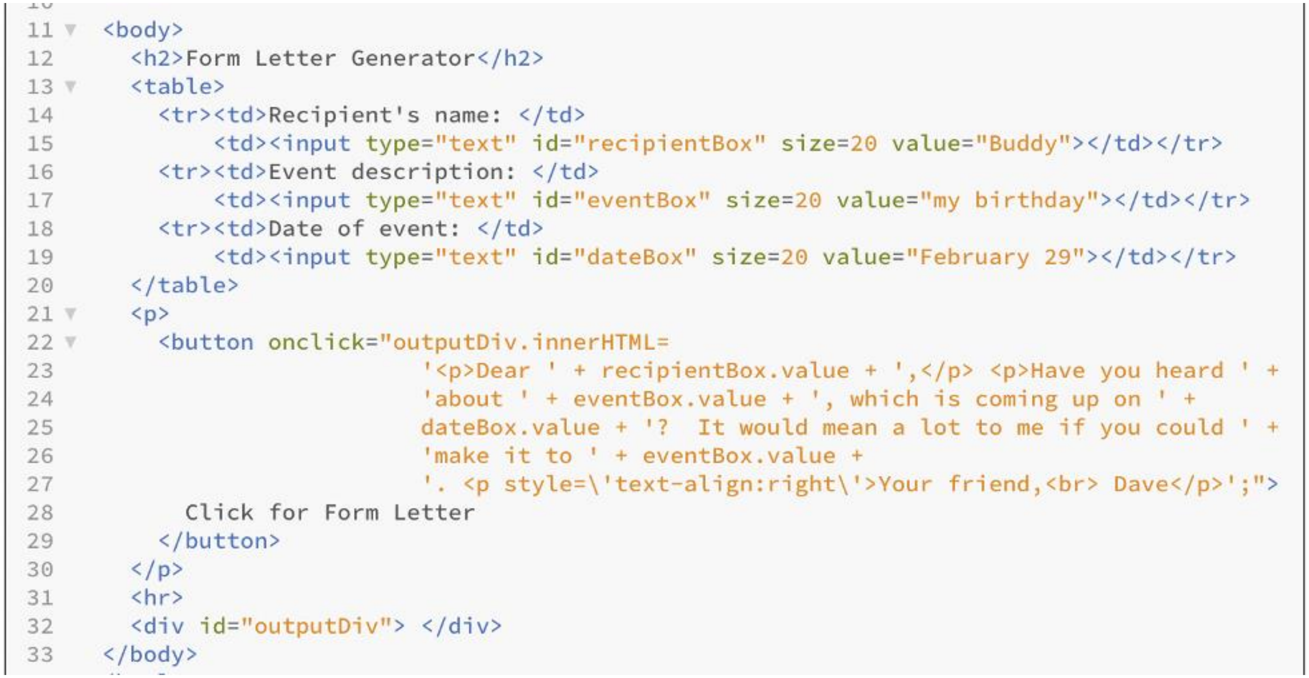
this page has 3 text boxes, uses contents to generate a custom form letter

note: each box must have a unique ID

Simplifying Pages

we should be able to glance at the body of a page and visualize its contents

- consider the body of the form generator page
- the onclick code has nothing to do with the look of the page but its size and complexity clutter the body



```
<body>
  <h2>Form Letter Generator</h2>
  <table>
    <tr><td>Recipient's name: </td>
        <td><input type="text" id="recipientBox" size=20 value="Buddy"></td></tr>
    <tr><td>Event description: </td>
        <td><input type="text" id="eventBox" size=20 value="my birthday"></td></tr>
    <tr><td>Date of event: </td>
        <td><input type="text" id="dateBox" size=20 value="February 29"></td></tr>
  </table>
  <p>
    <button onclick="outputDiv.innerHTML=
                       '<p>Dear ' + recipientBox.value + ',</p> <p>Have you heard ' +
                       'about ' + eventBox.value + ', which is coming up on ' +
                       dateBox.value + '?  It would mean a lot to me if you could ' +
                       'make it to ' + eventBox.value +
                       '. <p style=\'text-align:right\'>Your friend,<br> Dave</p>';">
      Click for Form Letter
    </button>
  </p>
  <hr>
  <div id="outputDiv"> </div>
</body>
```

User-defined Functions

can simplify the body by moving the JavaScript statements to the head

- define a function that encapsulates those statements; then call from `onclick`

mathematically speaking, a *function* is a mapping from inputs to an output

$\sqrt{9}$ → 3 |-2| → 2 max(5, 2) → 5

to a developer, a function is a *unit of computational abstraction*

- a function encapsulates statement(s) under a name; to execute, only need to know the name (i.e., call the function)

function definition:

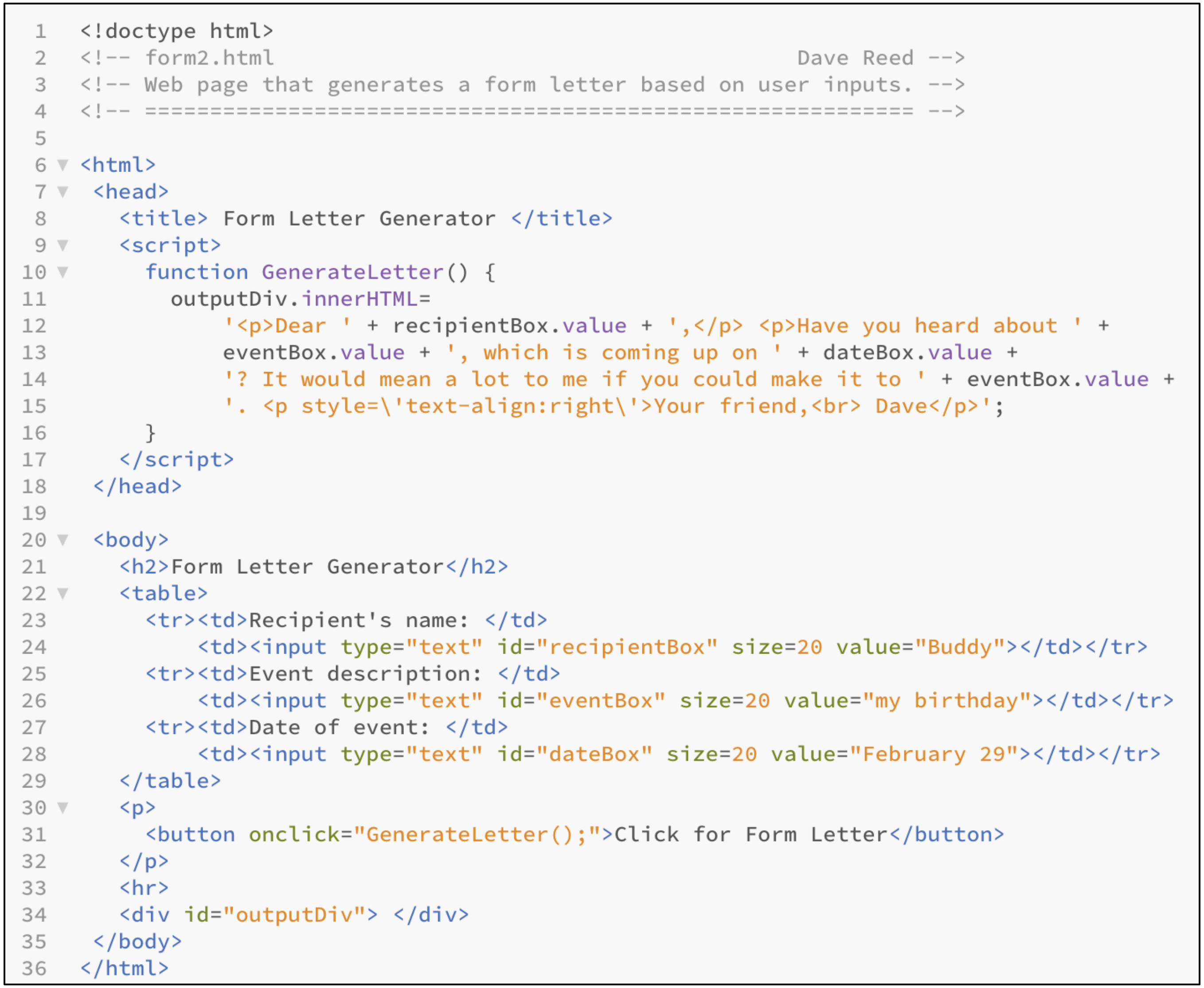
```
function FUNCTION_NAME() {
    STATEMENTS_TO_BE_EXECUTED
}
```

function call:

```
FUNCTION_NAME();
```

- function definitions are place in the head of a page, in a `script` element

Function Example



```
<!doctype html>
<!-- form2.html                                        Dave Reed -->
<!-- Web page that generates a form letter based on user inputs. -->
<!-- =========================================================== -->

<html>
 <head>
   <title> Form Letter Generator </title>
   <script>
     function GenerateLetter() {
       outputDiv.innerHTML=
           '<p>Dear ' + recipientBox.value + ',</p> <p>Have you heard about ' +
           eventBox.value + ', which is coming up on ' + dateBox.value +
           '? It would mean a lot to me if you could make it to ' + eventBox.value +
           '. <p style=\'text-align:right\'>Your friend,<br> Dave</p>';
     }
   </script>
 </head>

 <body>
   <h2>Form Letter Generator</h2>
   <table>
     <tr><td>Recipient's name: </td>
         <td><input type="text" id="recipientBox" size=20 value="Buddy"></td></tr>
     <tr><td>Event description: </td>
         <td><input type="text" id="eventBox" size=20 value="my birthday"></td></tr>
     <tr><td>Date of event: </td>
         <td><input type="text" id="dateBox" size=20 value="February 29"></td></tr>
   </table>
   <p>
     <button onclick="GenerateLetter();">Click for Form Letter</button>
   </p>
   <hr>
   <div id="outputDiv"> </div>
 </body>
</html>
```

this page behaves the same as `form1.html`

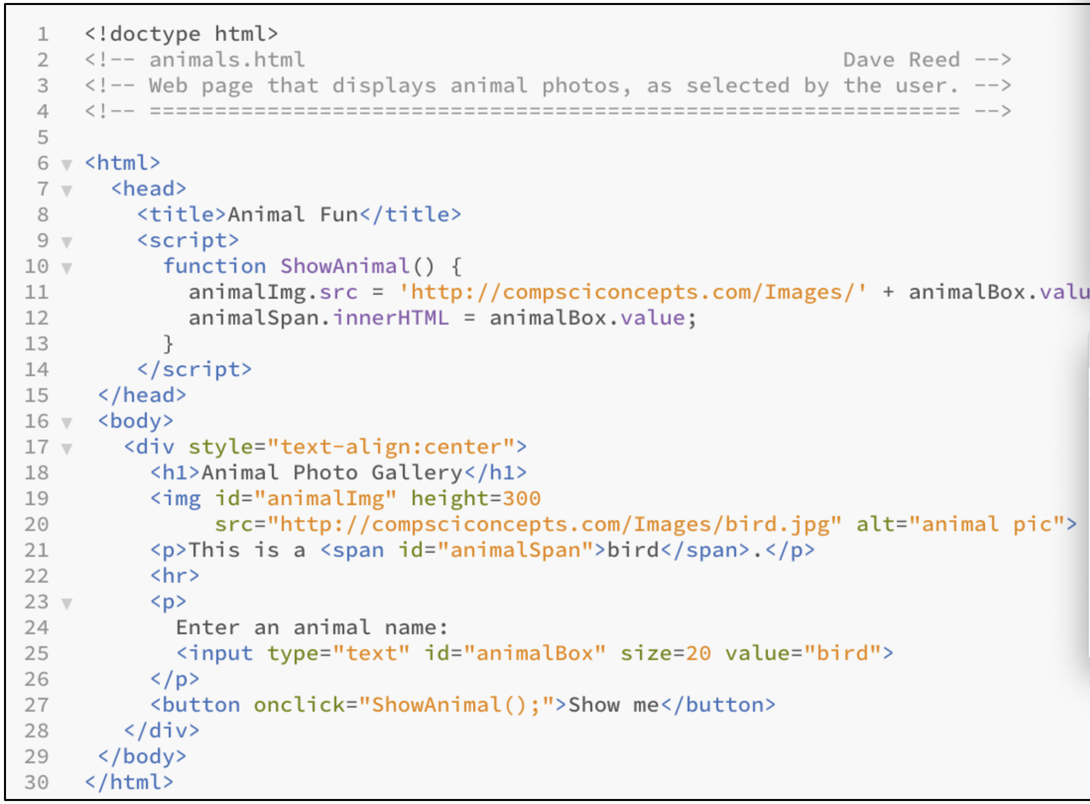
the action of the button is encapsulated in the `GenerateLetter` function in the HEAD (in SCRIPT tags)

the BODY is simplified, since the ONCLICK just contains a call to the function

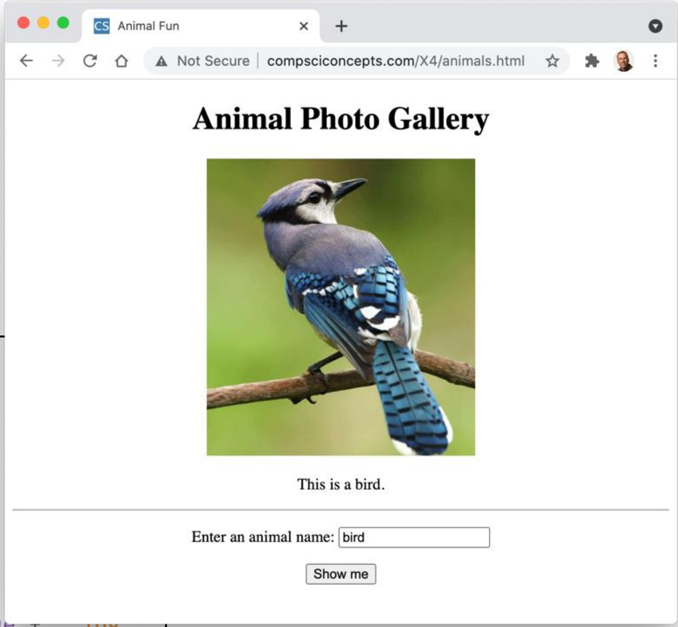
Another Example

Animal Gallery: contains

- image of an animal
- a text box where the user can enter an animal name
- a button for changing the image



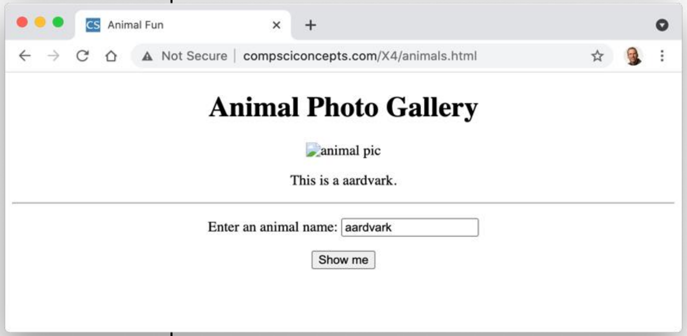
```
<!doctype html>
<!-- animals.html                                        Dave Reed -->
<!-- Web page that displays animal photos, as selected by the user. -->
<!-- ============================================================= -->

<html>
  <head>
    <title>Animal Fun</title>
    <script>
      function ShowAnimal() {
        animalImg.src = 'http://compsciconcepts.com/Images/' + animalBox.value + '.jpg';
        animalSpan.innerHTML = animalBox.value;
      }
    </script>
  </head>
  <body>
    <div style="text-align:center">
      <h1>Animal Photo Gallery</h1>
      <img id="animalImg" height=300
           src="http://compsciconcepts.com/Images/bird.jpg" alt="animal pic">
      <p>This is a <span id="animalSpan">bird</span>.</p>
      <hr>
      <p>
        Enter an animal name:
        <input type="text" id="animalBox" size=20 value="bird">
      </p>
      <button onclick="ShowAnimal();">Show me</button>
    </div>
  </body>
</html>
```



Animal Photo Gallery

This is a bird.

Enter an animal name: bird



Animal Photo Gallery

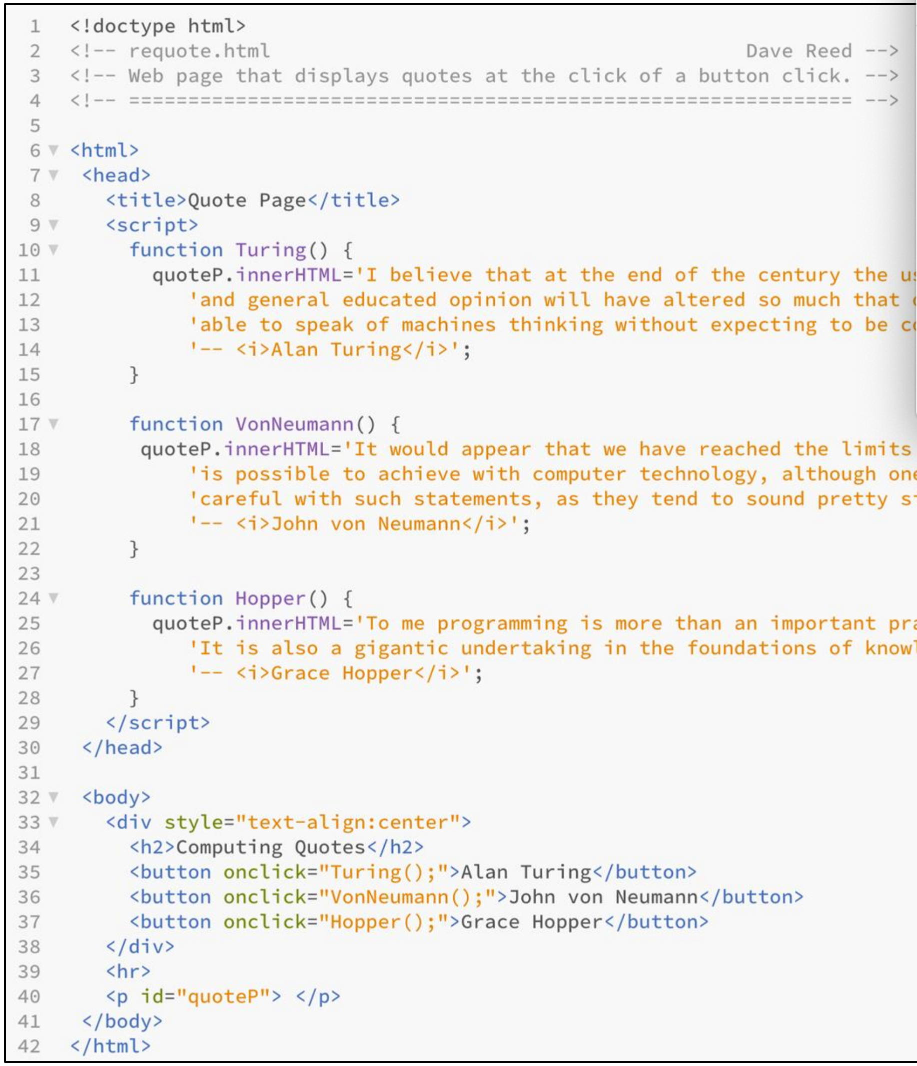
animal pic

This is a aardvark.

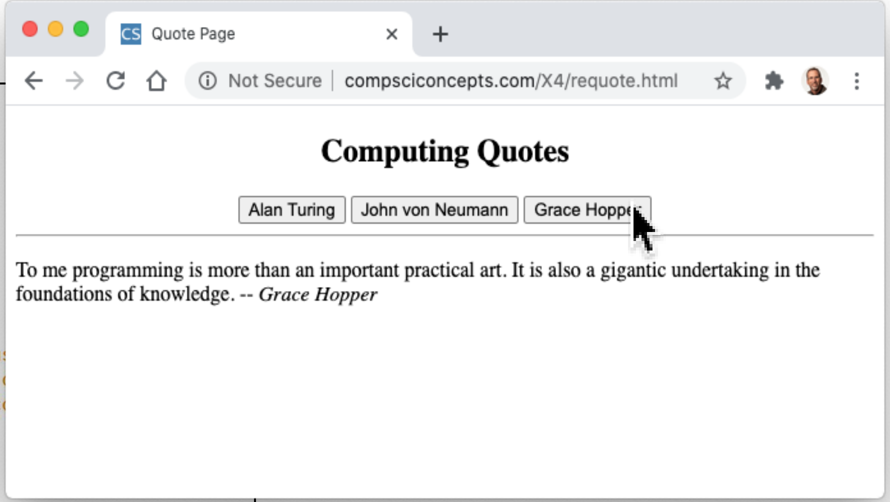
Enter an animal name: aardvark

note how simple the BODY is

Another Example



```
<!doctype html>
<!-- requote.html                                        Dave Reed -->
<!-- Web page that displays quotes at the click of a button click. -->
<!-- ============================================================= -->

<html>
 <head>
   <title>Quote Page</title>
   <script>
     function Turing() {
       quoteP.innerHTML='I believe that at the end of the century the u
          'and general educated opinion will have altered so much that
          'able to speak of machines thinking without expecting to be c
          '-- <i>Alan Turing</i>';
     }

     function VonNeumann() {
      quoteP.innerHTML='It would appear that we have reached the limits of what it ' +
          'is possible to achieve with computer technology, although one should be ' +
          'careful with such statements, as they tend to sound pretty silly in 5 years. ' +
          '-- <i>John von Neumann</i>';
     }

     function Hopper() {
       quoteP.innerHTML='To me programming is more than an important practical art. ' +
          'It is also a gigantic undertaking in the foundations of knowledge. ' +
          '-- <i>Grace Hopper</i>';
     }
   </script>
 </head>

 <body>
   <div style="text-align:center">
     <h2>Computing Quotes</h2>
     <button onclick="Turing();">Alan Turing</button>
     <button onclick="VonNeumann();">John von Neumann</button>
     <button onclick="Hopper();">Grace Hopper</button>
   </div>
   <hr>
   <p id="quoteP"> </p>
 </body>
</html>
```



Computing Quotes

To me programming is more than an important practical art. It is also a gigantic undertaking in the foundations of knowledge. -- *Grace Hopper*

each button has its own corresponding function

when clicked, the function is called to display a quotation in the page

note how simple the BODY is

Design Guidelines

- be conservative in your use of color
- be consistent in your use of formatting
- avoid overriding the browser defaults unless it is necessary
- don't center paragraphs
- label interactive elements (buttons, text boxes) clearly
- document the source code in case anyone views it
  - include a comment block at top with your name, file name, and brief description
- use descriptive id names
  - convention: descriptor + elementType, e.g., `vacationImg`, `outputP`
- when defining a complex action, place in function definition in the head and call the function from the event handler

Errors and Debugging

in computer jargon, the term *bug* refers to an error in a program

- the process of systematically locating and fixing errors is *debugging*

three types of errors can occur

1. *syntax errors:* typographic errors
   - e.g., omitting a quote or misspelling a function name
   - since the browser catches these, they are usually "easy" to identify and fix

     CHECK THE JAVASCRIPT CONSOLE FOR ERROR MESSAGES!

2. *run-time errors:* occur when operations are applied to illegal values
   - e.g., attempting to multiply a string or divide by zero
   - also caught by the browser, which either produces an error message or else returns a special value (string multiplication produces `NaN`, division by zero produces `Infinity`)

     AGAIN, CHECK THE JAVASCRIPT CONSOLE!

1. *logic errors:* flaws in the design or implementation of a program
   - whenever your program produces the wrong result
   - since they are not caught by the browser (the program is legal, just not what you wanted), logic errors are hardest to identify

useful technique for identifying logic errors: *diagnostic alert statements*

- at various intervals in the code, display the values of key variables using alert
- you can then isolate at what point the program is going wrong

Early science

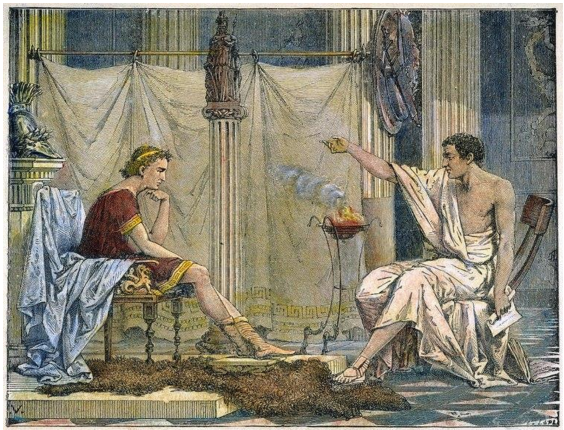
science: a system of knowledge covering general truths or the operation of general laws especially as obtained and tested through scientific method (Merriam-Webster dictionary)

science is important in our daily lives because:

- it advances our understanding of the world and our place in it
- scientific advances can lead to practical applications (e.g., technology, medicine, …)

modern science traces its roots back to the Greek natural philosophers

- Thales (6 c B.C.) was first to break from mythology
  - observed and devised theories about nature
- Plato (4 c B.C.) proposed a grand theory of cosmology
  - claimed heavenly bodies move uniformly in circles, because of their divine, geometric perfection
  - believed observation was confused and impure, truth was found through contemplation
- Aristotle (4 c B.C.) proposed a common-sense vision of the natural world that stood for 2,000 years
  - studied and wrote on a cosmology, physics, biology, anatomy, logic, …
  - placed greater emphasis on observation than Plato, but still not experimental



ARISTOTLE AND HIS PUPIL, ALEXANDER.

Greek natural philosophy is "pre-scientific", since it relied on contemplation/observation, but not experimentation

Roman times → Middle Ages

Roman civilization built upon the tradition of Greek natural philosophy

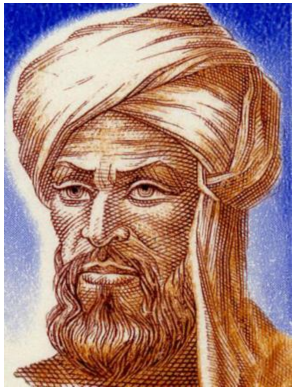
- the Romans are better known for engineering than theoretical science
- Pliny (1 c.) categorized plants, animals and minerals
- Galen (2nd century) studied human anatomy and physiology

the fall of Rome (in 476) led to a discontinuity in western civilization

- in western Europe, population dropped, literacy virtually disappeared, and Greek knowledge was lost
- in eastern Europe, Greek knowledge was suppressed by orthodox Christianity in the Byzantine Empire (which finally fell in 1453)

during Europe's "Dark Age," medieval Islam became the principal heir to Greek science

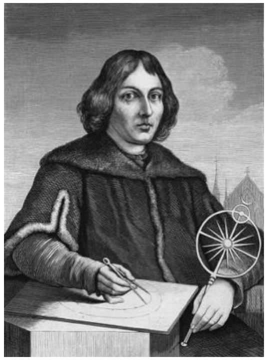
- in the 7th-14th centuries, the Islamic Empire covered parts of Europe, northern Africa, the Middle East, and western Asia
- Greek writings were preserved and advanced by Arab scholars
- the term "algorithm" is named after Persian scholar Muhammad ibn Musa al-Khwarismi

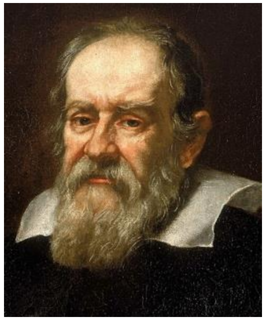


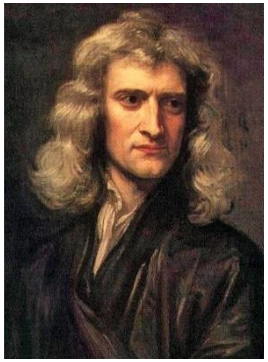
Scientific Revolution

the Renaissance (15th-16th centuries) was instigated by the rediscovery of Greek science

- Greek knowledge was rediscovered by Crusaders to the Middle East; retrieved from medieval monasteries
- Leonardo da Vinci (1452-1519) was artist, astronomer, geometer, engineer, …
- Gutenberg's printing press made the broad dissemination of knowledge possible







the Scientific Revolution (16th-17th centuries) was brought about by a period of intellectual upheaval in Europe

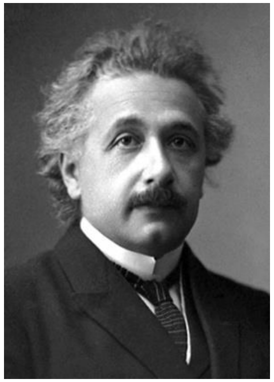
- the Protestant Reformation, new world exploration, …
- the cultural environment allowed for questioning religious and scientific dogma – the universe was viewed as a complex machine that could be understood through observation and experimentation

- Copernicus proposed a sun-centered cosmology (1543)
  - Kepler refined the heliocentric model, using elliptical orbits (1609)
- Galileo pioneered the use of experimentation to validate observational theories
  - father of modern science (as well as modern physics and astronomy)
- Newton described universal gravitation, laws of motion (1687)

Modern Science

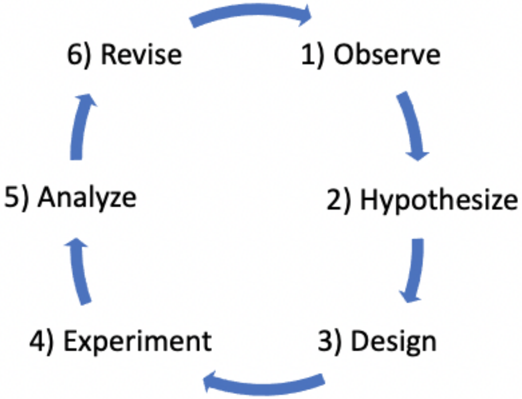
the Scientific Revolution established science as the preeminent source for the growth of knowledge

- biology: Pasteur, Watson & Crick, …
- chemistry: Dalton, Mendeleev, Curie, …
- physics: Maxwell, Curie, Einstein, …



the scientific method provides the common process by which modern science is conducted

1. **O**bserve a phenomenon
2. **H**ypothesize how it works
3. **D**esign an experiment to test it
4. **E**xperiment to confirm/deny
5. **A**nalyze the results
6. **R**evise or refine the hypothesis



generally, the process repeats since the results may lead to revisions to the hypothesis or experiment

Scientific Method

EXAMPLE: understanding planetary motion

1. Observe that some lights in the night sky move different, faster than others.
2. Hypothesize that those lights (planets) are closer to the earth.
3. Develop a model of motion (say, Copernicus' circular orbits around the sun).
4. Conduct the experiment to see telescopic observations match the model.
5. Analyze the results and see that the observations are close but not exact – revise the model to use Kepler's elliptical orbits and repeat.

the scientific method can be applied to real-world situations as well

- EXAMPLE: an auto mechanic observes a misfiring engine, hypothesizes that the cause is a bad spark plug and designs an experiment (replace it) to test
- EXAMPLE: a programmer observes a program that doesn't work, hypothesizes the cause if a malformed statement and designs an experiment (bug fix) to test

reproducibility is essential to the scientific method

- the same experiment, under the same conditions, should produce the same result
- if a scientific discovery is not reproducible, it will not be accepted

consistency is a measure of how close the results are each time you conduct the experiment

accuracy is a measure of how close the results are to the correct (or expected) value

Computational Thinking

the scientific method is designed for understanding a phenomenon

- may not be directly applicable to real-world problems solving

computational thinking is a problem-solving approach that involves expressing problems and their solutions in ways that a computer could execute

- first coined by Papert in 1980, made popular by Jeannette Wing in 2006
- computational thinking has been recognized by many as an essential 21st century skill (along with critical thinking, communication, collaboration, and creativity)

high-level characteristics of computational thinking

- DECOMPOSITION – breaking a large, complex problem into smaller, more manageable problems
- PATTERN MATCHING – recognizing how solutions to similar problems can be applied to new problems
- ABSTRACTION – focusing on important details while ignoring irrelevant information
- ALGORITHMS – designing and implementing the solution in the form of an algorithm

real-world example: assembling a bookcase

CT Example

consider the task of finding the oldest person in a room

- there are a number of issues to consider, different approaches you could take

DECOMPOSITION

- tasks that will need to be performed:
  - ✓ systematically process each person
  - ✓ be able to determine a person's age
  - ✓ record the names/ages so that will know the oldest at the end

PATTERN MATCHING

- learn from past experiences that were similar
  - ✓ lining the people up will make it easier to cover everyone
  - ✓ pencil & paper are effective for simple tasks like these

ABSTRACTION

- there are a lot of characteristics we don't care about (hair color, middle initial)
  - ✓ if born on the same day, we will consider the same age
  - ✓ if more than one "oldest" person, any one will do

ALGORITHMS

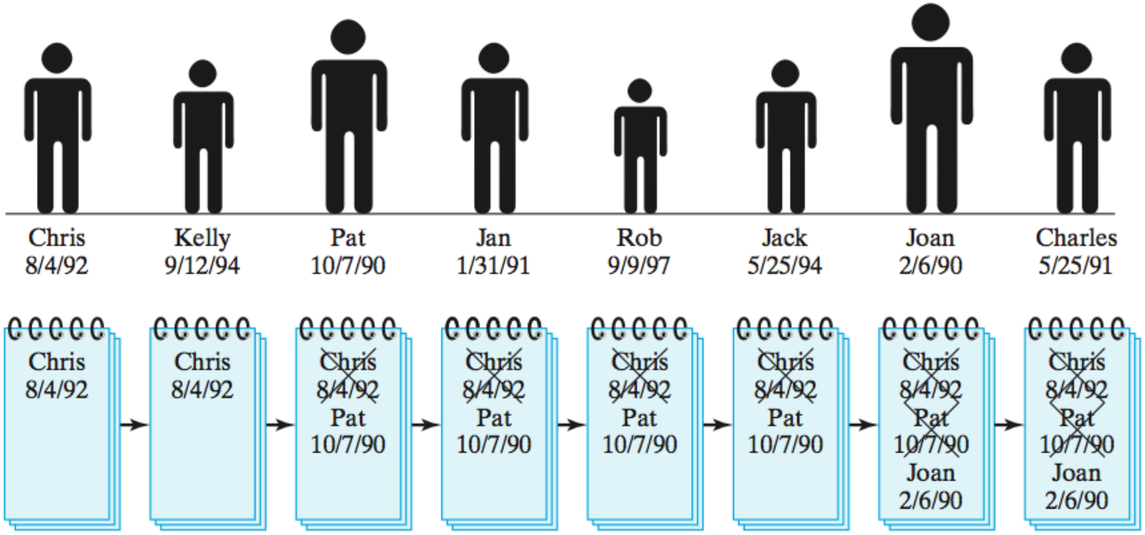
- can now devise an algorithm, step-by-step sequence of instructions, to solve this

Algorithm 1

Finding the oldest person (algorithm 1)

1. line up all the people along one wall
2. ask the first person to state his or her name and birthday, then write this information down on a piece of paper
3. for each successive person in line:
   1. ask the person for his or her name and birthday
   2. if the stated birthday is earlier than the birthday on the paper, cross out old information and write down the name and birthday of this person

when you reach the end of the line, the name and birthday of the oldest person will be written on the paper


Chris 8/4/92
Kelly 9/12/94
Pat 10/7/90
Jan 1/31/91
Rob 9/9/97
Jack 5/25/94
Joan 2/6/90
Charles 5/25/91
Chris 8/4/92
Chris 8/4/92
~~Chris 8/4/92~~ Pat 10/7/90
~~Chris 8/4/92~~ Pat 10/7/90
~~Chris 8/4/92~~ Pat 10/7/90
~~Chris 8/4/92~~ Pat 10/7/90
~~Chris 8/4/92~~ ~~Pat 10/7/90~~ Joan 2/6/90
~~Chris 8/4/92~~ ~~Pat 10/7/90~~ Joan 2/6/90

Algorithm 1 Analysis

algorithm 1 works, since the oldest person will eventually be found & recorded

- the amount of time to find the oldest person is *proportional to the number of people*
- if you double the amount of people, the time needed to find the oldest person will also double

for example, assume it takes 10 seconds to compare birthdays

- 8 people → 10*8 = 80 seconds (1.33 minutes)
- 16 people → 10*16 = 160 seconds (2.67 minutes)
- 32 people → 10*32 = 320 seconds (5.33 minutes)

  . . .
- 100 people → 10*100 = 1,000 seconds (16.67 minutes)

  . . .
- 400 people → 10*400 = 4,000 seconds (1 hour & 6.67 minutes)

this algorithm works, but it does not scale well if the number of people gets big

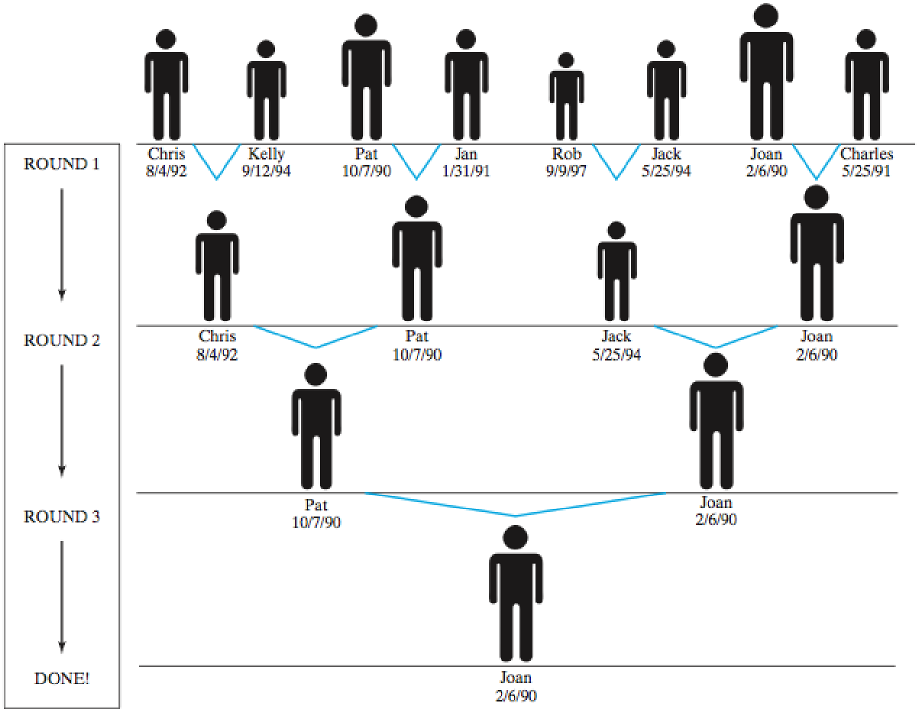
- consider a more complex but also more efficient algorithm

Algorithm 2

Finding the oldest person (algorithm 2)

1. line up all the people along one wall
2. as long as there is more than one person in the line, repeatedly
   1. have the people pair up (1st with 2nd, 3rd with 4th, etc) – if there is an odd number of people, the last person will be without a partner
   2. ask each pair of people to compare their birthdays
   3. request that the younger of the two leave the line

when there is only one person left in line, that person is the oldest



Algorithm 2 Analysis

algorithm 2 works, since the oldest person in a pair never sits and the process eventually reduces down to that oldest person

- the time needed to find the oldest person is *proportional to the number of rounds it takes to shrink the line down to one person* (since all pair comparisons in a round take place simultaneously)
  - the number of rounds is the number of times the people can repeatedly be divided in half (mathematically speaking, the $\log_2$ of the number of people)
- if you double the amount of people, the time needed to find the oldest person increases by the cost of one more comparison

for example, assume it takes 10 seconds to compare birthdays

- 8 people → $10 * \lceil \log_2 8 \rceil = 10*3 = 30$ seconds (0.5 minutes)
- 16 people → $10 * \lceil \log_2 16 \rceil = 10*4 = 40$ seconds (0.67 minutes)
- 32 people → $10 * \lceil \log_2 32 \rceil = 10*5 = 50$ seconds (0.83 minutes)

  . . .
- 100 people → $10 * \lceil \log_2 100 \rceil = 10*7 = 70$ seconds (1.16 minutes)

  . . .
- 400 people → $10 * \lceil \log_2 400 \rceil = 10*9 = 90$ seconds (1.5 minutes)

Multiple Solutions

# of people	time for Algorithm 1	time for Algorithm 2
8	80 sec	30 sec
16	160 sec	40 sec
32	320 sec	50 sec
…		
100	1,000 sec	70 sec
…		
400	4,000 sec	90 sec

many real-world problems can be solved in multiple ways

- when presented with this problem, most people would devise a solution similar to Algorithm 1 (with many different variations possible)
  - it is simple to describe and understand
  - it is reasonably fast for small numbers of people
- developing Algorithm 2 requires considerable experience solving similar problems
  - must be able to ABSRACT the relevant features of this problem, PATTERN MATCH with past solutions to similar problems, and DECOMPOSE the solution to fit this new problem.

like most endeavors, the more computational thinking you do, they better you become at it

Data Types

each unit of information processed by a computer belongs to a general category or *data type*

- JavaScript has three predefined data types
  1. string for representing text values (e.g., 'abcd', 'two words')
  2. number for representing numeric values (e.g. 12, 3.99)
  3. Boolean for representing logical values (`true` or `false`) LATER

each data type is associated with a specific set of predefined operators that may be used by programmers to manipulate values of that type

- e.g., we have seen string concatenation via +
- similarly, standard operators are predefined for numbers
  addition (+), subtraction (-), multiplication (*), division (/)

text boxes allow the user to enter strings and access those entries

- there is an equivalent element for numbers: *number box*

Number Box

a number box is an `input` element (similar to text box)

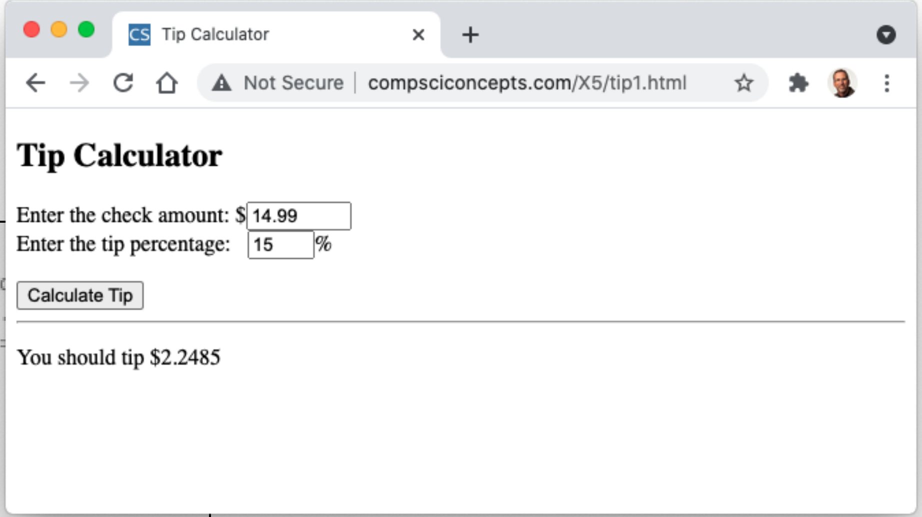
- `type` attribute is assigned "number" (instead of "text")
- optional value attribute is same as for text box – specifies default value that appears in the box
- instead of `size` attribute, number boxes specify min and max values – the size is adjusted to fit that range

```
<input type="number" id="numBox" min=0 max=100 value=50>
```

to access the number in a number box, use `valueAsNumber` attribute

```
numBox.valueAsNumber
```

Tip Calculator

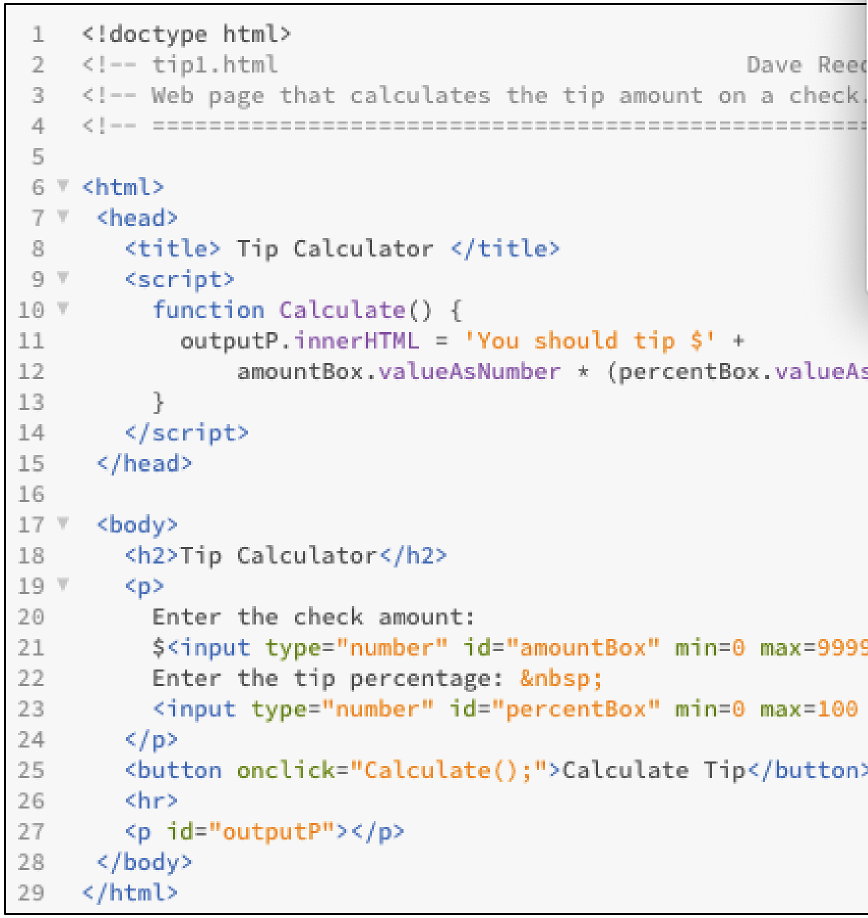


Tip Calculator

Enter the check amount: $ 14.99
Enter the tip percentage: 15 %

Calculate Tip

You should tip $2.2485



```
<!doctype html>
<!-- tip1.html                                Dave Ree
<!-- Web page that calculates the tip amount on a check.
<!-- ==========================================================

<html>
 <head>
   <title> Tip Calculator </title>
   <script>
     function Calculate() {
       outputP.innerHTML = 'You should tip $' +
           amountBox.valueAsNumber * (percentBox.valueAsNumber/100);
     }
   </script>
 </head>

 <body>
   <h2>Tip Calculator</h2>
   <p>
     Enter the check amount:
     $<input type="number" id="amountBox" min=0 max=9999.99> <br>
     Enter the tip percentage: &nbsp;
     <input type="number" id="percentBox" min=0 max=100 value=15>%
   </p>
   <button onclick="Calculate();">Calculate Tip</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```

here, the user enters a check amount and tip percentage (default is 15%)

when button is clicked, accesses those numbers, calculates & displays tip

`valueAsNumber` vs. `value`

similar to text boxes, number boxes have a `value` attribute

- the value attribute always returns the contents of the box *as a string*
- this is NOT what you normally want when accessing a number box

e.g., suppose the user enters 12 in a number box named `numBox`

`numBox.value` evaluates to '12'

`(numBox.value + 1)` evaluates to '121' ???

recall, when + is applied to 2 numbers, addition: 1 + 2 = 3
when + is applied to 2 strings, concatenation: 'a' + 'b' = 'ab'
when + is applied to a string and a number, it converts the number to a string and concatenates: '12' + 1 = '12' + '1' = '121'

BE CAREFUL TO ALWAYS USE `valueAsNumber` TO ACCESS NUMBER BOX CONTENTS

Variables

a *variable* is a name used to symbolize a dynamic (changeable) value

- as before, assign a value using '=':

```
VARIABLE = VALUE;
```

variables are commonly used to simplify code by:

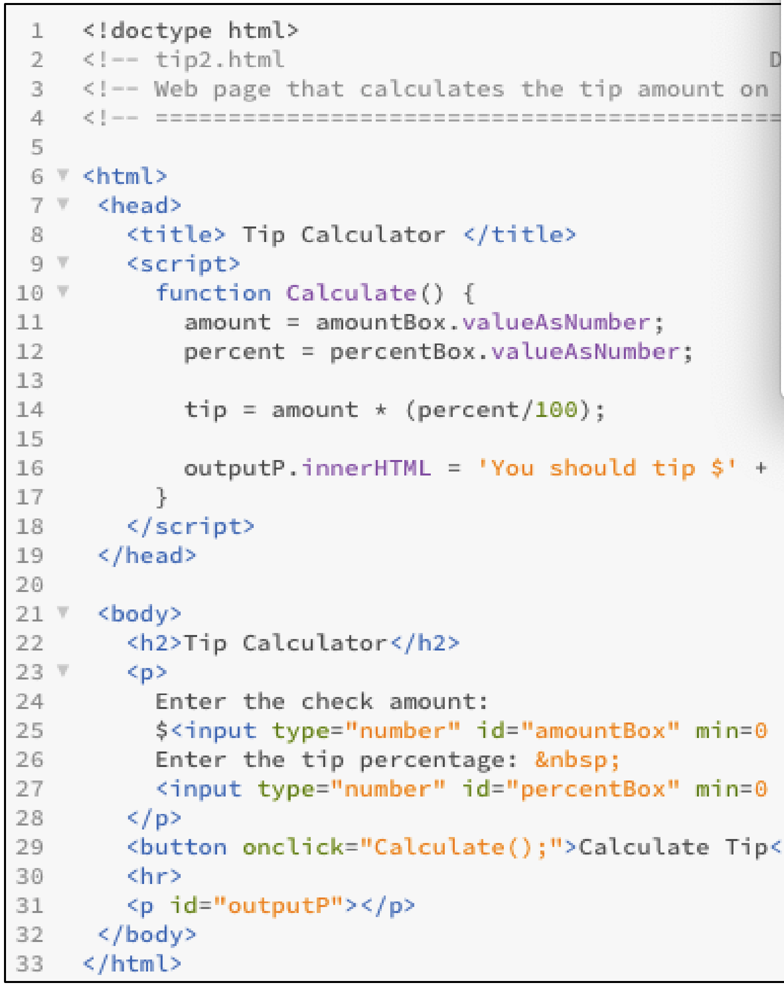
1. storing number values from boxes

```
amount = amountBox.valueAsNumber;

percent = percentBox.valueAsNumber;
```

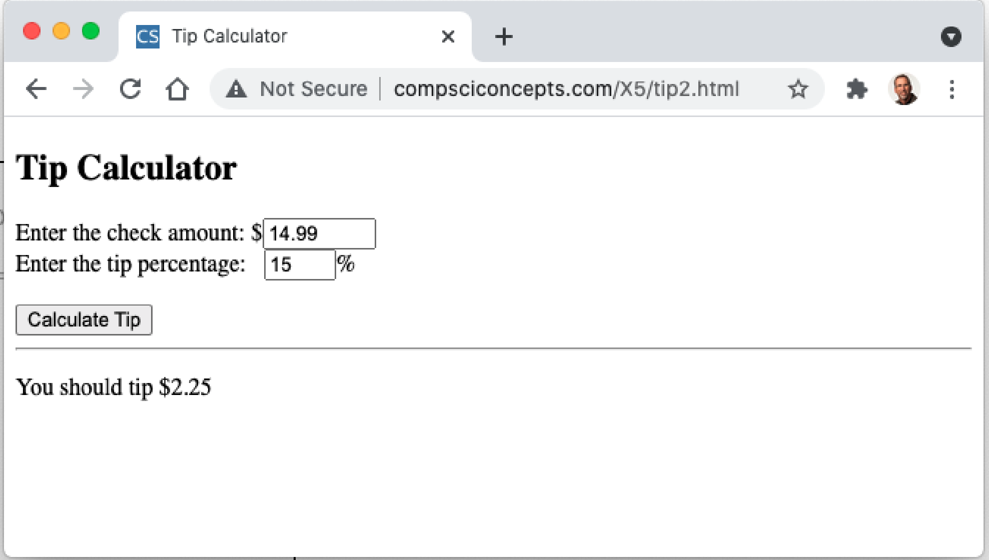
2. or, storing the results of computations

```
tip =  amount * (percent/100);
```

Tip Calculator



```
<!doctype html>
<!-- tip2.html                                        D
<!-- Web page that calculates the tip amount on
<!-- ==============================================

<html>
 <head>
   <title> Tip Calculator </title>
   <script>
     function Calculate() {
       amount = amountBox.valueAsNumber;
       percent = percentBox.valueAsNumber;

       tip = amount * (percent/100);

       outputP.innerHTML = 'You should tip $' + tip.toFixed(2);
     }
   </script>
 </head>

 <body>
   <h2>Tip Calculator</h2>
   <p>
     Enter the check amount:
     $<input type="number" id="amountBox" min=0 max=9999.99> <br>
     Enter the tip percentage: &nbsp;
     <input type="number" id="percentBox" min=0 max=100 value=15>%
   </p>
   <button onclick="Calculate();">Calculate Tip</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```



Tip Calculator

Enter the check amount: $ 14.99
Enter the tip percentage: 15 %

Calculate Tip

You should tip $2.25

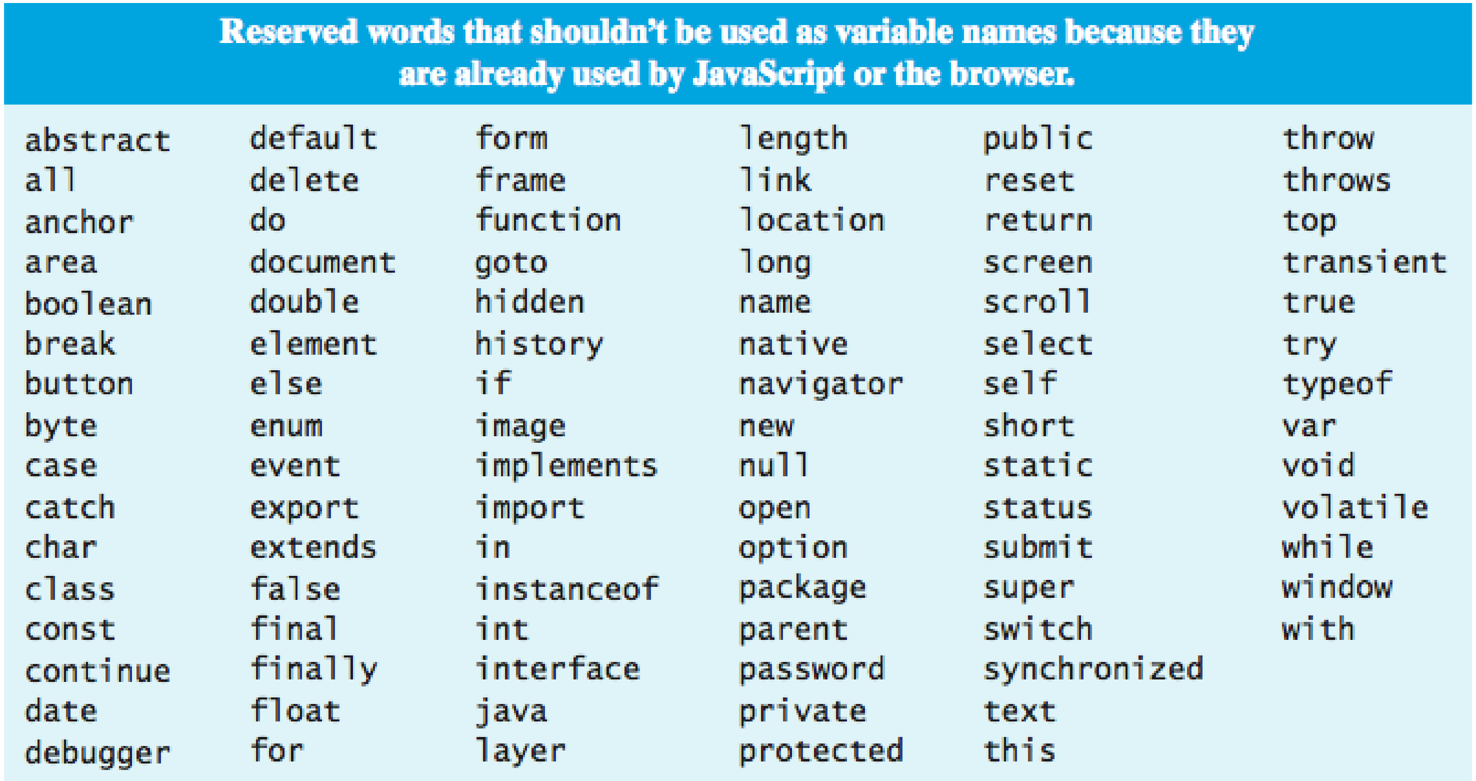
code is simplified by use of variables

tip.toFixed(2) rounds the tip to 2 decimal places

Variable Names

a variable name should start with a lowercase letter, consist of letters & digits

- a variable name should be chosen to be descriptive of its purpose
- e.g., `mysteryImg`, `outputSpan`



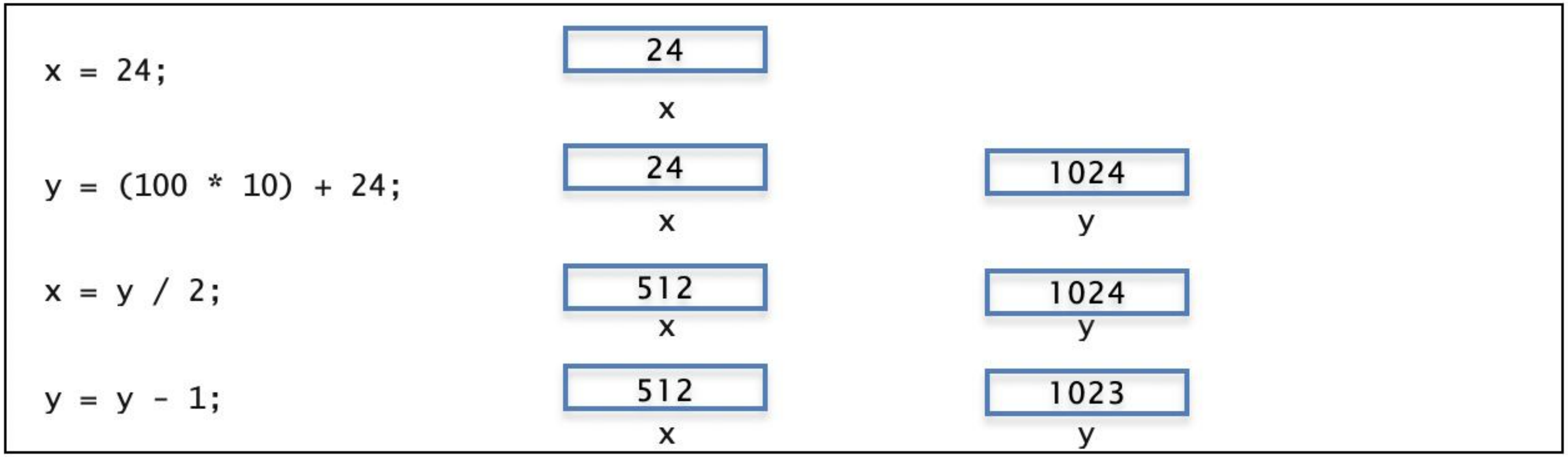
Reserved words that shouldn't be used as variable names because they are already used by JavaScript or the browser.

abstract
all
anchor
area
boolean
break
button
byte
case
catch
char
class
const
continue
date
debugger
default
delete
do
document
double
element
else
enum
event
export
extends
false
final
finally
float
for
form
frame
function
goto
hidden
history
if
image
implements
import
in
instanceof
int
interface
java
layer
length
link
location
long
name
native
navigator
new
null
open
option
package
parent
password
private
protected
public
reset
return
screen
scroll
select
self
short
static
status
submit
super
switch
synchronized
text
this
throw
throws
top
transient
true
try
typeof
var
void
volatile
while
window
with

Variables & Assignments

variables can be assigned various types of values, including numbers and mathematical expressions

- each variable has a memory cell associated with it, where a value can be stored
- when an expression appears on the right-hand side, it is evaluated and the result is assigned to the variable (i.e., stored in its memory cell)
- when a variable appears in an expression, its value (i.e., the value stored in its memory cell) is accessed and substituted into the expression



```
x = 24;

y = (100 * 10) + 24;

x = y / 2;

y = y - 1;
```

reminder: '=' is not the equality operator

to avoid confusion: read '=' as 'gets', as in *'y gets y-1'*

Number Representation

useful facts about JavaScript numbers

- to improve readability, very large or very small number are displayed in *scientific notation:* `XeY` represents the value $X \times 10^Y$
  - e.g., `1e24` → 1×10^{24} → `1000000000000000000000000`

- JavaScript stores all numbers in memory cells of a fixed size (64 bits)
  - as a result, only a finite number of values can be represented
  - e.g., `1e308` can be represented, but `1e309` is treated as `Infinity`
    `1e-323` can be represented, but `1e-324` is treated as `0`

- even within the range 1e-323 . . . 1e309, not all numbers can be represented
  - note that between any two numbers lie infinitely more numbers!
  - JavaScript can represent approximately 17 significant digits
  - e.g.., `0.9999999999999999` can be represented exactly
    `0.99999999999999999` is rounded up to `1`

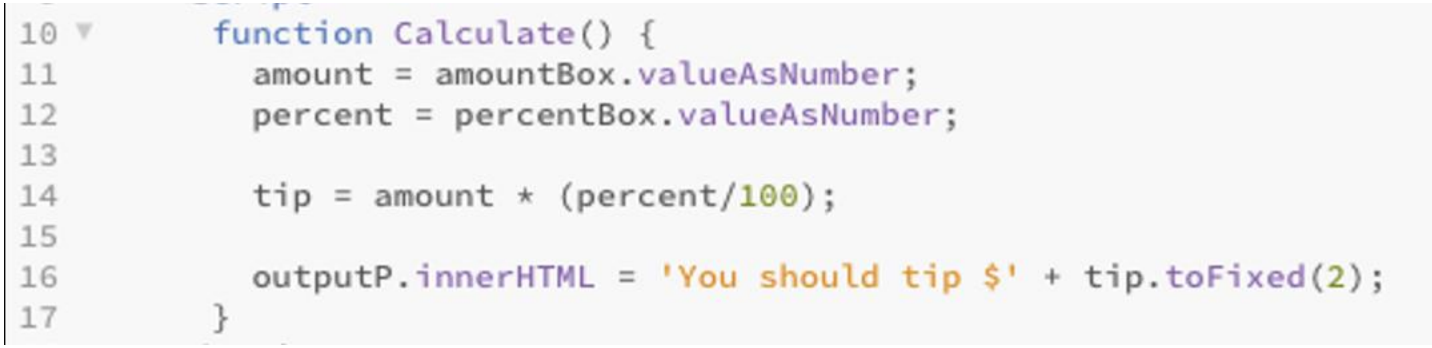
Patterns & Spacing

note the spacing in the `Calculate` function

- blank lines are ignored by the browser, but are helpful to a developer
- here, there are three main tasks
  1. get the (numeric) contents of the text boxes
  2. perform a computation on those numbers
  3. display the result of the computation in the page

- inserting blank lines to separate these tasks makes the code easier to understand
  - similar to dividing an essay into paragraphs

- be aware: this same pattern will reappear in many pages



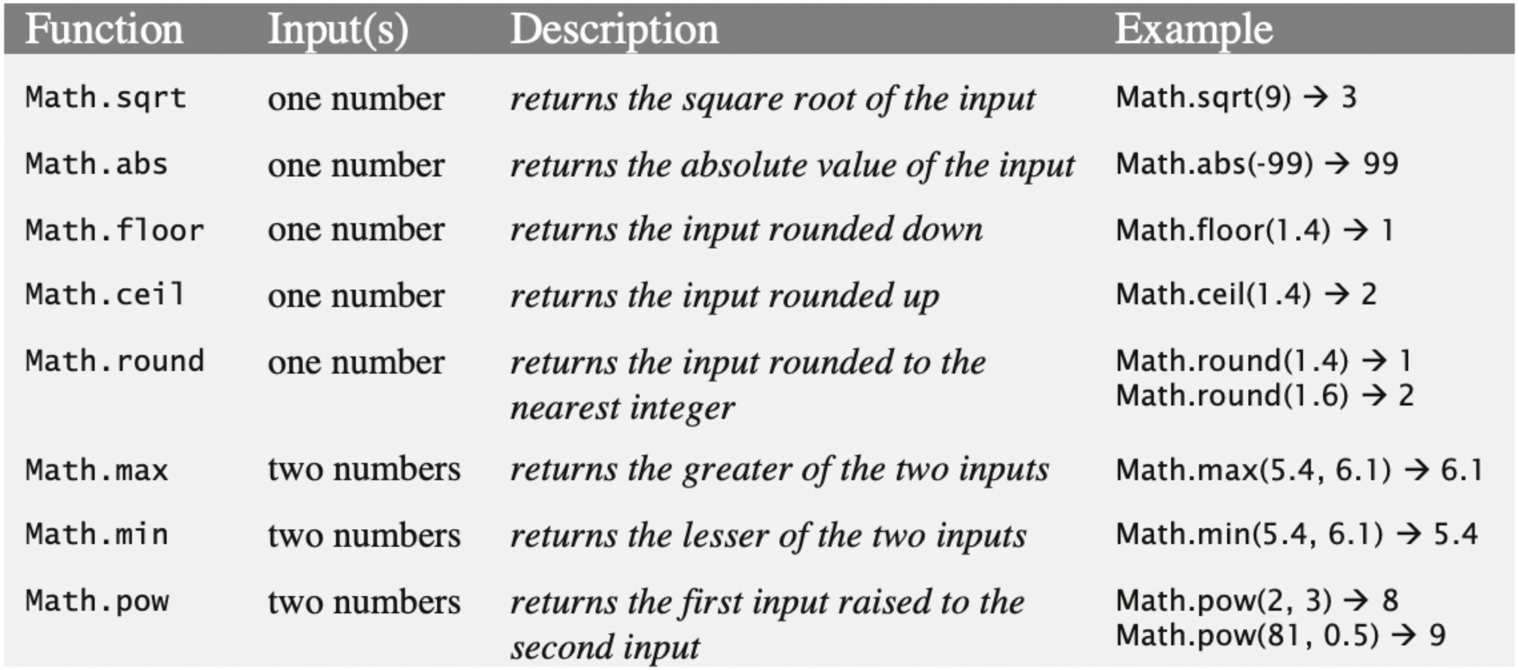
```
function Calculate() {
  amount = amountBox.valueAsNumber;
  percent = percentBox.valueAsNumber;

  tip = amount * (percent/100);

  outputP.innerHTML = 'You should tip $' + tip.toFixed(2);
}
```

Predefined Functions

in JavaScript, a function is applied to inputs via a *function call*

```
num = Math.sqrt(25);
```

here, `Math.sqrt` is being called with input 25; it returns the output 5

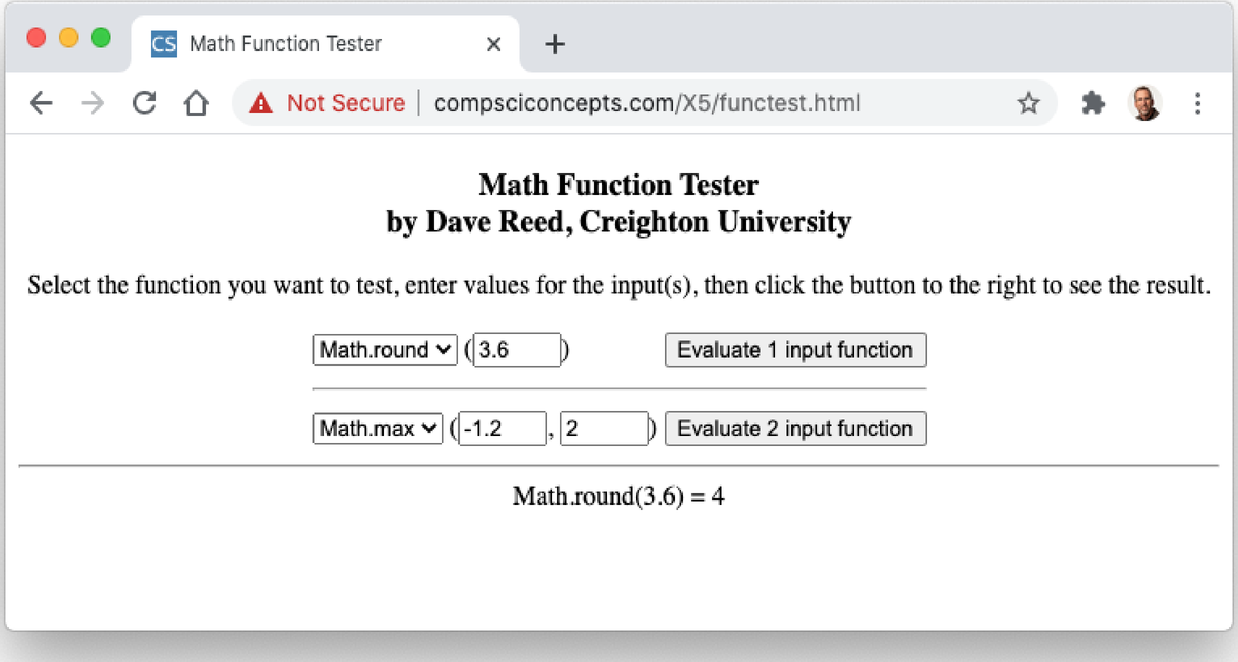


Function	Input(s)	Description	Example
`Math.sqrt`	one number	*returns the square root of the input*	Math.sqrt(9) → 3
`Math.abs`	one number	*returns the absolute value of the input*	Math.abs(-99) → 99
`Math.floor`	one number	*returns the input rounded down*	Math.floor(1.4) → 1
`Math.ceil`	one number	*returns the input rounded up*	Math.ceil(1.4) → 2
`Math.round`	one number	*returns the input rounded to the nearest integer*	Math.round(1.4) → 1 Math.round(1.6) → 2
`Math.max`	two numbers	*returns the greater of the two inputs*	Math.max(5.4, 6.1) → 6.1
`Math.min`	two numbers	*returns the lesser of the two inputs*	Math.min(5.4, 6.1) → 5.4
`Math.pow`	two numbers	*returns the first input raised to the second input*	Math.pow(2, 3) → 8 Math.pow(81, 0.5) → 9

functest

the `functest.html` page is provided for you to explore the different math functions

- select the function from a pull-down menu & inter the input(s) in text box(es)
- click the button to see the function call and its output



Math Function Tester
by Dave Reed, Creighton University

Select the function you want to test, enter values for the input(s), then click the button to the right to see the result.

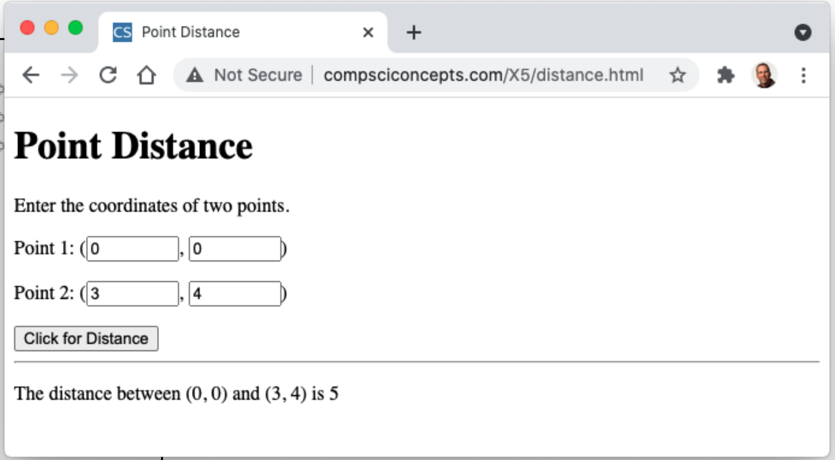
Math.round (3.6) Evaluate 1 input function

Math.max (-1.2 , 2) Evaluate 2 input function

Math.round(3.6) = 4

Point Distance

```
<!doctype html>
<!-- distance.html                                Dave Reed -->
<!-- This page calculates the distance between two points. -->
<!-- ===================================================== -->

<html>
 <head>
   <title> Point Distance </title>
   <script>
     function ShowDistance() {
       x1 = x1Box.valueAsNumber;
       y1 = y1Box.valueAsNumber;
       x2 = x2Box.valueAsNumber;
       y2 = y2Box.valueAsNumber;

       dist = Math.sqrt(Math.pow(x1-x2, 2) + Math.pow(y1-y2, 2));

       outputP.innerHTML = 'The distance between (' + x1 + ', ' + y1 +
                           ') and (' + x2 + ', ' + y2 + ') is ' + dist;
     }
   </script>
 </head>

 <body>
   <h1>Point Distance</h1>
   <p>Enter the coordinates of two points.</p>
   <p>Point 1:
      (<input type="number" id="x1Box" min=-9999.9 max=9999.9 value=0>,
       <input type="number" id="y1Box" min=-9999.9 max=9999.9 value=0>)
   </p>
   <p>Point 2:
      (<input type="number" id="x2Box" min=-9999.9 max=9999.9 value=3>,
       <input type="number" id="y2Box" min=-9999.9 max=9999.9 value=4>)
   </p>
   <button onclick="ShowDistance();">Click for Distance</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```



Point Distance

Enter the coordinates of two points.

Point 1: (0, 0)

Point 2: (3, 4)

Click for Distance

The distance between (0, 0) and (3, 4) is 5

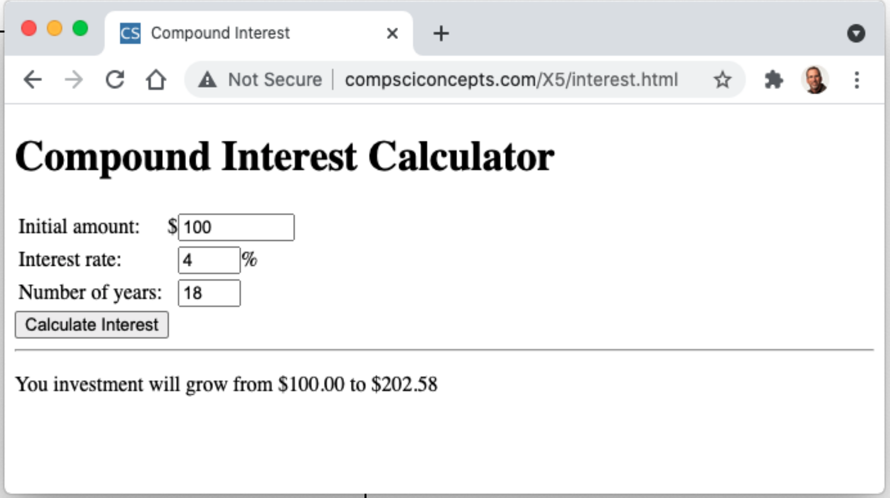
user enters two points (x1, y1) and (x2, y2) in text boxes

when button is clicked, the distance between those two points is calculated and displayed

$$\sqrt{(x1-x2)^2+(y1-y2)^2}$$

Compound Interest

```
<!doctype html>
<!-- interest.html                                     Dave Reed -->
<!-- This page calculates compund interest on an investment. -->
<!-- ======================================================= -->

<html>
 <head>
   <title> Compound Interest </title>
   <script>
     function ShowInterest() {
       amount = amountBox.valueAsNumber;
       rate = rateBox.valueAsNumber;
       years = yearsBox.valueAsNumber;

       total = amount * Math.pow(1+rate/100, years);

       outputP.innerHTML = 'You investment will grow from $' +
                           amount.toFixed(2) + ' to $' + total.toFixed(2);
     }
   </script>
 </head>

 <body>
   <h1>Compound Interest Calculator</h1>
   <table>
     <tr><td>Initial amount: </td>
         <td>$<input type="number" id="amountBox" min=0 max=999999.99 value=100></td></tr>
     <tr><td>Interest rate:</td>
         <td>&nbsp;&nbsp;<input type="number" id="rateBox" min=0 max=100 value=3.5>%</td></tr>
     <tr><td>Number of years:
         <td>&nbsp;&nbsp;<input type="number" id="yearsBox" min=0 max=999 value=12></td></tr>

   </table>
   <button onclick="ShowInterest();">Calculate Interest</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```



Compound Interest Calculator

Initial amount:	$ 100
Interest rate:	4 %
Number of years:	18

You investment will grow from $100.00 to $202.58

calculates compound interest on an investment

user enters initial amount, interest rate & number of years

total =
 amount*(+rate/100)^{years}

Math.random

in addition to the above math functions, JavaScript provides a function for generating a random number

- Math.random has no inputs, returns a random real number from the range [0, 1)
- note: smallest possible value = 0.0; largest possible value = 0.99999…

Math.random() → 0.3428794638

Math.random() → 0.8776243657

technically, it returns a *pseudo-random* number, since it uses a complex algorithm to generate numbers that appear random (using hidden inputs like the current time in milliseconds)

by itself, Math.random is not very useful

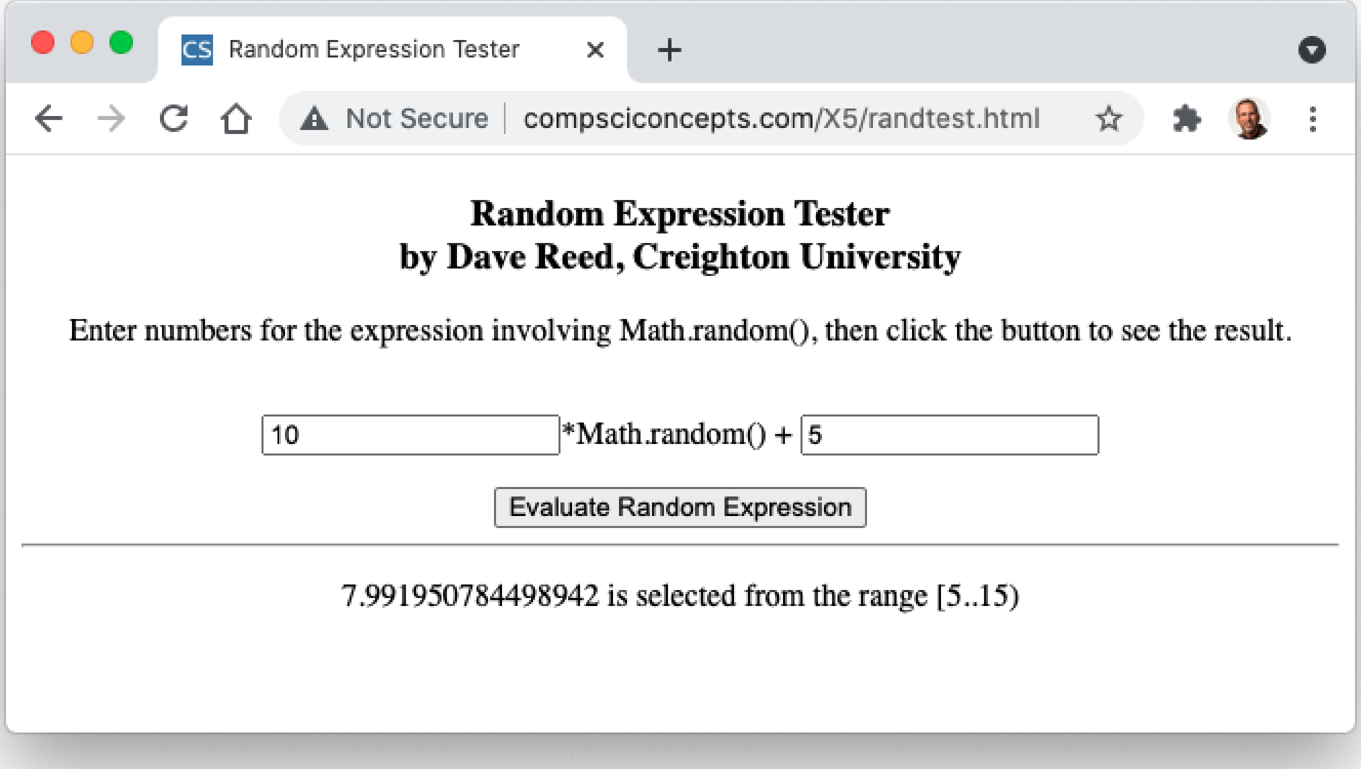
- but, can use in expressions to expand or shift the range

2*Math.random() → 2*(number from [0, 1) → number from [0, 2)

Math.random()+10 → (number from [0, 1)) + 10 → number from [10, 11)

X*Math.random()+Y → number from [Y, X+Y)

randtest



Random Expression Tester
Not Secure
compsciconcepts.com/X5/randtest.html

Random Expression Tester
by Dave Reed, Creighton University

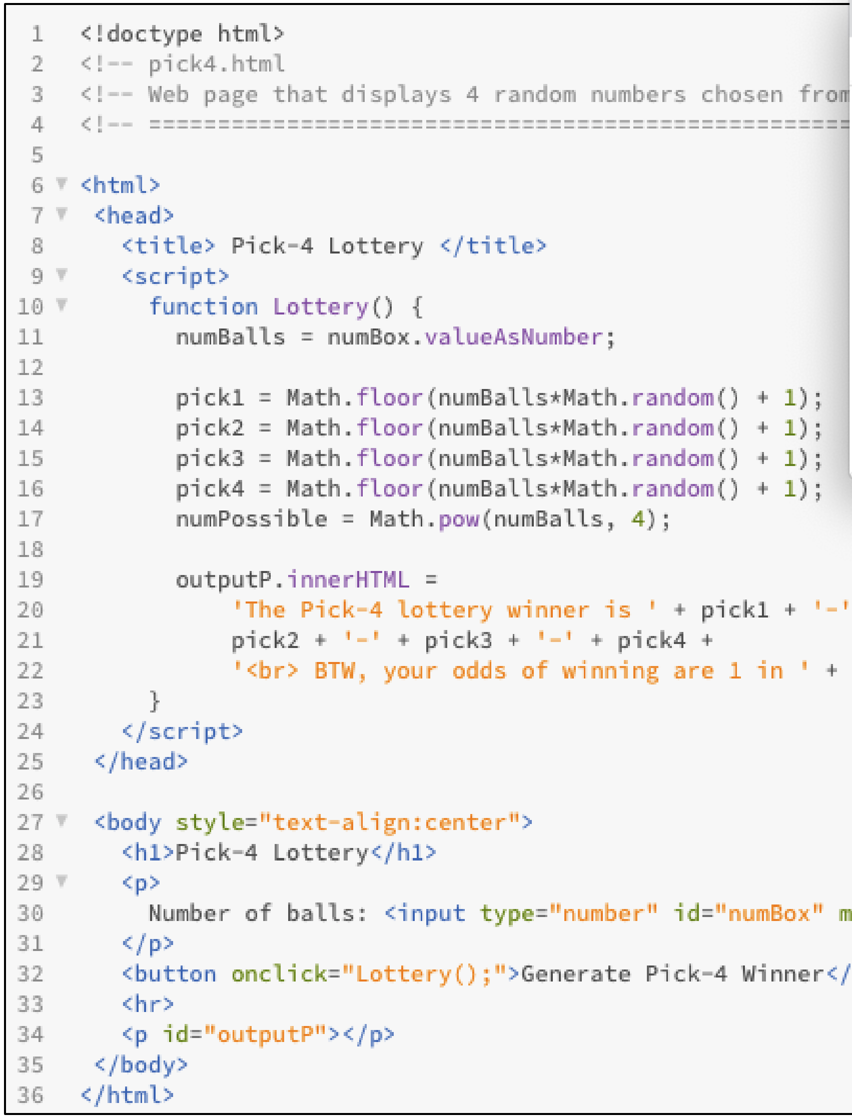
Enter numbers for the expression involving Math.random(), then click the button to see the result.

10 *Math.random() + 5

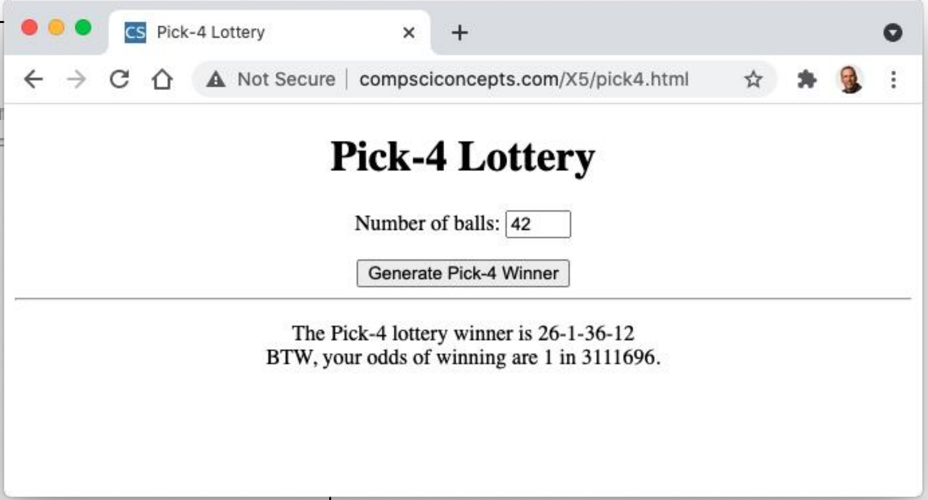
Evaluate Random Expression

7.991950784498942 is selected from the range [5..15)

Pick-4 Example



```
<!doctype html>
<!-- pick4.html
<!-- Web page that displays 4 random numbers chosen from
<!-- ==================================================

<html>
 <head>
   <title> Pick-4 Lottery </title>
   <script>
     function Lottery() {
       numBalls = numBox.valueAsNumber;

       pick1 = Math.floor(numBalls*Math.random() + 1);
       pick2 = Math.floor(numBalls*Math.random() + 1);
       pick3 = Math.floor(numBalls*Math.random() + 1);
       pick4 = Math.floor(numBalls*Math.random() + 1);
       numPossible = Math.pow(numBalls, 4);

       outputP.innerHTML =
           'The Pick-4 lottery winner is ' + pick1 + '-' +
           pick2 + '-' + pick3 + '-' + pick4 +
           '<br> BTW, your odds of winning are 1 in ' + numPossible + '.';
     }
   </script>
 </head>

 <body style="text-align:center">
   <h1>Pick-4 Lottery</h1>
   <p>
     Number of balls: <input type="number" id="numBox" min=0 max=999 value=42>
   </p>
   <button onclick="Lottery();">Generate Pick-4 Winner</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```



Pick-4 Lottery

Number of balls: 42

Generate Pick-4 Winner

The Pick-4 lottery winner is 26-1-36-12
BTW, your odds of winning are 1 in 3111696.

this page simulates picking 4 numbered balls from lottery bins

to pick a random integer from 1 to numBalls

1. pick a # from the range [1, numBalls+1)
2. then round down

Algorithms

the central concept underlying all computation is that of the *algorithm*

- an algorithm is a step-by-step sequence of instructions for carrying out some task

programming can be viewed as the process of designing and implementing algorithms that a computer can carry out

- a programmer's job is to:
  - create an algorithm for accomplishing a given objective, then
  - translate the individual steps of the algorithm into a programming language that the computer can understand

example: programming in JavaScript

- we have written programs that instruct the browser to carry out a particular task
- given the proper instructions, the browser is able to understand and produce the desired results

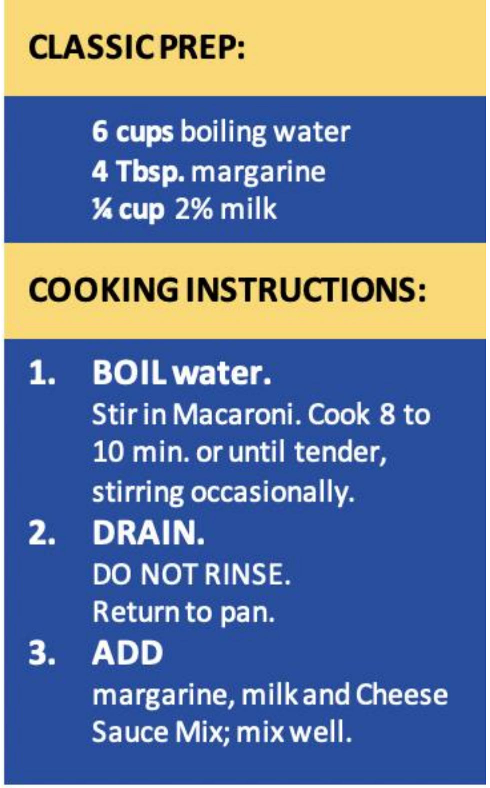
Algorithms in the Real World

the use of algorithms is not limited to the domain of computing

- e.g., recipes for baking cookies
- e.g., directions to your house

there are many unfamiliar tasks in life that we could not complete without the aid of instructions

- in order for an algorithm to be effective, it must be stated in a manner that its intended executor can understand
  - a recipe written for a master chef will look different than a recipe written for a college student
- as you have already experienced, computers are more demanding with regard to algorithm specifics than any human could be



CLASSIC PREP:

6 cups boiling water
4 Tbsp. margarine
¼ cup 2% milk

COOKING INSTRUCTIONS:

1. **BOIL water.**
   Stir in Macaroni. Cook 8 to 10 min. or until tender, stirring occasionally.
2. **DRAIN.**
   DO NOT RINSE.
   Return to pan.
3. **ADD**
   margarine, milk and Cheese Sauce Mix; mix well.

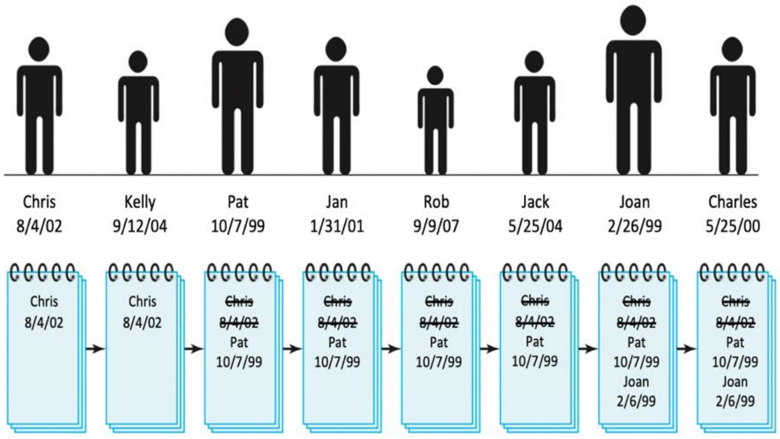
Algorithms & CT

recall the four characteristics of computations thinking: DECOMPOSITION, PATTERN RECOGNITION, ABSTRACTION and ALGORITHM

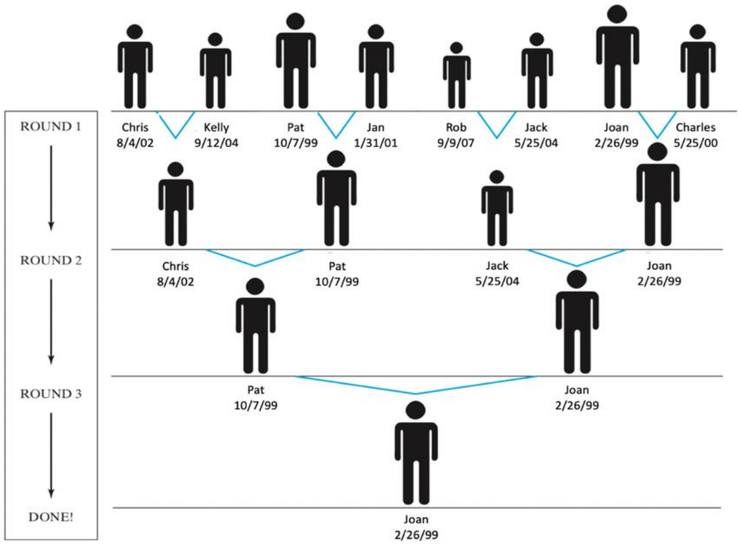
- the final step in CT is formalizing the solution as an algorithm
- a clearly stated algorithm is a blueprint for future problem solving

consider the 2 algorithms (from Ch. C5) for finding the oldest person in a room

Algorithm 1



Algorithm 2



Algorithm Analysis

determining which algorithm is "better" is not always clear cut

- it depends upon what features are most important to you
  - if you want to be sure it works, choose the clearer algorithm
  - if you care about the time or effort required, need to analyze performance

algorithm 1 involves asking each person’s birthday and then comparing it to the birthday written on the page

- the amount of time to find the oldest person is *proportional to the number of people*
- if you double the amount of people, the time needed to find the oldest person will also double

algorithm 2 allows you to perform multiple comparisons simultaneously

- the time needed to find the oldest person is *proportional to the number of rounds it takes to shrink the line down to one person*
  - which turns out to be the $\log_2$ of the number of people
- if you double the amount of people, the time needed to find the oldest person increases by the time required for one more round

Algorithm Analysis (cont.)

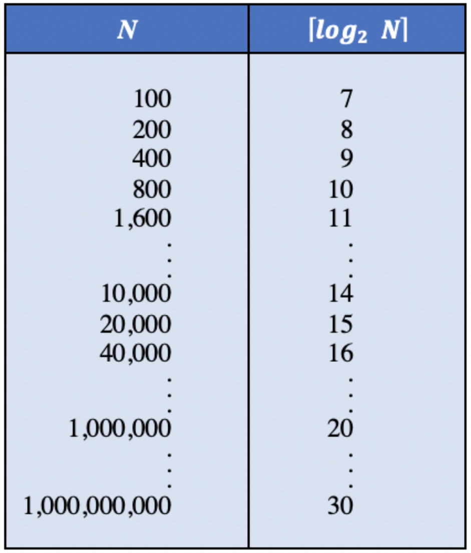
when the problem size is large, performance differences can be dramatic

for example, assume it takes 10 seconds to compare birthdays

- for algorithm 1:
  - 100 people → 10*100 = 1,000 seconds
  - 200 people → 10*200 = 2,000 seconds
  - 400 people → 10*400 = 4,000 seconds

    . . .
  - 1,000 people → 10*1,000 = 10,000 seconds
- for algorithm 2:
  - 100 people → $10*\lceil \log_2 100 \rceil$ = 70 seconds
  - 200 people → $10*\lceil \log_2 200 \rceil$ = 80 seconds
  - 400 people → $10*\lceil \log_2 400 \rceil$ = 90 seconds

    . . .
  - 1,000 people → $10*\lceil \log_2 1{,}000{,}000 \rceil$ = 100 seconds



N	$\lceil \log_2 N \rceil$
100	7
200	8
400	9
800	10
1,600	11
⋮	⋮
10,000	14
20,000	15
40,000	16
⋮	⋮
1,000,000	20
⋮	⋮
1,000,000,000	30

Big-Oh Notation

to represent an algorithm's performance in relation to the size of the problem, computer scientists use what is known as *Big-Oh* notation

- executing an O(N) algorithm requires time proportional to the size of problem
  - given an O(N) algorithm, doubling the problem size doubles the work
- executing an O(log N) algorithm requires time proportional to the logarithm of the problem size
  - given an O(log N) algorithm, doubling the problem size adds a constant amount of work

based on our previous analysis:

- algorithm 1 is classified as O(N)
- algorithm 2 is O(log N)

Another Algorithm Example

SEARCHING: a common problem in computer science involves storing and maintaining large amounts of data, and then searching the data for particular values

- data storage and retrieval are key to many industry applications
- search algorithms are necessary to storing and retrieving data efficiently

- e.g., consider searching a large payroll database for a particular record
  - if the computer selected entries at random, there is no assurance that the particular record will be found
  - even if the record is found, it is likely to take a large amount of time
  - a systematic approach assures that a given record will be found, and that it will be found more efficiently

there are two commonly used algorithms for searching a list of items

- sequential search – general purpose, but relatively slow
- binary search – restricted use, but fast

Sequential Search

sequential search is an algorithm that involves examining each list item in sequential order until the desired item is found

<u>sequential search for finding an item in a list</u>

1. start at the beginning of the list
2. for each item in the list
   1. examine the item - if that item is the one you are seeking, then you are done
   2. if it is not the item you are seeking, then go on to the next item in the list

if you reach the end of the list and have not found the item, then it was not in the list

sequential search guarantees that you will find the item if it is in the list

- but it is not very practical for very large databases
- *worst case:* you may have to look at every entry in the list

Binary Search

binary search involves continually cutting the desired search list in half until the item is found

- the algorithm is only applicable if the list is ordered
  - e.g., a list of numbers in increasing order
  - e.g., a list of words in alphabetical order

<u>binary search for finding an item in an ordered list</u>

1. initially, the potential range in which the item could occur is the entire list
2. as long as items remain in the potential range and the desired item has not been found, repeatedly
   i. examine at the middle entry in the potential range
   ii. if the middle entry is the item you are looking for, then you are done
   iii. if the middle entry is greater than the desired item, then reduce the potential range to those entries left of the middle
   iv. if the middle entry is less than the desired item, then reduce the potential range to those entries right of the middle

by repeatedly cutting the potential range in half, binary search can home in on the value very quickly

Binary Search Example

suppose you have a sorted list of state names, and want to find *MD*

1. start by examining the middle entry (*ND*)

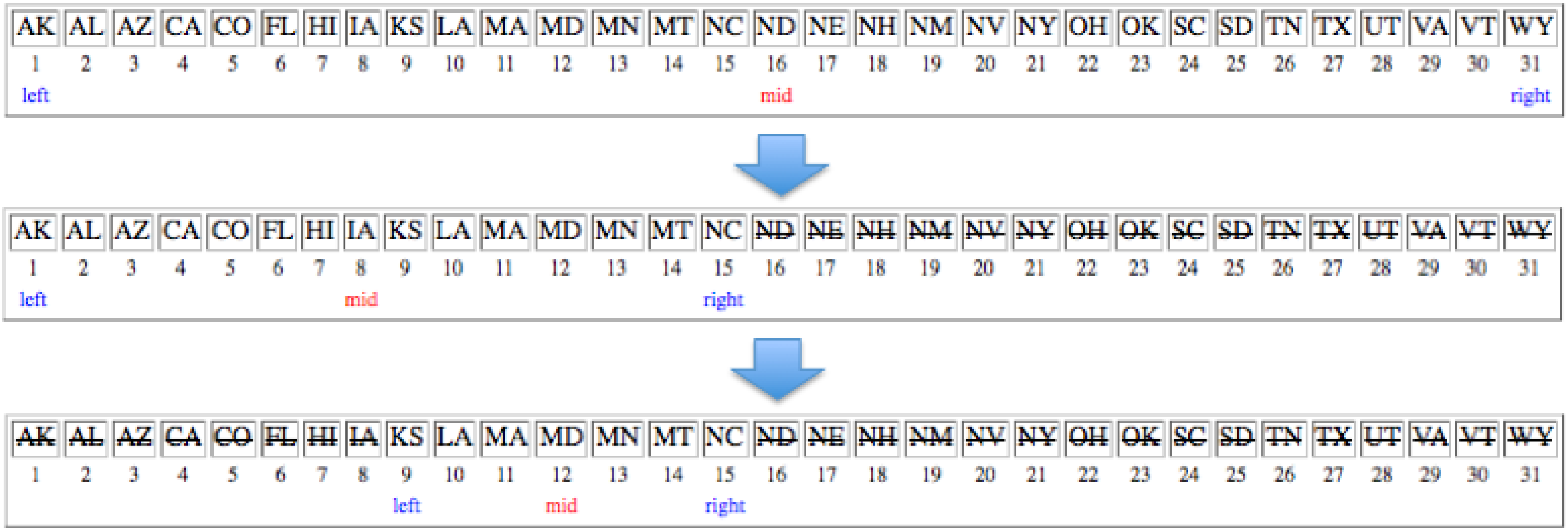
   since *ND* comes after *MD* alphabetically, can eliminate it and all entries that appear to the right

2. next, examine the middle of the remaining entries (*IA*)

   since *IA* comes before *MD* alphabetically, can eliminate it and all entries that appear to the left

3. next, examine the middle of the remaining entries (*MD*)

   the desired entry is found



AK	AL	AZ	CA	CO	FL	HI	IA	KS	LA	MA	MD	MN	MT	NC	ND	NE	NH	NM	NV	NY	OH	OK	SC	SD	TN	TX	UT	VA	VT	WY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
left															mid															right

AK	AL	AZ	CA	CO	FL	HI	IA	KS	LA	MA	MD	MN	MT	NC	~~ND~~	~~NE~~	~~NH~~	~~NM~~	~~NV~~	~~NY~~	~~OH~~	~~OK~~	~~SC~~	~~SD~~	~~TN~~	~~TX~~	~~UT~~	~~VA~~	~~VT~~	~~WY~~
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
left							mid							right																

~~AK~~	~~AL~~	~~AZ~~	~~CA~~	~~CO~~	~~FL~~	~~HI~~	~~IA~~	KS	LA	MA	MD	MN	MT	NC	~~ND~~	~~NE~~	~~NH~~	~~NM~~	~~NV~~	~~NY~~	~~OH~~	~~OK~~	~~SC~~	~~SD~~	~~TN~~	~~TX~~	~~UT~~	~~VA~~	~~VT~~	~~WY~~
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
								left			mid			right																

Search Analysis

sequential search

- in the worst case, the item you are looking for is in the last spot in the list (or not in the list at all)
  - as a result, you will have to inspect and compare every entry in the list
- the amount of work required is proportional to the list size

  → sequential search is an O(N) algorithm

binary search

- in the worst case, you will have to keep halving the list until it gets down to a single entry
  - each time you inspect/compare an entry, you rule out roughly half the remaining entries
- the amount of work required is proportional to the logarithm of the list size

  → binary search is an O(log N) algorithm

imagine searching a phone book of the United States (330 million people)

- sequential search requires at most 330 million inspections/comparisons
- binary search requires at most $\lceil \log_2(330{,}000{,}000) \rceil = 29$ inspections/comparisons

Another Algorithm Example

Newton's Algorithm for finding the square root of N

1. start with an initial approximation of 1
2. as long as the approximation isn't close enough, repeatedly
   1. refine the approximation using the formula:

      newApproximation = (oldApproximation + N/oldApproximation)/2

example: finding the square root of 1024

```
1    →  512.5
     → 257.2490243902439
     → 130.61480157022683
     →  69.22732405448894
     →  42.00958563100827
     →  33.19248741685438
     →  32.02142090500024
     →  32.0000071648159
     →  32.0000000000008
     →  32
```

algorithm analysis:

- Newton's Algorithm does converge on the square root because each successive approximation is closer than the previous one
  - however, since the square root might be a non-terminating fraction it is difficult to define the exact number of steps for convergence
- in general, the difference between the given approximation and the actual square root is roughly cut in half by each successive refinement

  → demonstrates O(log N) behavior

Algorithms and Programming

programming is all about designing and coding algorithms for solving problems

- the intended executor is the computer or a program executing on that computer
- instructions are written in programming languages which are more constrained and exact than human languages

the level of precision necessary to write programs can be frustrating to beginners

- but it is much easier than it was 50 years ago
- early computers (ENIAC) needed to be wired to perform computations
- with the advent of the von Neumann architecture, computers could be programmed instead of rewired
    - an algorithm could be coded as instructions, loaded into the memory of the computer, and executed

Evolution of Languages

the first programming languages were known as *machine languages*

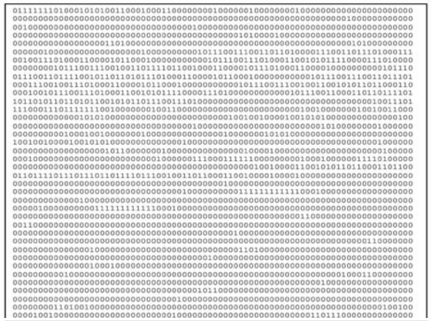
- consist of instructions that correspond directly to the hardware operations of a particular machine
  - i.e., instructions deal directly with the computer's physical components including main memory and CPU registers
  - very low level of abstraction
- machine language instructions are written in binary
  - programming in machine language is tedious and error prone

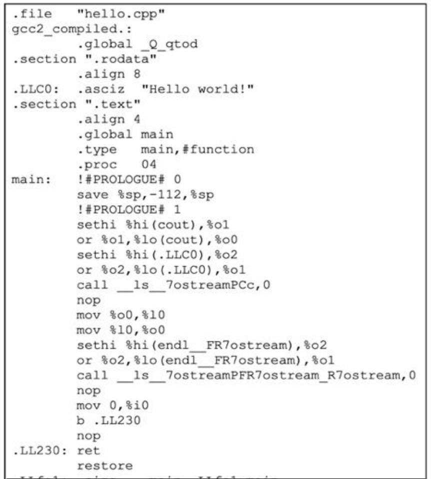
in early 1950s, *assembly languages* evolved from machine languages

- replaced binary codes with words like ADD, MOVE
  - easier to remember & debug, but still machine-specific
- a separate program called an *assembler* translated the assembly instructions into machine language

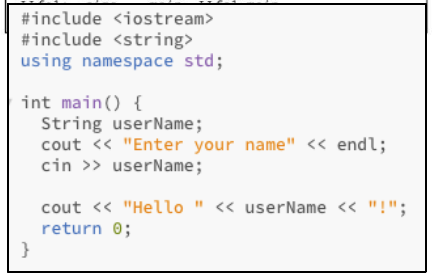
in the late 1950's, *high-level languages* were introduced

- they allow the programmer to write code closer to the way humans think (as opposed to mimicking hardware operations)
- more natural, plus machine-independent





```
.file   "hello.cpp"
gcc2_compiled.:
        .global _Q_qtod
.section ".rodata"
        .align 8
.LLC0:  .asciz  "Hello world!"
.section ".text"
        .align 4
        .global main
        .type   main,#function
        .proc   04
main:   !#PROLOGUE# 0
        save %sp,-112,%sp
        !#PROLOGUE# 1
        sethi %hi(cout),%o1
        or %o1,%lo(cout),%o0
        sethi %hi(.LLC0),%o2
        or %o2,%lo(.LLC0),%o1
        call __ls__7ostreamPCc,0
        nop
        mov %o0,%l0
        mov %l0,%o0
        sethi %hi(endl__FR7ostream),%o2
        or %o2,%lo(endl__FR7ostream),%o1
        call __ls__7ostreamPFR7ostream_R7ostream,0
        nop
        mov 0,%i0
        b .LL230
        nop
.LL230: ret
        restore
```



```
#include <iostream>
#include <string>
using namespace std;

int main() {
  String userName;
  cout << "Enter your name" << endl;
  cin >> userName;

  cout << "Hello " << userName << "!";
  return 0;
}
```

Program Translation

using a high-level language, the programmer is able to reason at a high-level of abstraction

- but programs must still be translated into machine language that the computer hardware can understand/execute

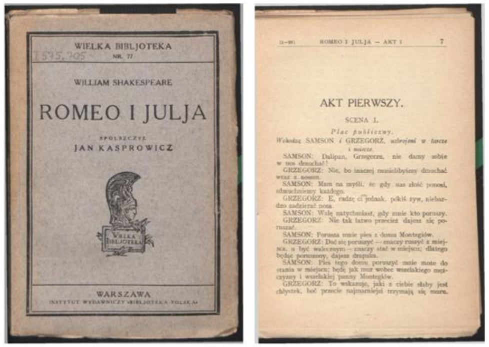
real-world analogy: translating a speech from one language to another

an *interpreter* can be used provide a real-time translation

- the interpreter hears a phrase, translates, and immediately speaks the translation
- ADVANTAGE: the translation is immediate
- DISADVANTAGE: if you want to hear the speech again, must interpret all over again

a *translator* (or *compiler*) translates the entire speech offline

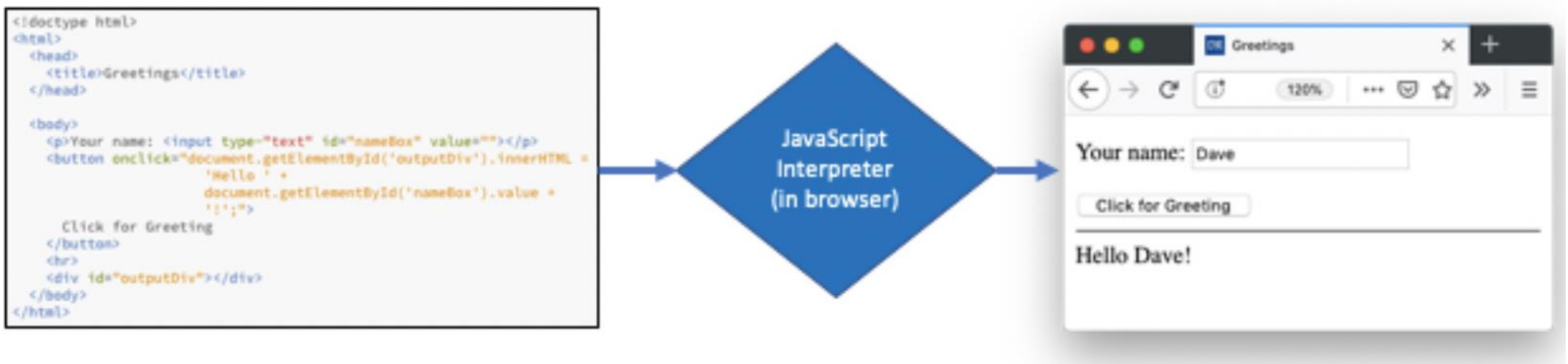
- the translator takes a copy of the speech, returns when the entire speech is translated
- ADVANTAGE: once translated, it can be read over and over very quickly
- DISADVANTAGE: must wait for the entire speech to be translated



Interpreters

for program translation, the interpretation approach relies on a program known as an *interpreter* to translate and execute high-level statements

- the interpreter reads one high-level statement at a time, immediately translating and executing the statement before processing the next one
- particularly useful for dynamic, interactive applications (e.g., Web pages)
- each execution requires translating again, can be slow
- JavaScript is interpreted



```
<!doctype html>
<html>
  <head>
    <title>Greetings</title>
  </head>

  <body>
    <p>Your name: <input type="text" id="nameBox" value=""></p>
    <button onclick="document.getElementById('outputDiv').innerHTML =
                     'Hello ' +
                     document.getElementById('nameBox').value +
                     '!';">
      Click for Greeting
    </button>
    <hr>
    <div id="outputDiv"></div>
  </body>
</html>
```

Your name: Dave

Hello Dave!

1. Start with a JavaScript program (embedded in hello.html).

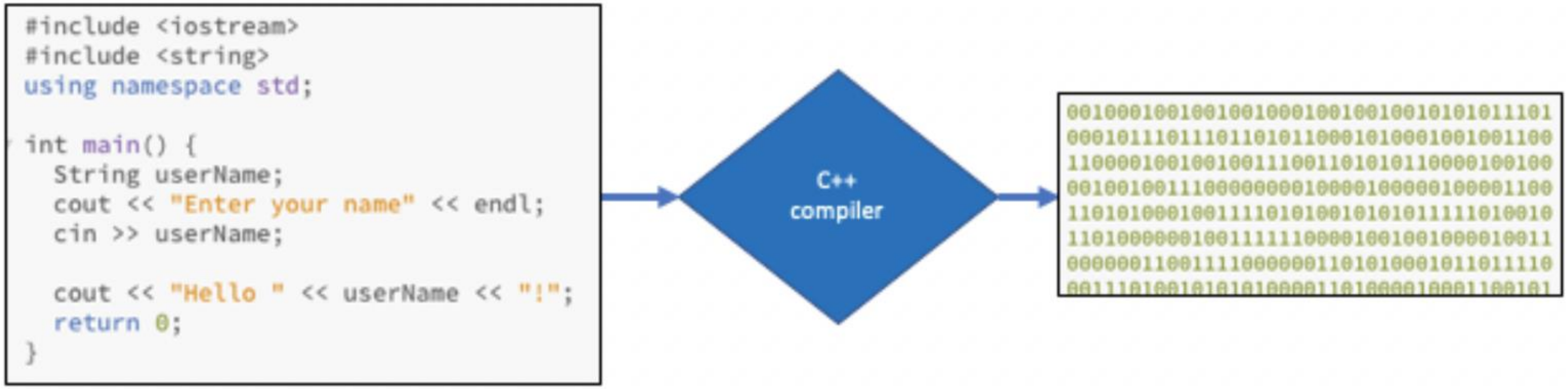
2. The interpreter in the Web browser reads each statement, translates, and executes within the browser.

3. The user sees the result of each step displayed in the page.

Compilers

the compilation approach relies on a program known as a *compiler* to translate the entire high-level language program into its equivalent machine-language instructions

- the resulting machine-language program can be executed directly on the computer, very fast
- used in large software applications when speed is of the utmost importance
- but must compile the entire program before execution, so initial delay
- C, C++, Java are compiled*



```
#include <iostream>
#include <string>
using namespace std;

int main() {
   String userName;
   cout << "Enter your name" << endl;
   cin >> userName;

   cout << "Hello " << userName << "!";
   return 0;
}
```

```
0010001001001001000100100100101010111101
0001011101110110101100010100010010011000
1100000100100100111001101010110000100100
0010010011100000000010000100000100001100
1101010001001111010100101010111110100010
1101000000100111111000010010010000010011
0000001100111110000000110101000101101110
0011101001010101000001101000001000110010101
```

1. Start with a C++ program (stored as hello.cpp).

2. A compiler takes the program, translates, and produces a machine language version (hello.exe).

3. The user may execute the machine language translation directly (and repeatedly).

Abstraction

abstraction is the process of ignoring minutiae and focusing on the big picture

- in modern life, we are constantly confronted with complexity
- we don't necessarily know how it works, but we know how to use it
  e.g., how does a TV work? a car? a computer?

we survive in the face of complexity by abstracting away details

- to use a TV/car/computer, it's not important to understand the inner workings
- we ignore unimportant details and focus on those features relevant to using it
- e.g., TV has power switch, volume control, channel changer, …

JavaScript functions (like `Math.sqrt`) provide computational abstraction

- a function encapsulates some computation & hides the details
- the user only needs to know how to call the function, not how it works
- simple user-defined functions similarly provide abstractions that simplify the page

User-defined Functions

functions simplify the programmer's task

- minimize the amount of detail the developer must remember
  e.g., to calculate a square root, only need to remember the name `Math.sqrt`
- minimize the size and complexity of code
  e.g., simple user-defined functions in the head simplify buttons in the body

the general form of user-defined functions:

```
function FUNCTION_NAME(PARAMETER1, PARAMETER2, …) {
  STATEMENTS_TO_PERFORM_THE_DESIRED_COMPUTATION

  return OUTPUT_VALUE;
}
```

- parameters are variables that correspond to the inputs when the function is called
  will have as many parameters as there are inputs to the function (could be 0)

- a return statement specifies the output value for that function
  optional, as some functions don't return a value (e.g., write a message in page)

Simple example

consider the following function for converting distances:

```
function InchesToCentimeters(inches) {
    cm = inches * 2.54;

    return cm;
}
```

- the function has one parameter, named inches
- when the function is called, an input value must be specified in the parentheses

```
InchesToCentimeters(10)
```

- the input value from the call is assigned to the parameter variable
- that variable can then be used in calculations (here, cm = 2.54*10 = 25.4)
- when a return statement is reached, the expression is evaluated and returned

a function call can appear in any expression

- the return value is substituted for the call when evaluating

```
outputP.innerHTML = InchesToCentimeters(10);
```

displays 25.4 in outputP

Simple example (cont.)

as with any variable, parameter names should convey their purpose:

- `inches`, `distanceInInches`, `lengthInInches`, …

convention used throughout the book:

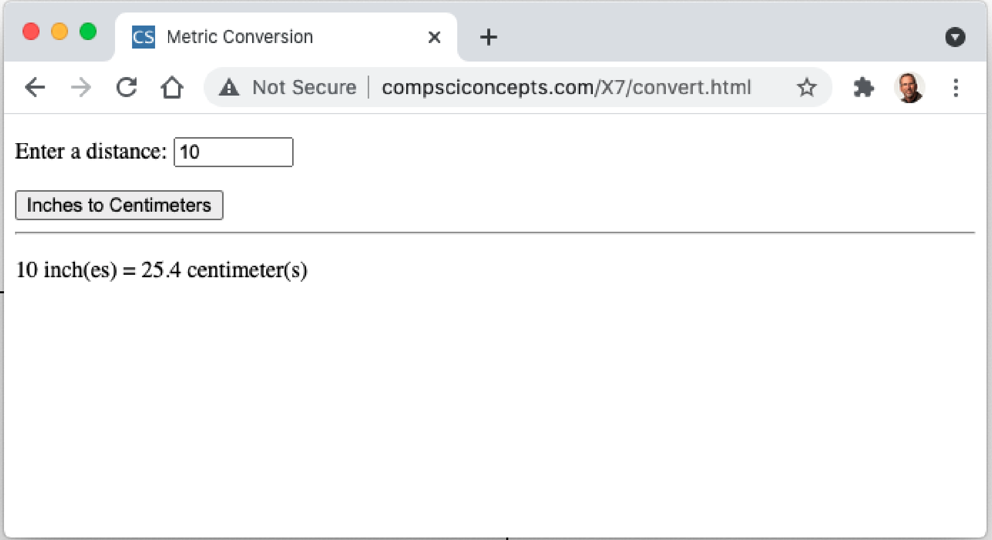
- variable/parameter names start with a lowercase letter
- function names start with an uppercase letter
- if a variable name consists of multiple words, use internal capitalization

since a return statement can specify an expression, we could equivalently define the function as:

```
function InchesToCentimeters(distanceInInches) {
    return distanceInInches * 2.54;
}
```

Conversion Page

this shows that a function can contain a call to another function



Enter a distance: 10

Inches to Centimeters

10 inch(es) = 25.4 centimeter(s)

```
<!doctype html>
<!-- convert.html                          Dave Reed
<!-- This page converts from inches to centimeters..
<!-- ===========================================

<html>
 <head>
   <title> Metric Conversion  </title>
   <script>
     function InchesToCentimeters(distanceInInches) {
         return distanceInInches * 2.54;
     }

     function ShowInToCm() {
       inches = distBox.valueAsNumber;
       centimeters = InchesToCentimeters(inches);
       outputP.innerHTML =
           inches + ' inch(es) = ' + centimeters + ' centimeter(s)';
     }
   </script>
 </head>

 <body>
   <p>Enter a distance: <input type="number" id="distBox" min=0 max=99999.99 value=1></p>
   <button onclick="ShowInToCm();">Inches to Centimeters</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```

note 2 function definitions in head

still need parameterless function connected to the button

Pick-4 Revisited

InchesToCentimeters is easy to understand, but not very motivational

for a better example, consider the Pick-4 page from Chapter X5

- used Math.random to generate random lottery balls in the range 1..numBalls

```
pick1 = Math.floor(numBalls*Math.random() + 1);
pick2 = Math.floor(numBalls*Math.random() + 1);
pick3 = Math.floor(numBalls*Math.random() + 1);
pick4 = Math.floor(numBalls*Math.random() + 1);
```

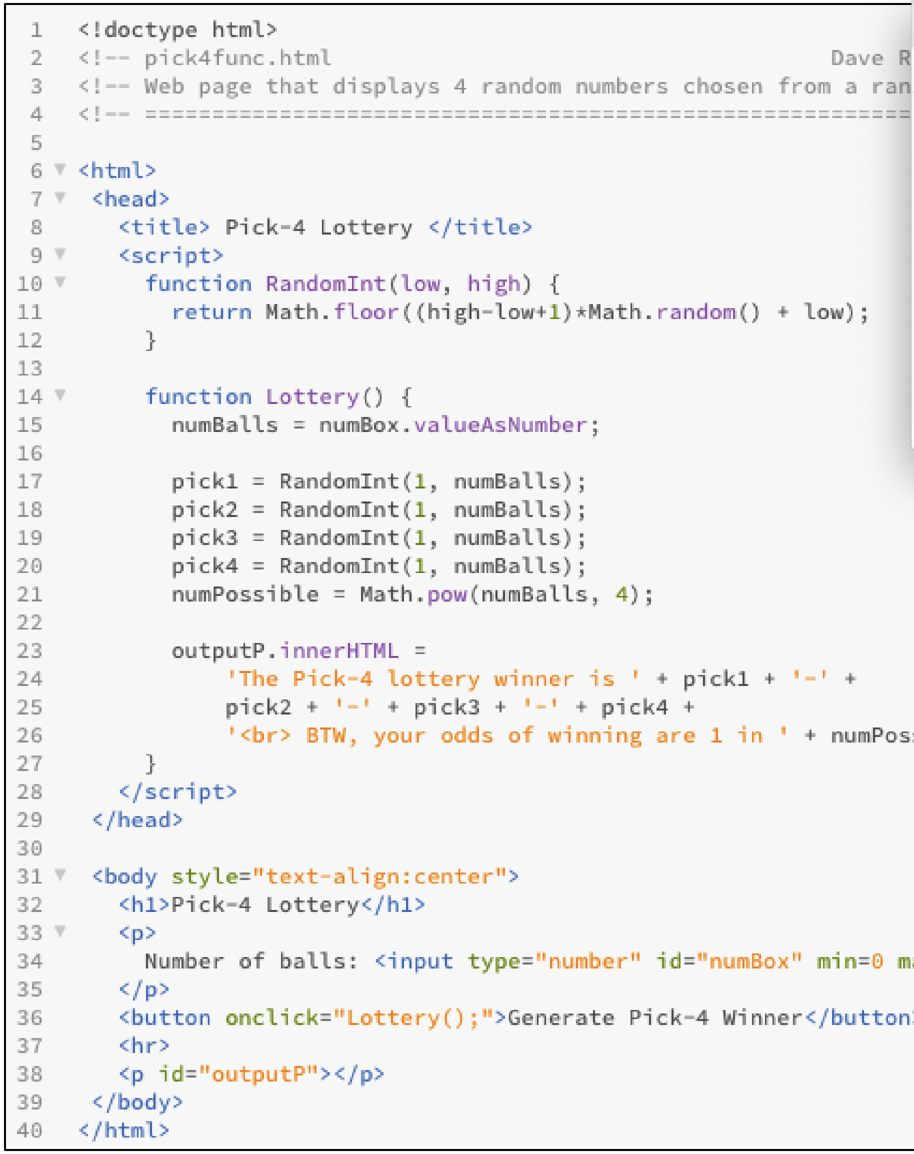
- tricky, messy, difficult to modify (e.g., suppose we learned balls started at 0)

better solution: capture the tricky expression in a function

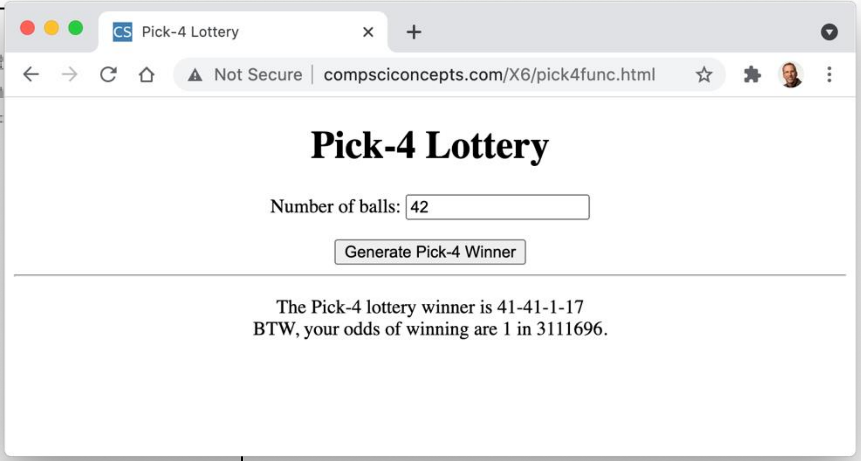
```
function RandomInt(low, high) {
  return Math.floor(Math.random()*(high-low+1)) + low;
}
```

- once defined & tested, can be used anywhere a random integer is needed

Pick-4 with Function



```
<!doctype html>
<!-- pick4func.html                                    Dave R
<!-- Web page that displays 4 random numbers chosen from a ran
<!-- ==========================================================

<html>
 <head>
   <title> Pick-4 Lottery </title>
   <script>
     function RandomInt(low, high) {
       return Math.floor((high-low+1)*Math.random() + low);
     }

     function Lottery() {
       numBalls = numBox.valueAsNumber;

       pick1 = RandomInt(1, numBalls);
       pick2 = RandomInt(1, numBalls);
       pick3 = RandomInt(1, numBalls);
       pick4 = RandomInt(1, numBalls);
       numPossible = Math.pow(numBalls, 4);

       outputP.innerHTML =
           'The Pick-4 lottery winner is ' + pick1 + '-' +
           pick2 + '-' + pick3 + '-' + pick4 +
           '<br> BTW, your odds of winning are 1 in ' + numPossible + '.';
     }
   </script>
 </head>

 <body style="text-align:center">
   <h1>Pick-4 Lottery</h1>
   <p>
     Number of balls: <input type="number" id="numBox" min=0 max=999 value=42>
   </p>
   <button onclick="Lottery();">Generate Pick-4 Winner</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```



Pick-4 Lottery

Number of balls: 42

The Pick-4 lottery winner is 41-41-1-17
BTW, your odds of winning are 1 in 3111696.

`RandomInt` function makes the code much easier to understand & modify

e.g., for balls starting at 0:

```
pick1 = RandomInt(0, numBalls-1);
pick2 = RandomInt(0, numBalls-1);
pick3 = RandomInt(0, numBalls-1);
pick4 = RandomInt(0, numBalls-1);
```

More Examples

```
function Greeting(firstName, lastName) {
  return 'Hello ' + firstName + ' ' + lastName +
         '. May I call you ' + firstName + '?';
}
```

parameters & return values can be strings

```
function Distance(x1, y1, x2, y2) {
  term1 = Math.pow(x1-x2, 2);
  term2 = Math.pow(y1-y2, 2));

  return Math.sqrt(term1 + term2);
}
```

function for calculating point distance (Ch. X5)

```
function CompoundInterest(amount, rate, years) {
  return amount * Math.pow(1+rate/100, years);
}
```

function for calculating compound interest (Ch. X5)

```
function Sqrt(n) {
  approx = 1;
  while (Math.abs(approx*approx - n) > 0.000001) {
      approx = (approx + n/approx)/2;
  }

  return approx;
}
```

function for calculating square root (using Newton's algorithm from Ch. C6)

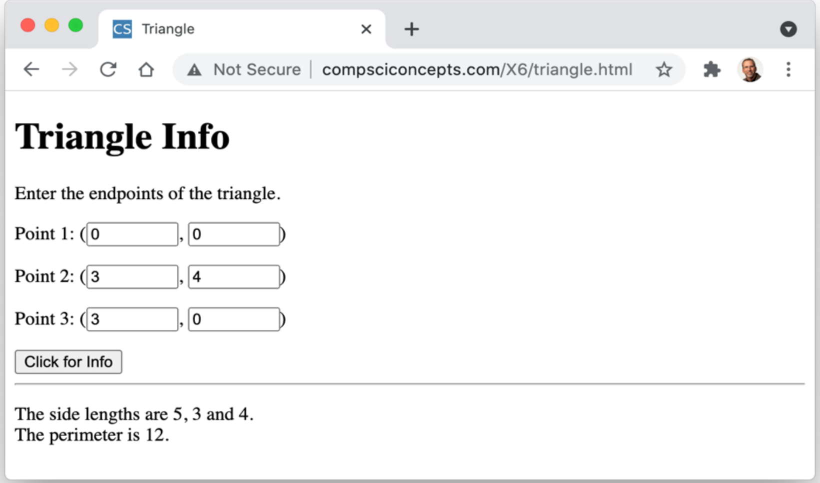
Triangle Page

recall `distance.html` page from Ch. X5

- can simplify by adding Distance function

functions are especially useful if the calculation is performed multiple times

- here, need to calculate the lengths of three triangle sides
- define the complex distance calculation once in the function
- can then call the function repeatedly



Triangle Info

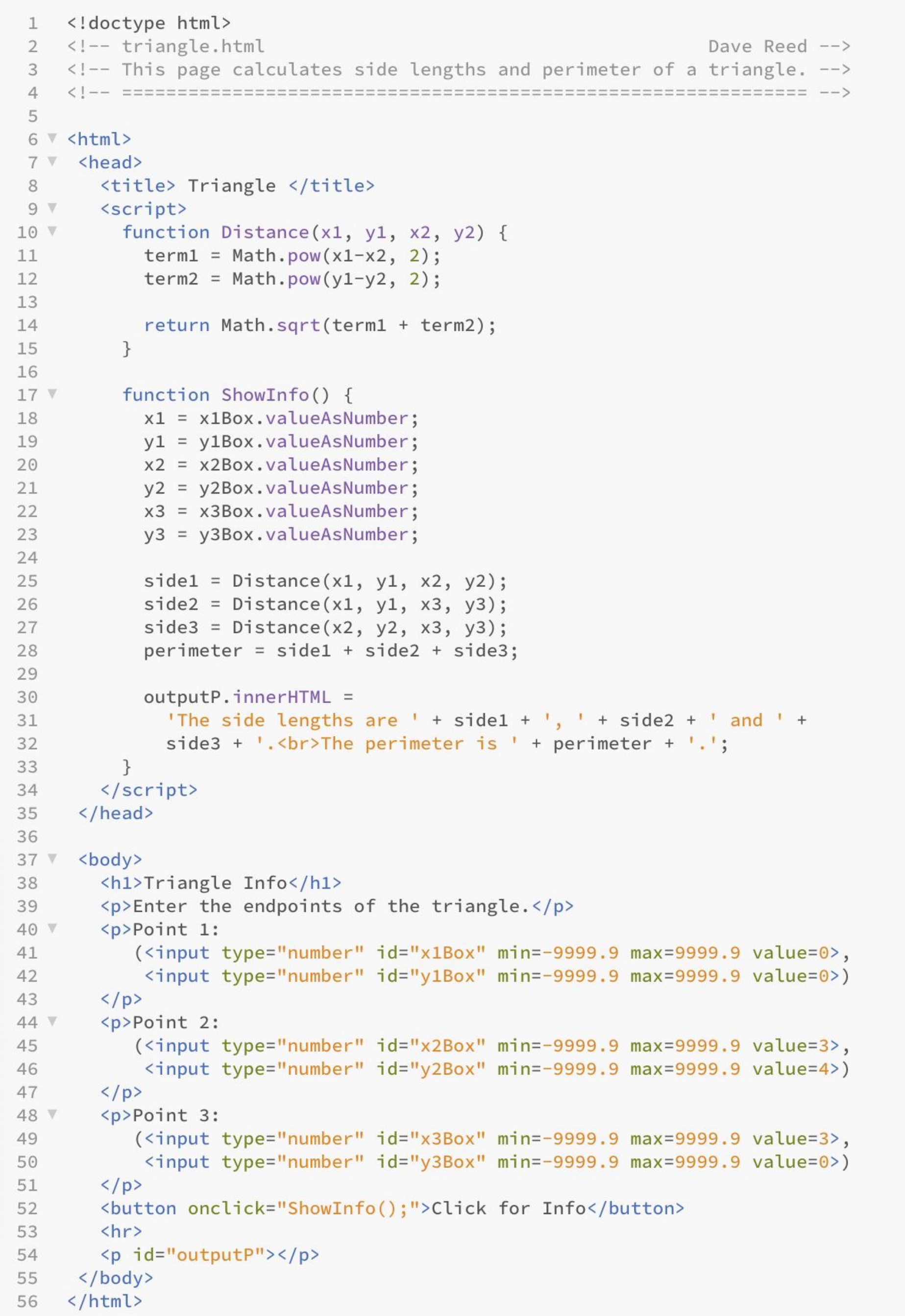
Enter the endpoints of the triangle.

Point 1: (0, 0)

Point 2: (3, 4)

Point 3: (3, 0)

The side lengths are 5, 3 and 4.
The perimeter is 12.



```html
<!doctype html>
<!-- triangle.html                                             Dave Reed -->
<!-- This page calculates side lengths and perimeter of a triangle. -->
<!-- ============================================================== -->

<html>
 <head>
   <title> Triangle </title>
   <script>
     function Distance(x1, y1, x2, y2) {
       term1 = Math.pow(x1-x2, 2);
       term2 = Math.pow(y1-y2, 2);

       return Math.sqrt(term1 + term2);
     }

     function ShowInfo() {
       x1 = x1Box.valueAsNumber;
       y1 = y1Box.valueAsNumber;
       x2 = x2Box.valueAsNumber;
       y2 = y2Box.valueAsNumber;
       x3 = x3Box.valueAsNumber;
       y3 = y3Box.valueAsNumber;

       side1 = Distance(x1, y1, x2, y2);
       side2 = Distance(x1, y1, x3, y3);
       side3 = Distance(x2, y2, x3, y3);
       perimeter = side1 + side2 + side3;

       outputP.innerHTML =
         'The side lengths are ' + side1 + ', ' + side2 + ' and ' +
         side3 + '.<br>The perimeter is ' + perimeter + '.';
     }
   </script>
 </head>

 <body>
   <h1>Triangle Info</h1>
   <p>Enter the endpoints of the triangle.</p>
   <p>Point 1:
      (<input type="number" id="x1Box" min=-9999.9 max=9999.9 value=0>,
       <input type="number" id="y1Box" min=-9999.9 max=9999.9 value=0>)
   </p>
   <p>Point 2:
      (<input type="number" id="x2Box" min=-9999.9 max=9999.9 value=3>,
       <input type="number" id="y2Box" min=-9999.9 max=9999.9 value=4>)
   </p>
   <p>Point 3:
      (<input type="number" id="x3Box" min=-9999.9 max=9999.9 value=3>,
       <input type="number" id="y3Box" min=-9999.9 max=9999.9 value=0>)
   </p>
   <button onclick="ShowInfo();">Click for Info</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```

Function libraries

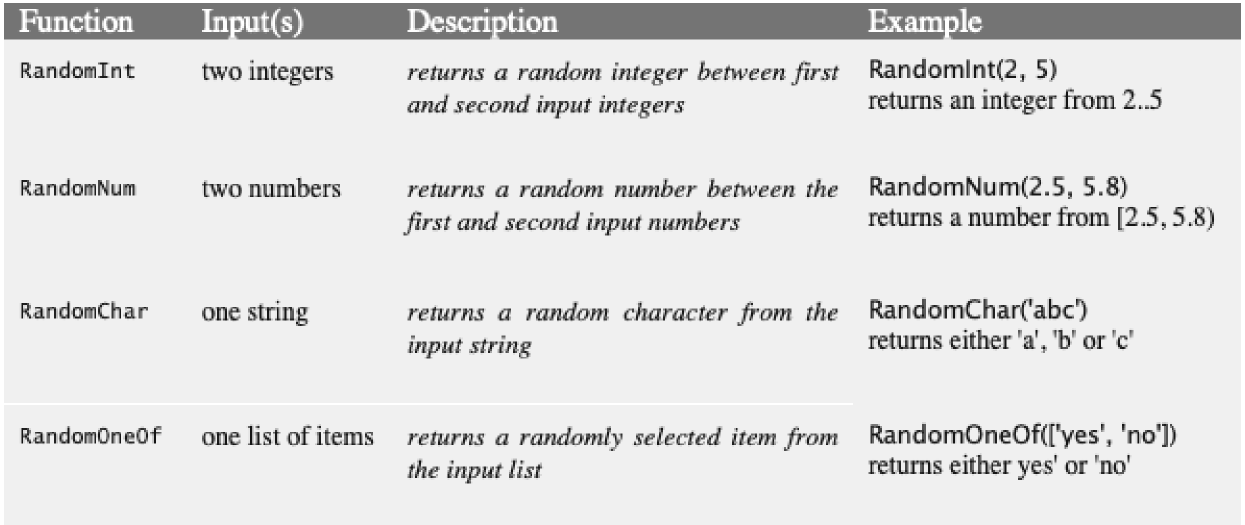
functions such as `RandomInt` can be added to head of a page

- tedious if functions are to be used in many pages
- involves creating lots of copies that all must be maintained for consistency

the alternative for general purpose functions is to place them in a library file

- a *library file* is a separate text file that contains the definitions of one or more JavaScript functions
- by convention, function library files end in `.js` since they contain JavaScript code

e.g., `http://compsciconcepts.com/random.js`



Function	Input(s)	Description	Example
`RandomInt`	two integers	*returns a random integer between first and second input integers*	RandomInt(2, 5) returns an integer from 2..5
`RandomNum`	two numbers	*returns a random number between the first and second input numbers*	RandomNum(2.5, 5.8) returns a number from [2.5, 5.8)
`RandomChar`	one string	*returns a random character from the input string*	RandomChar('abc') returns either 'a', 'b' or 'c'
`RandomOneOf`	one list of items	*returns a randomly selected item from the input list*	RandomOneOf(['yes', 'no']) returns either yes' or 'no'

Function libraries

once a function library file has been created, it can be loaded into any page by adding a `script` element whose `src` is that file

```
<script src="LIBRARY FILENAME"><script>
```

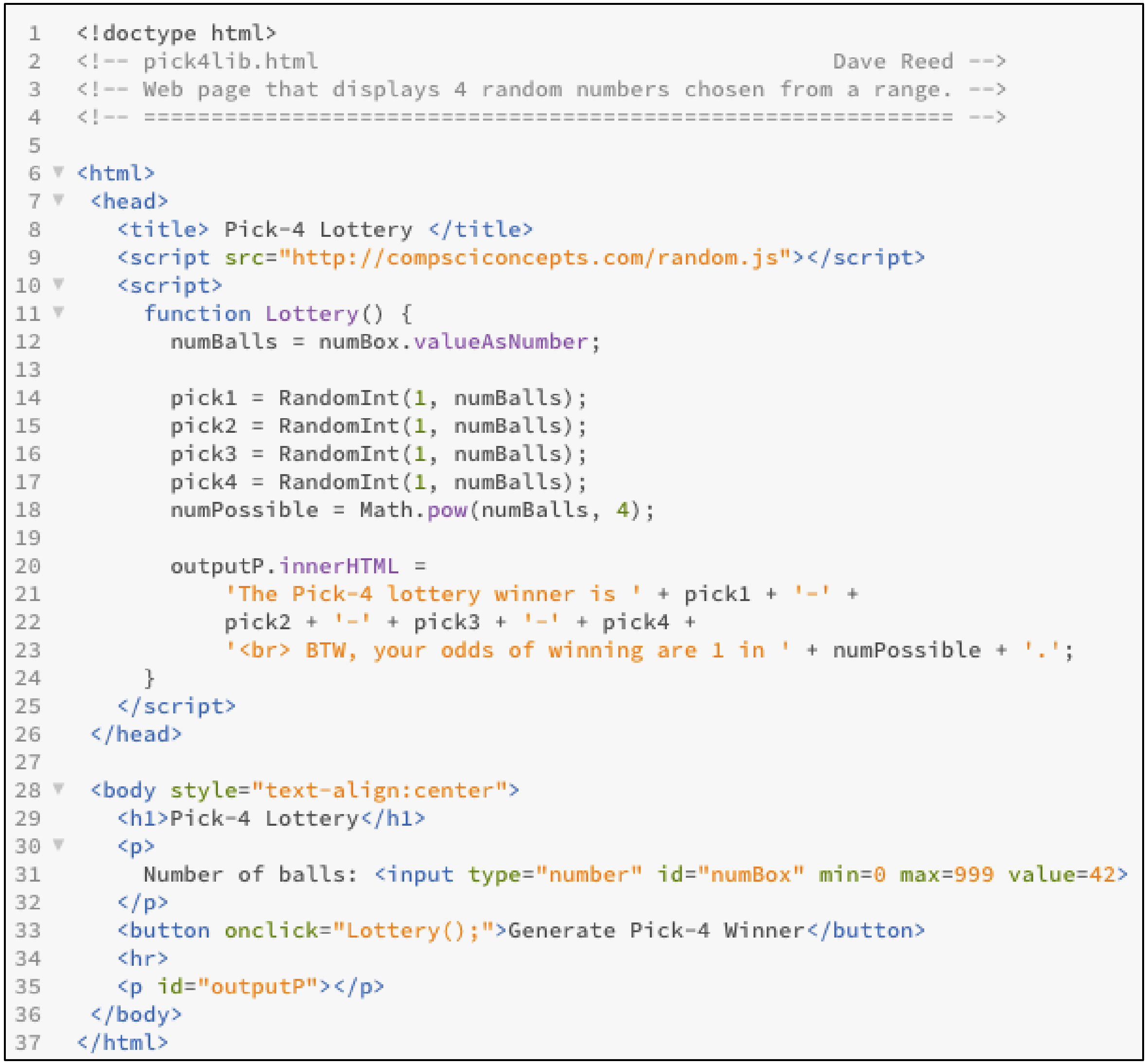
- this new `script` elements is added to the head of the page
- note that no actual code appears in between the `script` tags

  the code from the library is inserted the `script` element when the page loads

advantages of library files:

- avoid duplication (only one copy of the function definition)
- easier to reuse functions (simply load the library file into any page)
- easier to modify functions (a single change to the library file automatically affects all pages that load the library

Pick-4 Yet Again



```
<!doctype html>
<!-- pick4lib.html                                      Dave Reed -->
<!-- Web page that displays 4 random numbers chosen from a range. -->
<!-- ============================================================ -->

<html>
 <head>
   <title> Pick-4 Lottery </title>
   <script src="http://compsciconcepts.com/random.js"></script>
   <script>
     function Lottery() {
       numBalls = numBox.valueAsNumber;

       pick1 = RandomInt(1, numBalls);
       pick2 = RandomInt(1, numBalls);
       pick3 = RandomInt(1, numBalls);
       pick4 = RandomInt(1, numBalls);
       numPossible = Math.pow(numBalls, 4);

       outputP.innerHTML =
           'The Pick-4 lottery winner is ' + pick1 + '-' +
           pick2 + '-' + pick3 + '-' + pick4 +
           '<br> BTW, your odds of winning are 1 in ' + numPossible + '.';
     }
   </script>
 </head>

 <body style="text-align:center">
   <h1>Pick-4 Lottery</h1>
   <p>
     Number of balls: <input type="number" id="numBox" min=0 max=999 value=42>
   </p>
   <button onclick="Lottery();">Generate Pick-4 Winner</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```

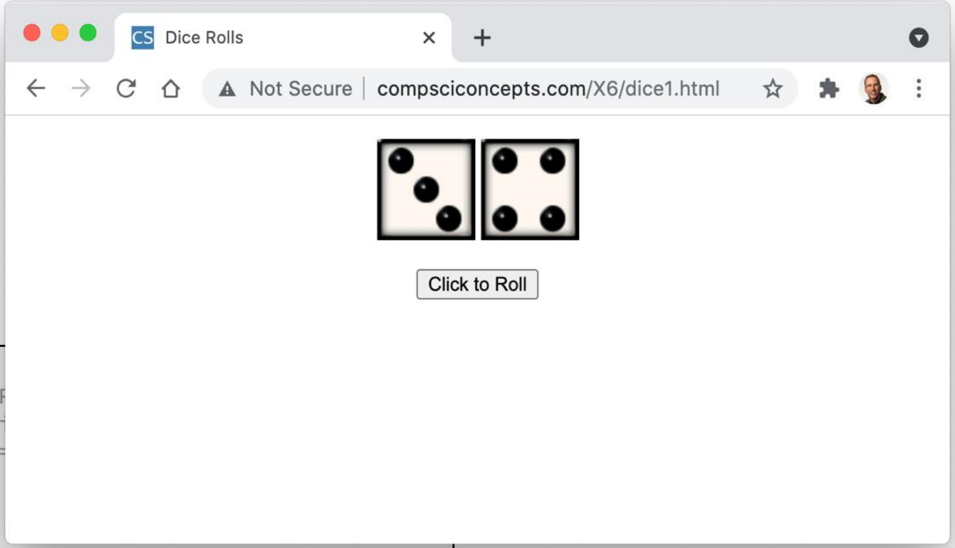
note: there is one `script` element for loading the library

- this makes the `RandomInt` function accessible within the page

there is separate `script` element for defining the page-specific function

- this function can call the library function that has been loaded

Dice Example (v. 1)

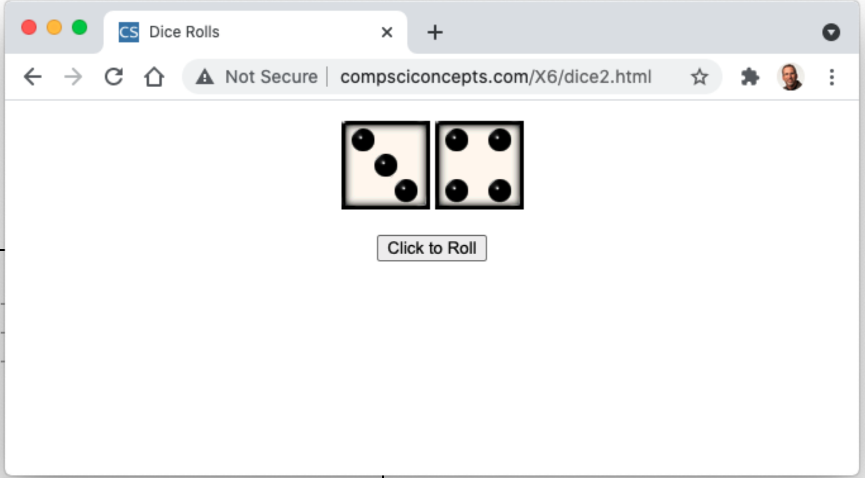


```
<!doctype html>
<!-- dice1.html                                 Dave
<!-- This page simulates and displays the roll of two d
<!-- ===================================================

<html>
 <head>
   <title> Dice Rolls </title>
   <script src="http://compsciconcepts.com/random.js"></script>
   <script>
     function Roll() {
       roll1 = RandomInt(1, 6);
       roll2 = RandomInt(1, 6);

       die1Img.src = 'http://compsciconcepts.com/Images/die' + roll1 + '.gif';
       die2Img.src = 'http://compsciconcepts.com/Images/die' + roll2 + '.gif';
     }
   </script>
 </head>

 <body style="text-align:center">
   <p>
     <img id="die1Img" alt="die image" src="http://compsciconcepts.com/Images/die3.gif">
     <img id="die2Img" alt="die image" src="http://compsciconcepts.com/Images/die4.gif">
   </p>
   <button onclick="Roll();">Click to Roll</button>
 </body>
</html>
```

this page uses the `RandomInt` function from `random.js` to pick the random die images

note: die images are named `die1.gif`, `die2.gif`, …, `die6.gif`

Dice Example (v. 2)



```
<!doctype html>
<!-- dice2.html                                 Dave Reed --
<!-- This page simulates and displays the roll of two dice. --
<!-- ====================================================== --

<html>
 <head>
   <title> Dice Rolls </title>
   <script src="http://compsciconcepts.com/random.js"></script>
   <script>
     function Roll() {
       roll1 = RandomOneOf(['die1.gif', 'die2.gif', 'die3.gif',
                            'die4.gif', 'die5.gif', 'die6.gif']);
       roll2 = RandomOneOf(['die1.gif', 'die2.gif', 'die3.gif',
                            'die4.gif', 'die5.gif', 'die6.gif']);

       die1Img.src = 'http://compsciconcepts.com/Images/' + roll1;
       die2Img.src = 'http://compsciconcepts.com/Images/' + roll2;
     }
   </script>
 </head>

 <body style="text-align:center">
   <p>
     <img id="die1Img" alt="die image" src="http://compsciconcepts.com/Images/die3.gif">
     <img id="die2Img" alt="die image" src="http://compsciconcepts.com/Images/die4.gif">
   </p>
   <button onclick="Roll();">Click to Roll</button>
 </body>
</html>
```

alternatively, could use the RandomOneOf function from the random.js library to pick the die images

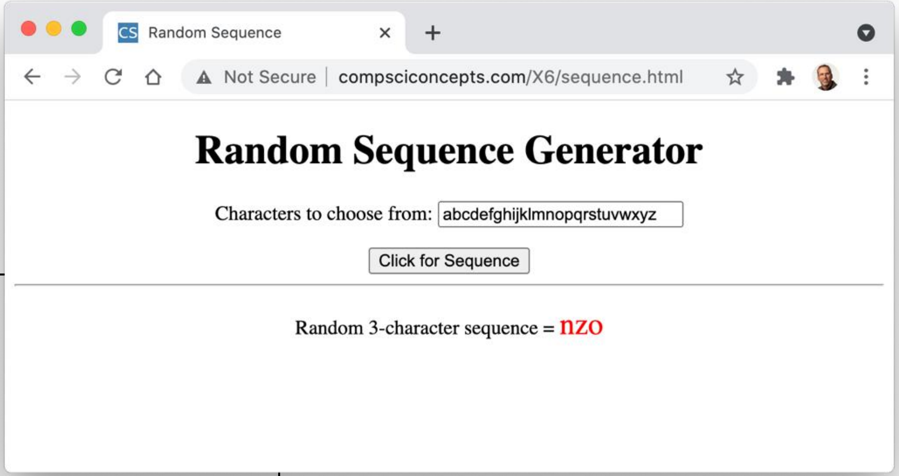
note: function input must be a list of options, contained in [ ] and separated by commas

Sequences Example

consider the task of generating random character sequences

e.g., for creating secure passwords

```
<!doctype html>
<!-- sequence.html                              Dave Reed -->
<!-- This page generates random 3-character sequences. -->
<!-- ================================================= -->

<html>
 <head>
   <title> Random Sequence </title>
   <script src="http://compsciconcepts.com/random.js"></script>

   <script>
     function Sequence() {
       chars = charsBox.value;

       seq3 = RandomChar(chars) + RandomChar(chars) + RandomChar(chars);

       outputP.innerHTML = 'Random 3-character sequence = ' +
                           '<span style="color:red; font-size:150%">' +
                           seq3 + '</span>';
     }
   </script>
 </head>

 <body  style="text-align:center">
   <h1>Random Sequence Generator</h1>
   <p>Characters to choose from:
      <input type="text" id="charsBox" size=26 value="abcdefghijklmnopqrstuvwxyz">
   </p>
   <button onclick="Sequence();">Click for Sequence</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```



Random Sequence Generator

Characters to choose from: abcdefghijklmnopqrstuvwxyz

Click for Sequence

Random 3-character sequence = nzo

the user can specify the characters to choose from in the text box

uses the `RandomChar` function from `random.js` to pick 3 random characters & concatenate them

Designing Functions

functions do not add any computational power to the language

- a function definition simply encapsulates other statements

still, the capacity to define and use functions is key to solving complex problems, as well as to developing reusable code

- encapsulating repetitive tasks can shorten and simplify code
- provide units of computational abstraction – user can ignore details

when you write a general-purpose function that could be used in many pages

- create a library file and package that function with related functions
- these can then be loaded into pages that need those functions

when you define a function that is page-specific:

- define it in the head of that page (within a `script` element)
- can have more than one function defined in the same `script` element

Computer “Science”

some people argue that computer science is not a science in the same sense that biology and chemistry are

- the interdisciplinary nature of computer science has made it hard to classify

computer science is the study of *computation* (more than just machinery)

- it involves all aspects of problem solving, including
  - the design and analysis of algorithms
  - the formalization of algorithms as programs
  - the development of computational devices for executing programs
  - the theoretical study of the power and limitations of computing

whether this constitutes a "science" is a matter of interpretation

- certainly, computer science represents a rigorous approach to understanding complex phenomena and problem solving

Artificial Science

many activities carried out by computer scientists utilize the scientific method

- e.g., designing and implementing a large database system requires hypothesizing about its behavior under various conditioning, experimenting to test those hypotheses, analyzing the results, and possibly redesigning
- e.g., debugging a complex program requires forming hypotheses about where an error might be occurring, experimenting to test those hypotheses, analyzing the results, and fixing the bugs

the distinction between computer science and natural sciences like biology, chemistry, and physics is the type of systems being studied

- natural sciences study naturally occurring phenomena and attempt to extract underlying laws of nature
- computer science study human-made constructs: programs, computers, and computational modes

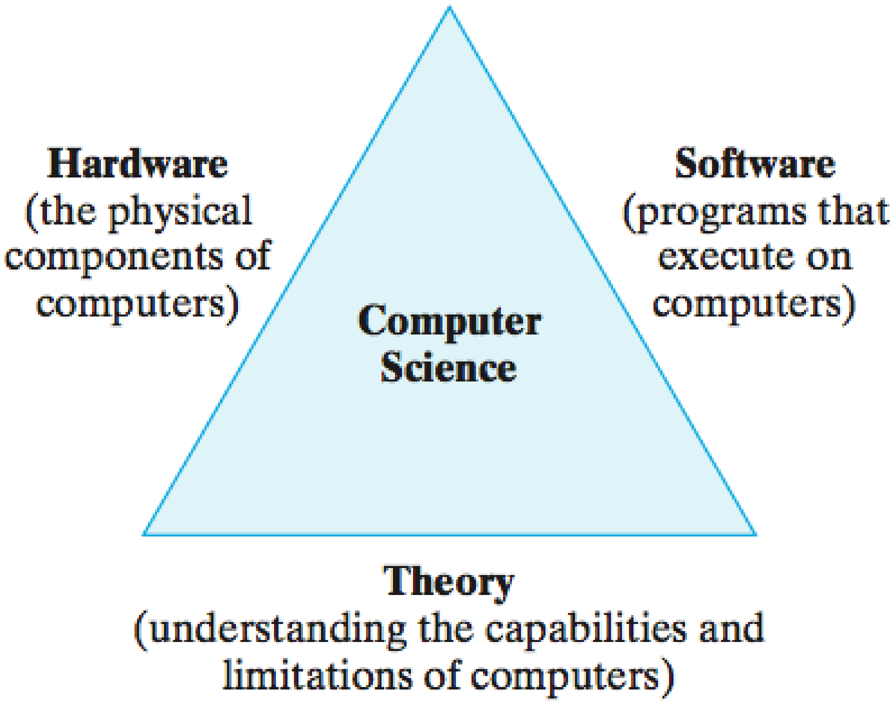
Herbert Simon coined the phrase "artificial science" to distinguish computer science from the natural sciences

- in Europe, computer science is commonly called "Informatics"

Computer Science Themes

since computation encompasses many different types of activities, computer science research is often difficult to classify

- three recurring themes define the discipline


Hardware
(the physical components of computers)
Software
(programs that execute on computers)
Computer Science
Theory
(understanding the capabilities and limitations of computers)

Hardware

hardware refers to the physical components of a computer and its supporting devices

most modern computers implement the von Neumann architecture

- CPU + memory + input/output devices

ongoing research seeks to improve hardware design and organization

- *circuit designers* create smaller, faster, more energy-efficient chips
- *microchip manufacturers* seek to miniaturize and streamline production
- *systems architects* research methods to increase *throughput* (the amount of work done in a given time period)
  - e.g., parallel processing – splitting the computation across multiple CPUs
  - e.g., networking – connecting computers to share information and work

Software

software refers to the programs that execute on computers

3 basic software categories

1. *systems software:* programs that directly control the execution of hardware components (e.g., operating systems)
2. *development software:* programs that are used as tools in the development of other programs (e.g. Microsoft.NET, Java SDK)
3. *applications software:* all other programs, which perform a wide variety of tasks (e.g., web browsers, word processors, games)

many careers in computer science are related to the design, development, testing, and maintenance of software

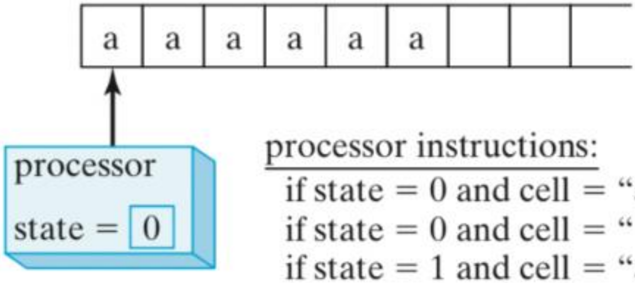
- *language designers* develop and extend programming languages for easier and more efficient solutions
- *programmers* design and code algorithms for execution on a computer
- *systems analysts* analyze program designs and manage development

Theory

theoretical computer scientists strive to understand the capabilities of algorithms and computers (deeply rooted in math and formal logic)

example: the *Turing machine* is an abstract computational machine invented by computer pioneer Alan Turing

- consists of: a tape on which characters can be written (I/O)
  a processor that can read/write on the tape, move left or right (CPU)
  space for storing the machine state - a number (memory)
- significance of the Turing machine
  - it is programmable (example below is programmed to distinguish between an even or odd number of a's on the tape)
  - provably as powerful as any modern computer, but simpler so provides a manageable tool for studying computation



processor instructions:

if state = 0 and cell = "a", move right and set state = 1.
if state = 0 and cell = " ", write " Y" and HALT.
if state = 1 and cell = "a", move right and set state = 0.
if state = 1 and cell = " ", write " N" and HALT.

Turing used this simpler model to prove there are problems that cannot be solved by any computer!

Subfields of Computer Science

computer science can be divided into subfields

- each subfield takes a unique approach to computation
- however the common themes of computer science (hardware, software, and theory) influence every subfield

4 highly visible subfields

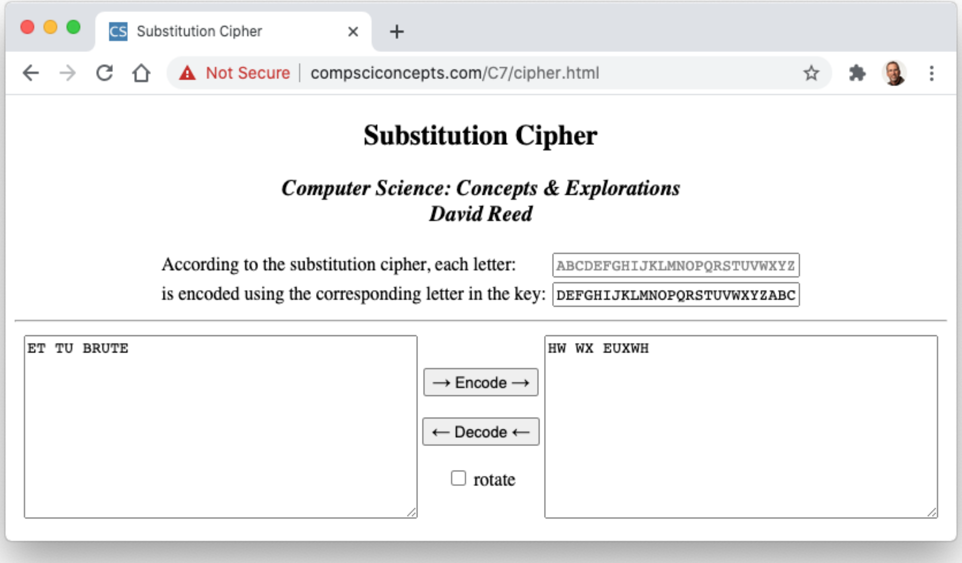
1. algorithms and data structures
2. architecture
3. software engineering
4. artificial intelligence and robotics

Algorithms and Data Structures

subfield that involves developing, analyzing, and implementing algorithms for solving problems

application: *encryption*

- encryption is the process of encoding a message so that it is decipherable only by its intended recipient
  - Caesar cipher: shift each letter three down in the alphabet
    e.g., ET TU BRUTE → HW WX EUXWH



Substitution Cipher

Computer Science: Concepts & Explorations
David Reed

According to the substitution cipher, each letter: ABCDEFGHIJKLMNOPQRSTUVWXYZ
is encoded using the corresponding letter in the key: DEFGHIJKLMNOPQRSTUVWXYZABC

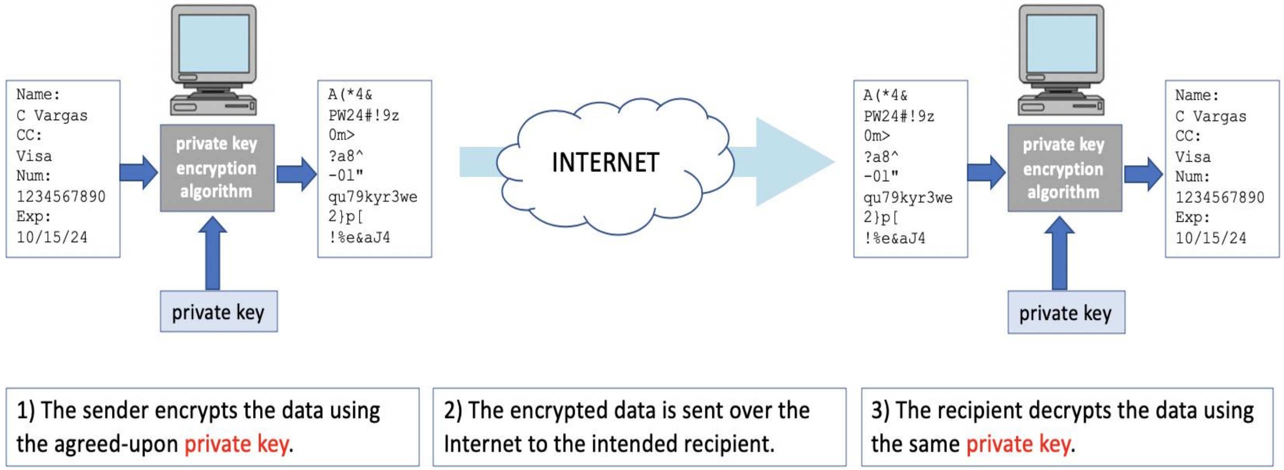
```
ET TU BRUTE
```

```
HW WX EUXWH
```

Private-key Encryption

Caesar cipher is an example of *private-key encryption*

- relies on the sender and the recipient sharing a secret key



1) The sender encrypts the data using the agreed-upon private key.

2) The encrypted data is sent over the Internet to the intended recipient.

3) The recipient decrypts the data using the same private key.

some modern encryption algorithms rely on private keys

- e.g., Advanced Encryption Standard (AES) utilizes 256-bit keys ($2^{256} \approx 10^{77}$ possibilities)

note: the private key must be shared securely

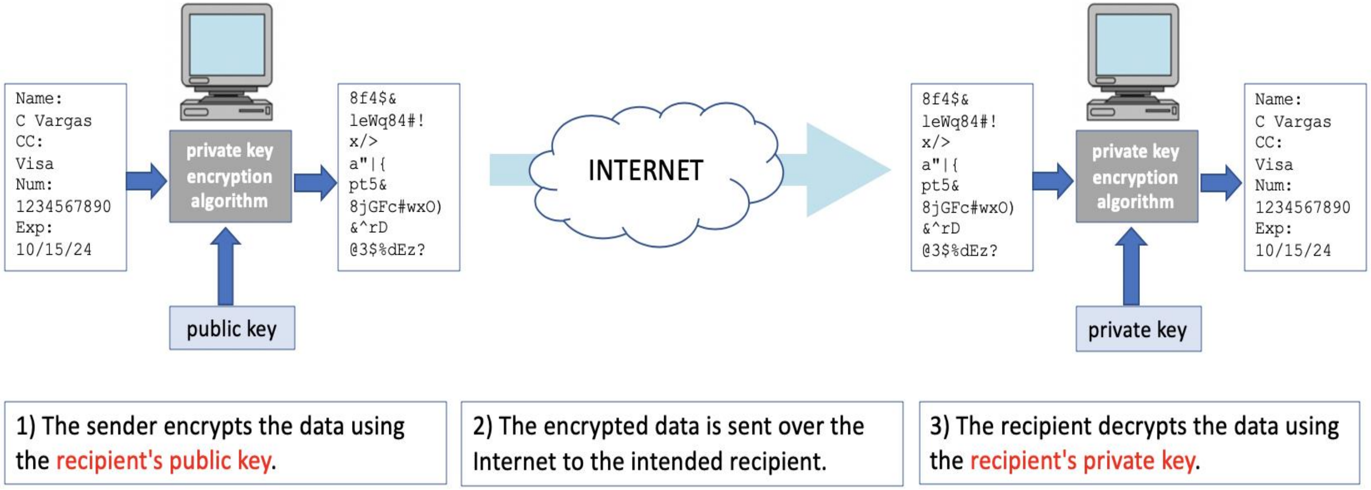
- face-to-face meeting? guarded courier?
- not feasible for online commerce

Public-Key Encryption

in 1976, Whitfield Diffie and Martin Hellman proposed a new approach

- instead of a single, private key, *public-key encryption* utilizes a pair of keys
  - a *public key* is used to encrypt messages
  - a *private key* is required to decrypt the message

- the only way to decrypt a message encrypted with a public key is using the corresponding private key
  - as long as the recipient keeps the private key secure, they are free to share the public key with anyone



1) The sender encrypts the data using the recipient's public key.

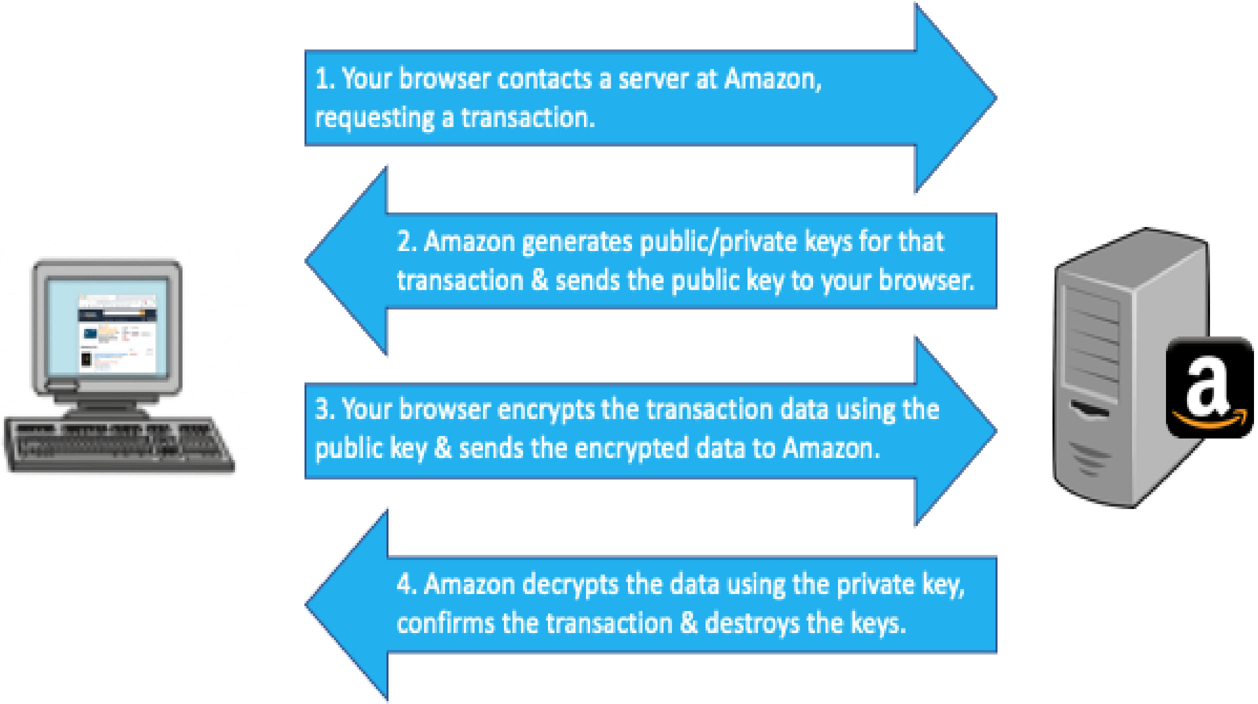
2) The encrypted data is sent over the Internet to the intended recipient.

3) The recipient decrypts the data using the recipient's private key.

Encryption and e-commerce

public-key encryption is the basis of almost all secure communication over the Internet

- without it, e-commerce would not be possible
- consider what happens when you buy something online



1. Your browser contacts a server at Amazon, requesting a transaction.

2. Amazon generates public/private keys for that transaction & sends the public key to your browser.

3. Your browser encrypts the transaction data using the public key & sends the encrypted data to Amazon.

4. Amazon decrypts the data using the private key, confirms the transaction & destroys the keys.

public-key encryption is also used whenever you surf the Web with `https://` or use a secure Wi-Fi connection

Architecture

subfield concerned with methods of organizing hardware components into efficient, reliable systems

application: *parallel processing*

- multiple processors can sometimes be utilized to share the computational load
- there are costs associated with coordinating the processors and dividing the work, so not well suited for all tasks
- understanding when parallel processing can be used effectively is a common task for computer architects

- e.g., Core 2 Duo and i3 processors are dual core - integrate the circuitry for 2 processors
  - can execute two different instructions simultaneously, potentially double execution speed
  - similarly, i5 and i7 have 4 cores, i9 has 8 cores

- e.g., high-end Web Servers utilize multiple processors
  - can service multiple requests simultaneously by distributing the load among the processors

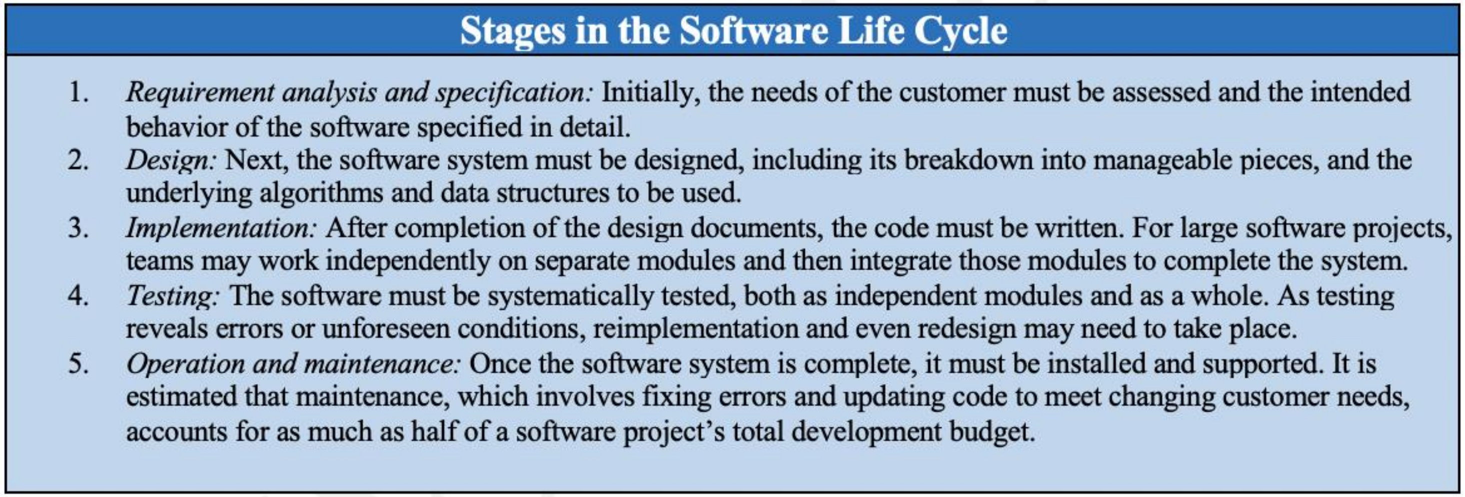
- IBM's Deep Blue contained 32 general-purpose processors and 512 special-purpose chess processors
  - worked in parallel to evaluate 200 million chess moves per second)
  - first computer to beat a world champ (1997)

- its descendent, Watson, contains 2,880 processors
  - won Jeopardy challenge in 2011
  - used in many applications (weather modeling, medical diagnosis, satellite imagery analysis)

Software Engineering

subfield concerned with creating effective software systems

- large projects can encompass millions of lines of code
- teams of programmers work together to make an integrated whole
  - coordination and testing are key to successful projects



Stages in the Software Life Cycle

1. *Requirement analysis and specification:* Initially, the needs of the customer must be assessed and the intended behavior of the software specified in detail.
2. *Design:* Next, the software system must be designed, including its breakdown into manageable pieces, and the underlying algorithms and data structures to be used.
3. *Implementation:* After completion of the design documents, the code must be written. For large software projects, teams may work independently on separate modules and then integrate those modules to complete the system.
4. *Testing:* The software must be systematically tested, both as independent modules and as a whole. As testing reveals errors or unforeseen conditions, reimplementation and even redesign may need to take place.
5. *Operation and maintenance:* Once the software system is complete, it must be installed and supported. It is estimated that maintenance, which involves fixing errors and updating code to meet changing customer needs, accounts for as much as half of a software project's total development budget.

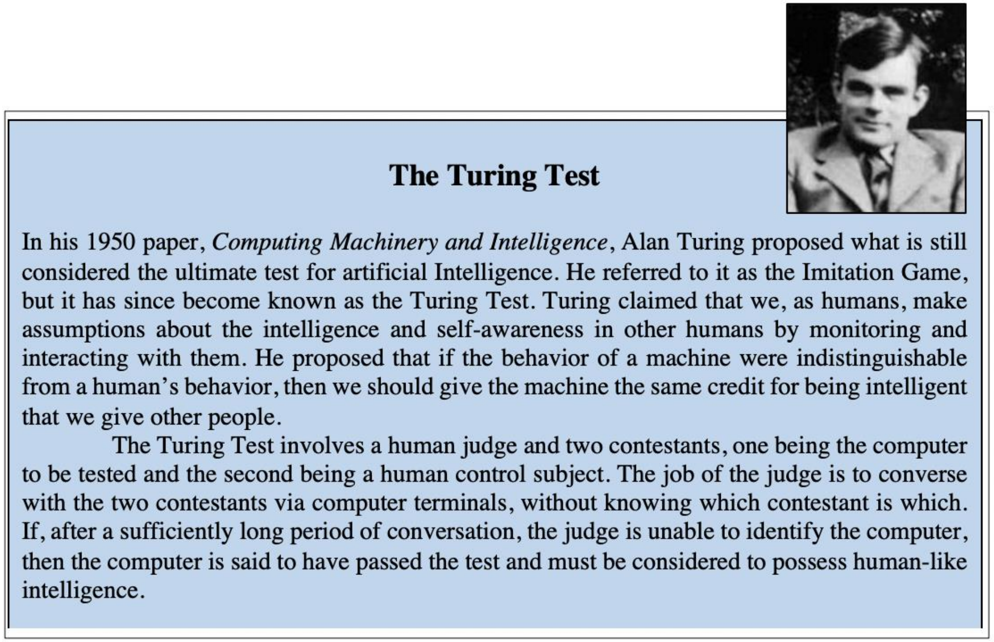
- software demand continues to grow, placing pressure on programmers to produce at faster rates
  - clearly, there is a limit to personal productivity
  - simply adding more programmers does not solve the problem: increasing numbers means increased complexity, and coordination becomes an even bigger challenge
- the adoption of the object-oriented programming methodology has made it easier to reuse code

Artificial Intelligence

subfield that attempts to make computers exhibit human-like characteristics (e.g., the ability to reason and think)

- in 1950, Turing predicted intelligent computers by 2000 (still not even close)
- but, progress has been made in many A.I. realms
  - robots in manufacturing
  - expert systems – programs that encapsulate expert knowledge in a specific domain (e.g., for medical diagnosis, credit card fraud detection)
  - neural computing – design of architectures that mimic the brain

    neural networks are used in handwriting analysis, self-driving cars, facial recognition, …



The Turing Test

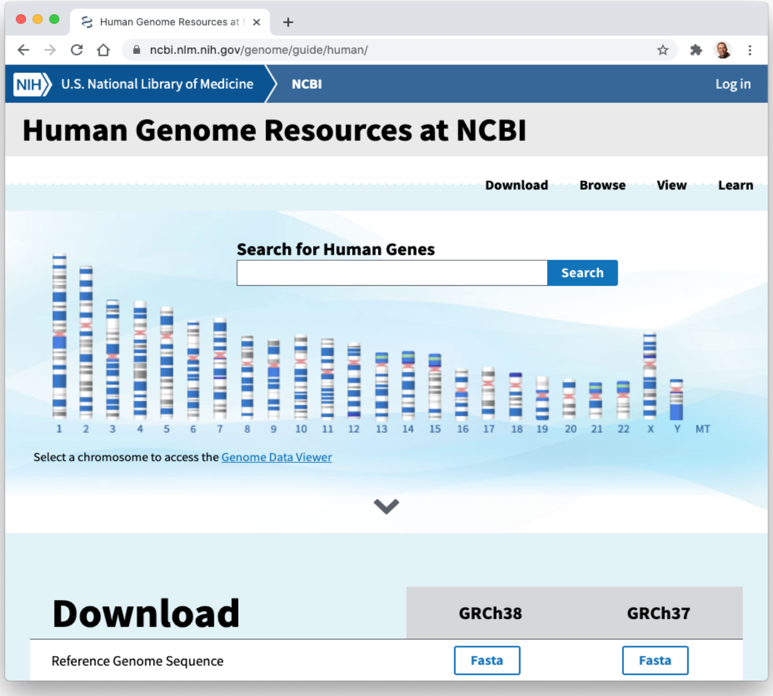
In his 1950 paper, *Computing Machinery and Intelligence*, Alan Turing proposed what is still considered the ultimate test for artificial Intelligence. He referred to it as the Imitation Game, but it has since become known as the Turing Test. Turing claimed that we, as humans, make assumptions about the intelligence and self-awareness in other humans by monitoring and interacting with them. He proposed that if the behavior of a machine were indistinguishable from a human's behavior, then we should give the machine the same credit for being intelligent that we give other people.

The Turing Test involves a human judge and two contestants, one being the computer to be tested and the second being a human control subject. The job of the judge is to converse with the two contestants via computer terminals, without knowing which contestant is which. If, after a sufficiently long period of conversation, the judge is unable to identify the computer, then the computer is said to have passed the test and must be considered to possess human-like intelligence.

Bioinformatics

multi-disciplinary field that bridges the gap between biology and computer science

- focuses on using computers and computer science techniques to solve biological problems
- computers are integrated with various scientific tools
  - e.g., microscopes connected to computers and digital cameras
- computer are used to model biological systems
  - e.g., pharmaceutical companies model drug interactions to save time and money
- computers are used to store and process large amounts of biological data
  - e.g., Human Genome Project stores and provides tools for studying DNA



Human Genome Resources at
ncbi.nlm.nih.gov/genome/guide/human/
NIH
U.S. National Library of Medicine
NCBI
Log in
Download
Browse
View
Learn
Search
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
X
Y
MT

Human Genome Resources at NCBI

Search for Human Genes

Select a chromosome to access the Genome Data Viewer

Download

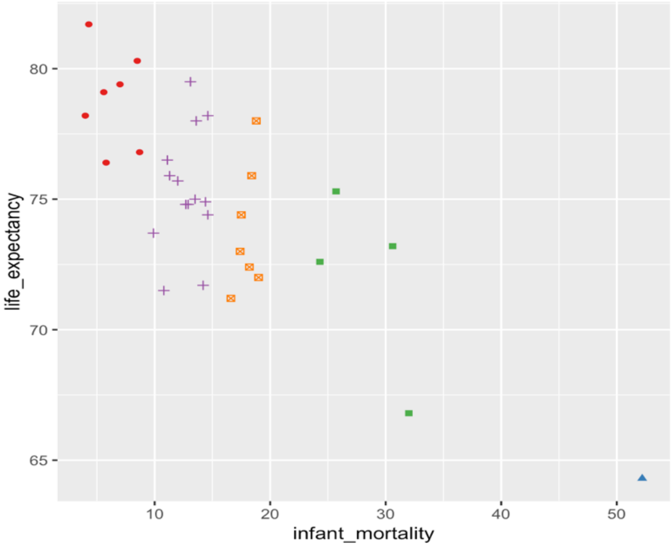
	GRCh38	GRCh37
Reference Genome Sequence	Fasta	Fasta

Data Science

data science utilizes concepts and methods from computer science, math and statistics to predict outcomes and extract insights from collections of data

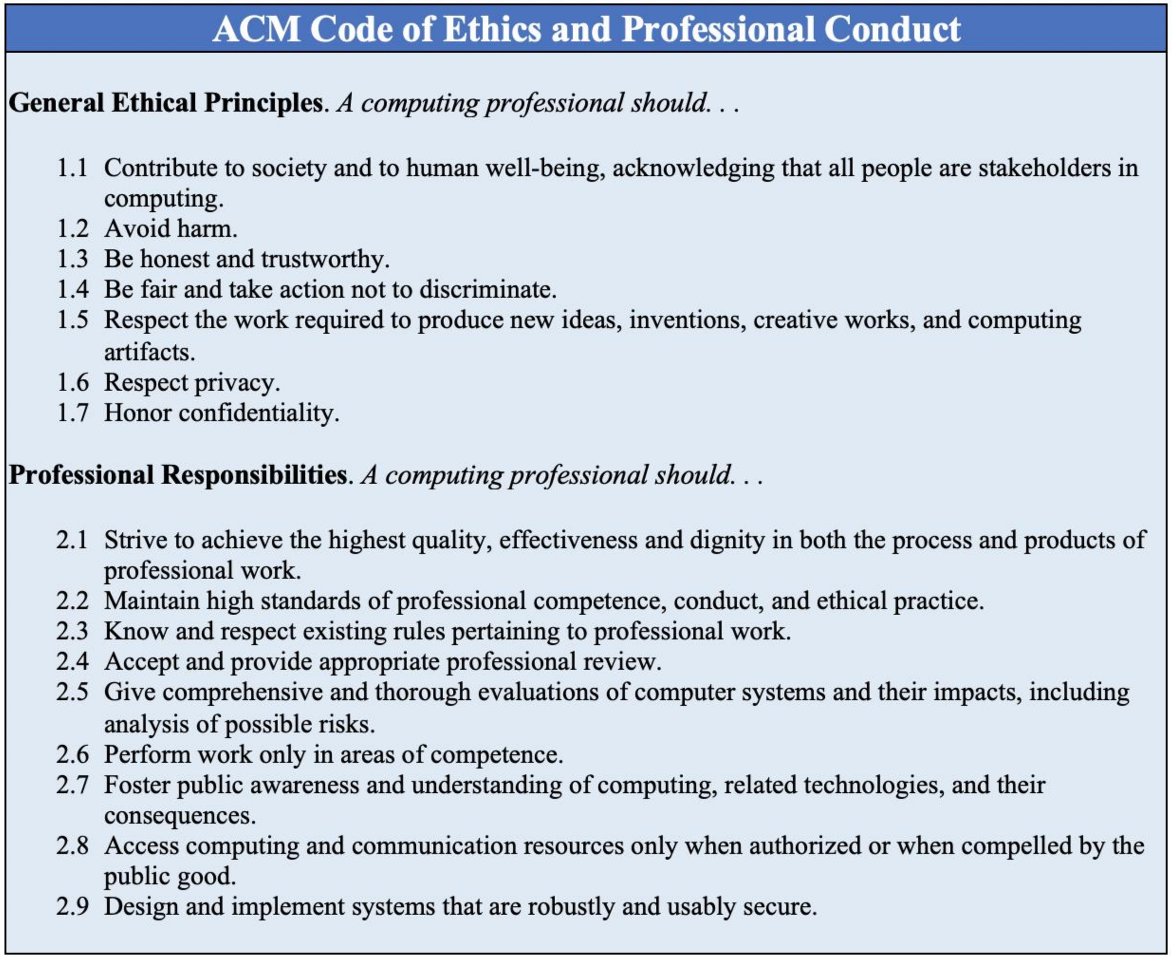
- supervised learning: learn to predict future outcomes based on past behavior
  - predicting the fluctuations in the stock market based on past performance
  - Amazon predicting which products you may want based on past purchases
  - Netflix predicting which shows you might like based on your viewing history

- unsupervised learning: process data to discover patterns and extract insights
  - a baseball team discovering player tendencies or weaknesses
  - a government agency using data insights to set policy



The Ethics of Computing

as technology becomes more prevalent in society, computing professionals must ensure that hardware and software are used safely, fairly, and effectively



ACM Code of Ethics and Professional Conduct

General Ethical Principles. *A computing professional should. . .*

1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.
1.2 Avoid harm.
1.3 Be honest and trustworthy.
1.4 Be fair and take action not to discriminate.
1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.
1.6 Respect privacy.
1.7 Honor confidentiality.

Professional Responsibilities. *A computing professional should. . .*

2.1 Strive to achieve the highest quality, effectiveness and dignity in both the process and products of professional work.
2.2 Maintain high standards of professional competence, conduct, and ethical practice.
2.3 Know and respect existing rules pertaining to professional work.
2.4 Accept and provide appropriate professional review.
2.5 Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.
2.6 Perform work only in areas of competence.
2.7 Foster public awareness and understanding of computing, related technologies, and their consequences.
2.8 Access computing and communication resources only when authorized or when compelled by the public good.
2.9 Design and implement systems that are robustly and usably secure.

Biology

biology is roughly defined as "the study of life"

- it is concerned with the characteristics and behaviors of organisms, how species and individuals come into existence, and the interactions they have with each other and with the environment

  (en.wikipedia.org/wiki/Biology)

biology encompasses a broad spectrum of academic fields that are often viewed as independent disciplines

- *ecology* and *evolutionary biology* study life at the habitat or population level
- *developmental biology* and *genetics* study life at organism level
- *physiology*, *anatomy*, and *histology* study life at the multicellular level
- *cell biology* studies life at the cellular level
- *molecular biology*, *biochemistry*, and *molecular genetics* study life at the atomic and molecular level

Impact of Computers

the history of biology dates as far back as the rise of various civilization

- while computers are relatively new, they have had a monumental impact on biological research

3 examples of impact:

1. computer technology is rapidly advancing the tools of scientific research
2. computer models are being used to study complex systems
3. computers are being used to store, process, and analyze large collections of biological data

note: this list is in no way exhaustive

- many aspects of biology and computer science are converging
- biology researchers must be savvy computer users and even programmers
- computer scientists must be able to solve interdisciplinary problems

Technology Tools/Resources

many of the traditional tools of biological research are integrating computer technology

e.g., the confocal microscope

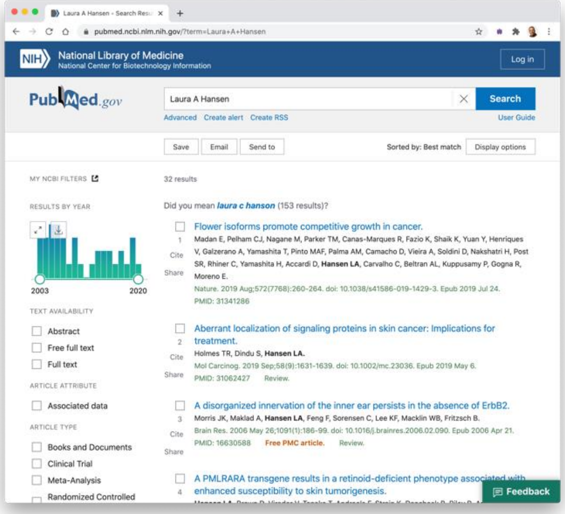
- invented by Marvin Minsky (computer science pioneer)
- works by focusing a laser on a dyed sample and measuring the fluorescent light emitted
- can be used to build up a 3-D model of a sample, stored on a computer

e.g., DNA Microarrays to measure the expression levels of genes

the Internet and the Web allow researchers to share data and publications

- speeds the dissemination of information and the advancement of science

e.g., PubMed, from the National Library of Medicine



Laura A Hansen

Sorted by: Best match

32 results

Did you mean *laura c hanson* (153 results)?

□ Abstract
□ Free full text
□ Full text

□ Associated data

□ Books and Documents
□ Clinical Trial
□ Meta-Analysis
□ Randomized Controlled

□ 1 Flower isoforms promote competitive growth in cancer.
Madan E, Pelham CJ, Nagane M, Parker TM, Canas-Marques R, Fazio K, Shaik K, Yuan Y, Henriques V, Galzerano A, Yamashita T, Pinto MAF, Palma AM, Camacho D, Vieira A, Soldini D, Nakshatri H, Post SR, Rhiner C, Yamashita H, Accardi D, **Hansen LA**, Carvalho C, Beltran AL, Kuppusamy P, Gogna R, Moreno E.
Nature. 2019 Aug;572(7768):260-264. doi: 10.1038/s41586-019-1429-3. Epub 2019 Jul 24.
PMID: 31341286

□ 2 Aberrant localization of signaling proteins in skin cancer: Implications for treatment.
Holmes TR, Dindu S, **Hansen LA.**
Mol Carcinog. 2019 Sep;58(9):1631-1639. doi: 10.1002/mc.23036. Epub 2019 May 6.
PMID: 31062427 Review.

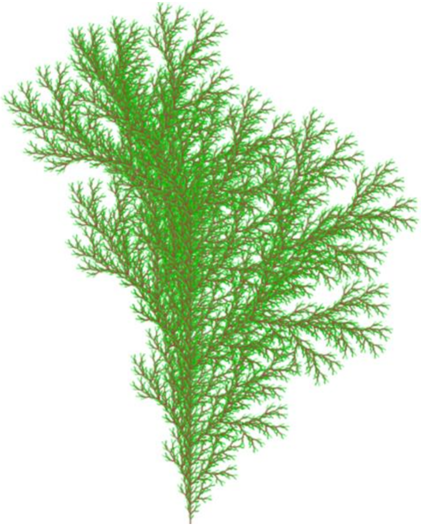
□ 3 A disorganized innervation of the inner ear persists in the absence of ErbB2.
Morris JK, Maklad A, **Hansen LA**, Feng F, Sorensen C, Lee KF, Macklin WB, Fritzsch B.
Brain Res. 2006 May 26;1091(1):186-99. doi: 10.1016/j.brainres.2006.02.090. Epub 2006 Apr 21.
PMID: 16630588 Free PMC article. Review.

□ 4 A PMLRARA transgene results in a retinoid-deficient phenotype associated with enhanced susceptibility to skin tumorigenesis.

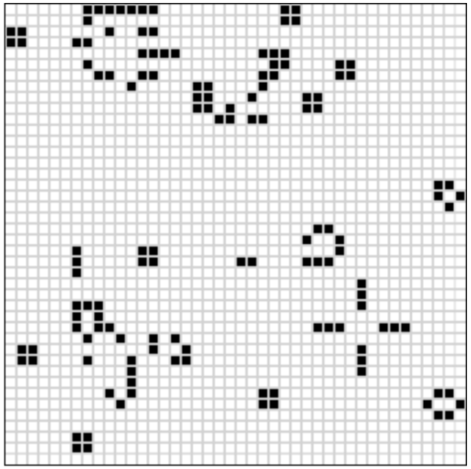
System Modeling

as computer memory and processing power has increased, it has become possible to model complex biological systems in software

- can attempt to discern natural laws or behaviors by observing the model under varying conditions
  - e.g., models of plant or seashell growth
  - e.g., the evolution of cooperative behavior in species, such as bird flocking
- can predict the effects of actions over long periods
  - e.g., the effects of automobile emissions on global warming
  - e.g., the effects of increased fishing on worldwide fishery stocks
- can avoid infeasible, unethical, or costly experimentation
  - e.g., predict the toxicity of a new drug based on a chemical/biological model as opposed to animal testing
  - e.g., study brain trauma using a neural network model

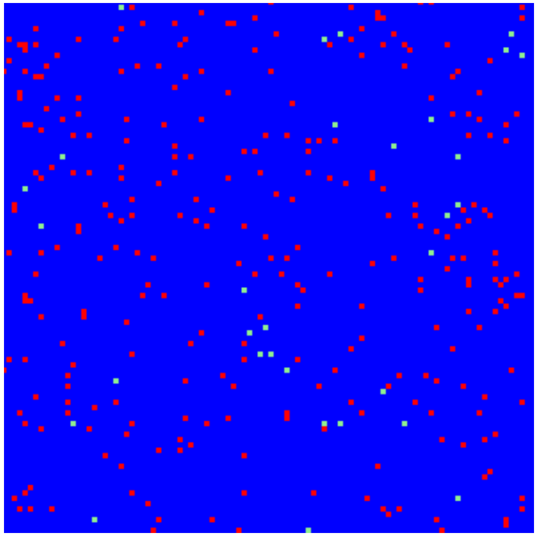


Ecosystem Modeling



in the late 1960s, John Conway showed that a simple model of an environment could produce complex and interesting behavior

- the environment is modeled as a 2-D grid of cells
- a cell can be alive (contain an organism) or dead
- simple rules model evolution
  1. a dead cell becomes alive in the next generation if it has exactly 3 neighbors
  2. a living cell survives in the next generation if it has 2 or 3 neighbors



Conway's ideas have been extended to a variety of ecosystems

- here, different colored cells denote different organisms (sharks & fish)

- other systems have modeled:
  - ✓ the growth of viruses
  - ✓ the spread of infectious diseases in a population
  - ✓ the behavior of an ant colony

Bioinformatics

perhaps the biggest impact of computers in biology is in storing, accessing, and processing large amounts of biological data

the new field of bioinformatics bridges biology and computer science (or informatics, as it is known in Europe)

- *broad definition of bioinformatics:* the use of computer science techniques to solve biological problems
- *narrower but common definition:* the application of computer science techniques to the representation and processing of biological data

as research tools advance, biologists are generating enormous amount of data

- a single experiment with genetic material can produce thousands or millions of data points
- computational and statistical tools are needed to analyze and understand such volumes of data

DNA Overview

DNA is the genetic blue-print of life

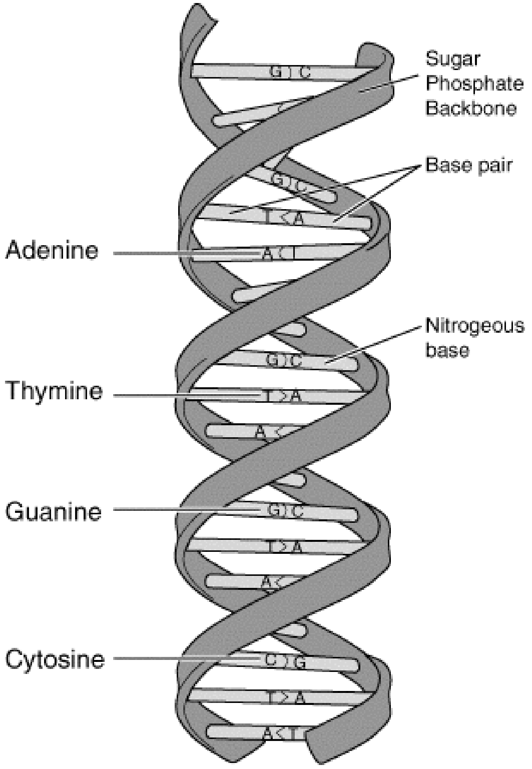
- made of nucleotides with four bases (A, T, G, C), organized in a double-helix
- the two strands match A+T and C+G base pairs
- can think of DNA as encoding information in base 4

a gene is a region of DNA that encodes the chemical structure of a protein

- proteins (e.g., enzymes, hormones, antibodies) control cellular and organ functions

it is currently believed that there are 20,000-30,000 different genes in human DNA

- roughly 3 billion base pairs



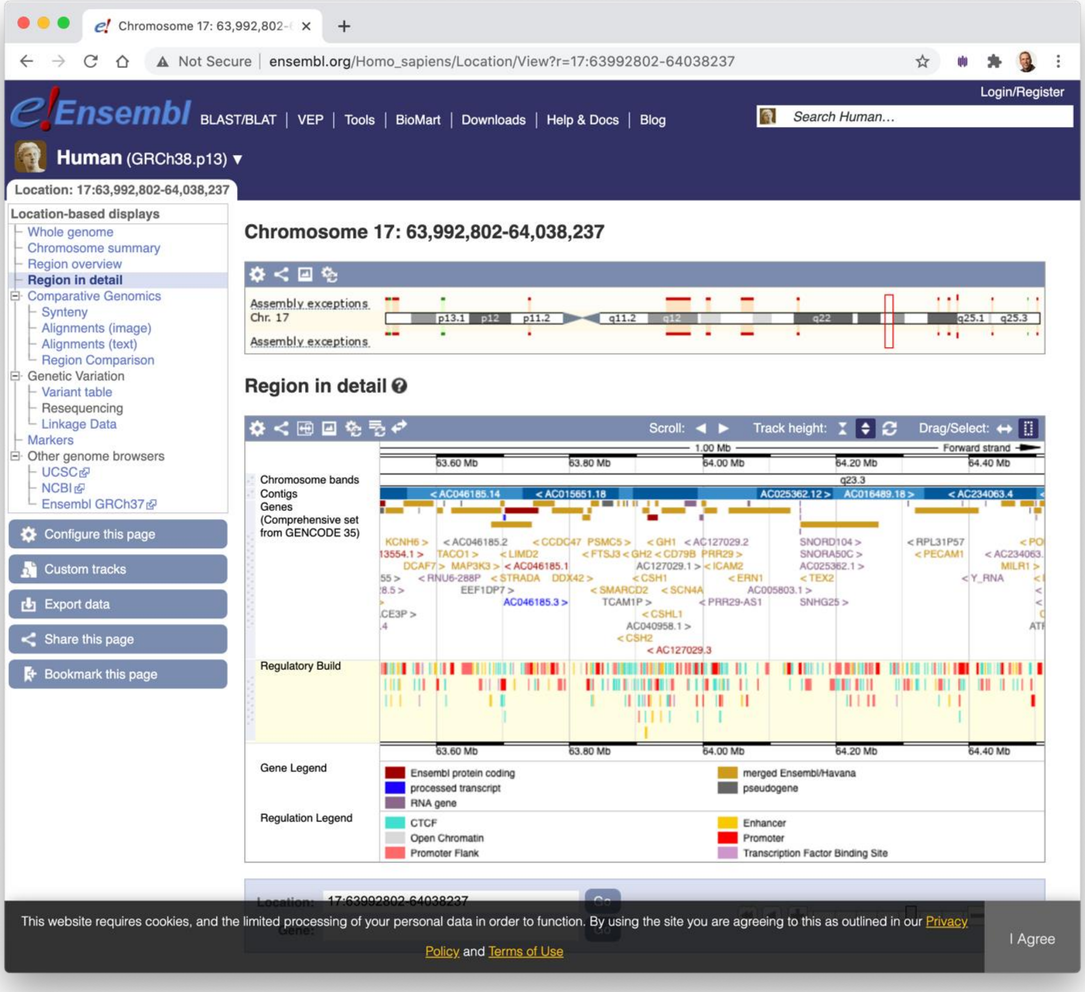
"If our strands of DNA were stretched out in a line, the 46 chromosomes making up the human genome would extend more than six feet. If the ... length of the 100 trillion cells could be stretched out, it would be ... over 113 billion miles. That is enough material to reach to the sun and back 610 times." *[Source: Centre for Integrated Genomics]*

DNA Databases and Tools

often, the source or purpose of a DNA sequence can be determined by comparing it with documented genetic material

- several large databases are available online
- tools for visualizing and/or searching the databases are also available

e.g., the Ensemble site (www.ensembl.org) contains visualizations of the human genome and other DNA sequences

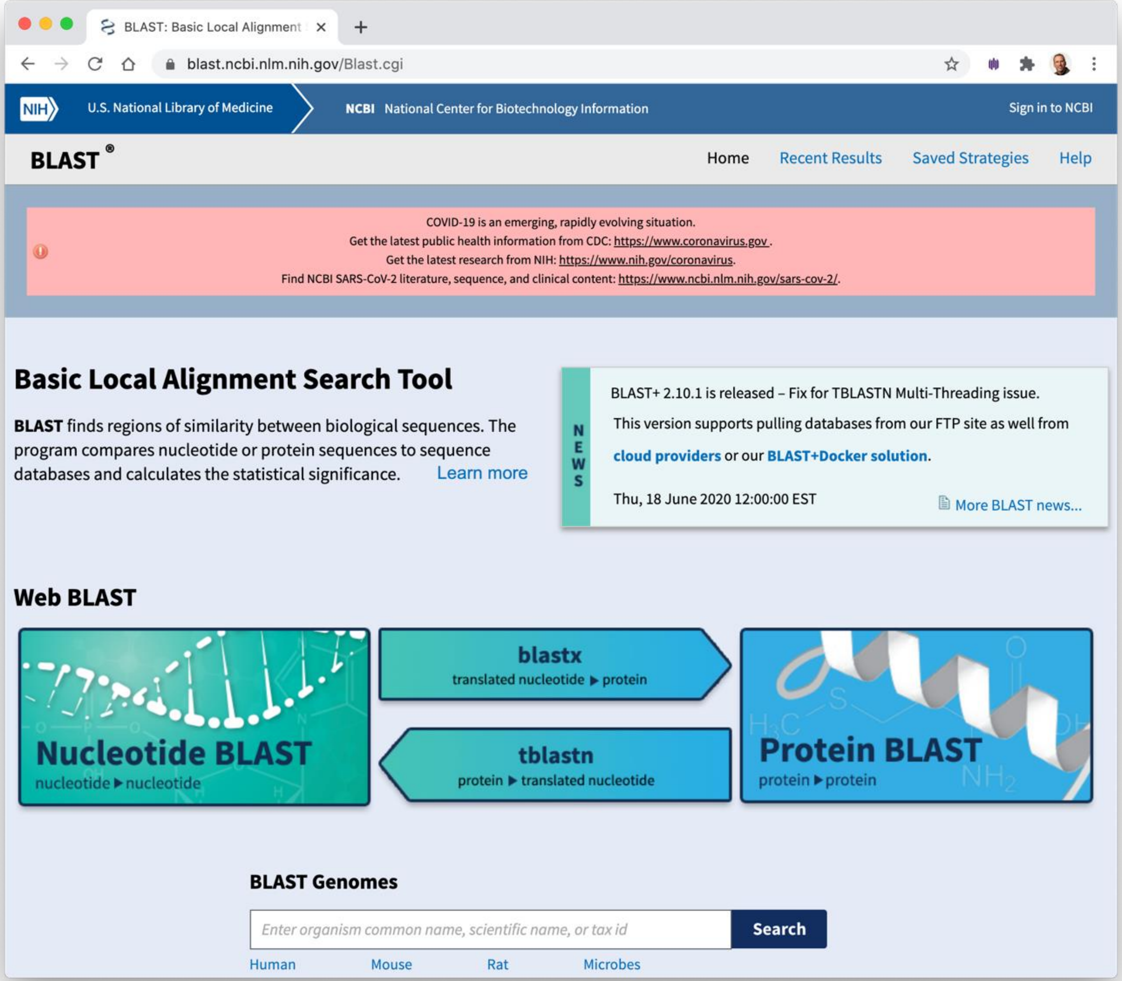


GenBank

the GenBank public repository of DNA and RNA sequence data contains

- partial or complete genomes for more than 300,000 organisms
- more than 1 trillion bases of sequence data
- roughly 250 million new DNA sequences are added per month

the database can be accessed and searched using various tools at www.ncbi.nlm.nih.gov



BLAST: Basic Local Alignment
blast.ncbi.nlm.nih.gov/Blast.cgi
NIH
U.S. National Library of Medicine
NCBI National Center for Biotechnology Information
Sign in to NCBI
BLAST®
Home
Recent Results
Saved Strategies
Help

COVID-19 is an emerging, rapidly evolving situation.
Get the latest public health information from CDC: https://www.coronavirus.gov .
Get the latest research from NIH: https://www.nih.gov/coronavirus.
Find NCBI SARS-CoV-2 literature, sequence, and clinical content: https://www.ncbi.nlm.nih.gov/sars-cov-2/.

Basic Local Alignment Search Tool

BLAST finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance. Learn more

NEWS

BLAST+ 2.10.1 is released – Fix for TBLASTN Multi-Threading issue.

This version supports pulling databases from our FTP site as well from **cloud providers** or our **BLAST+Docker solution**.

Thu, 18 June 2020 12:00:00 EST

More BLAST news...

Web BLAST

BLAST Genomes

Analog vs. Digital

there are two ways data can be stored electronically

1. *analog* values represent data in a way that is analogous to real life
   - signals can vary continuously across an infinite range of values
2. *digital* values utilize only a finite set of values

Analog/Digital Tradeoffs

the major tradeoff between analog and digital is variability vs. reproducibility

- analog allows for a (potentially) infinite number of unique signals, but they are harder to reproduce
  - good for storing data that is highly variable but does not need to be reproduced exactly
- digital signals limit the number of representable signals, but they are easily remembered and reproduced
  - good for storing data when reproducibility is paramount

when storing data on a computer, reproducibility is paramount

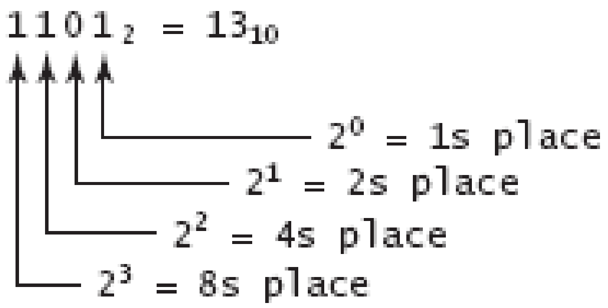
- changing a single bit in a file can drastically change the data

modern computers save and manipulate data as discrete (digital) values

- the most effective systems use two distinct (binary) states for representing data
- in essence, all data is stored as *binary numbers*

Binary Numbers

in the binary number system, all values are represented using only the two binary digits 0 and 1, which are called *bits*



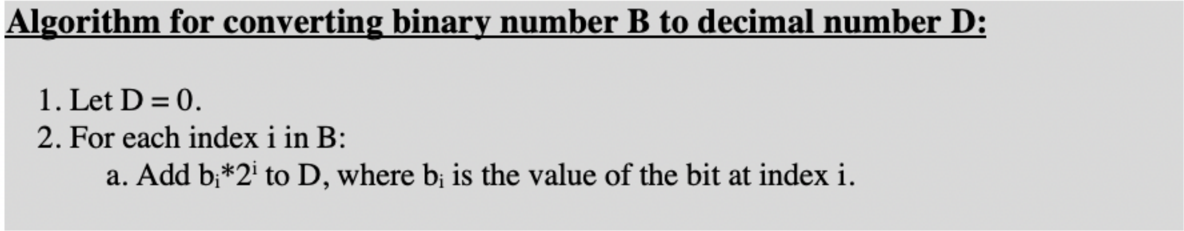
we can also refer to bits by index

- rightmost bit is index 0 (representing 2^0 = 1s place)
- next from right is index 1 (representing 2^1 = 2s place
- next over is index 2 (representing 2^2 = 4s place
- . . .
- i^{th} index (from right) represents 2^i s place

note:

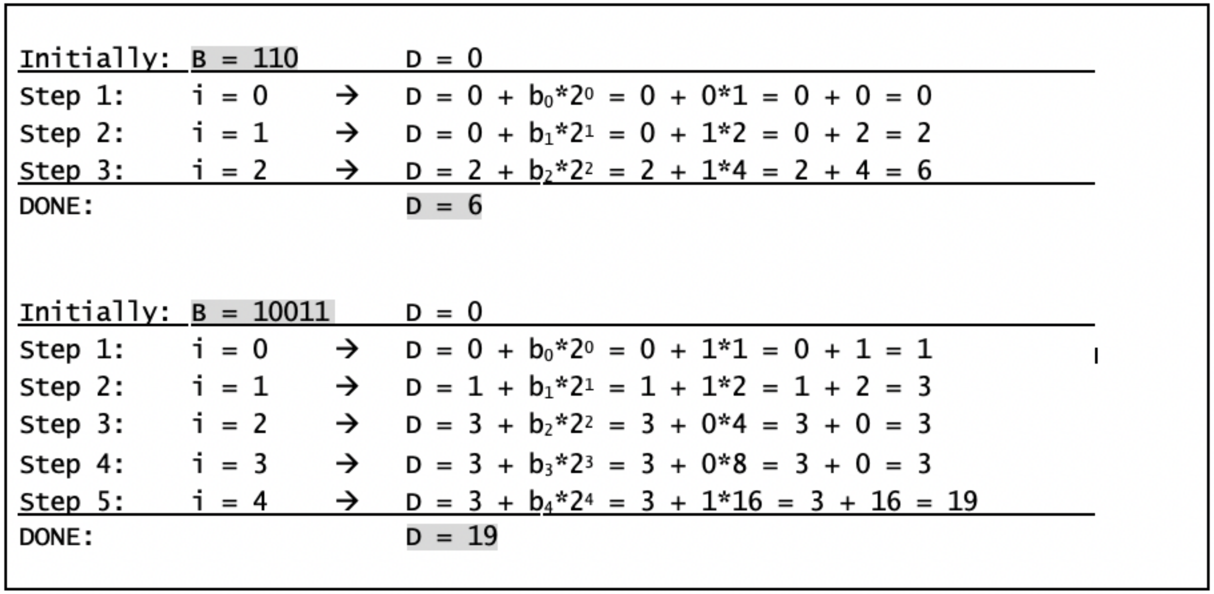
- all even numbers end with a 0 bit; odd numbers with 1 bit WHY?
- adding a 0 bit at the end it doubles its value WHY?

Binary → Decimal



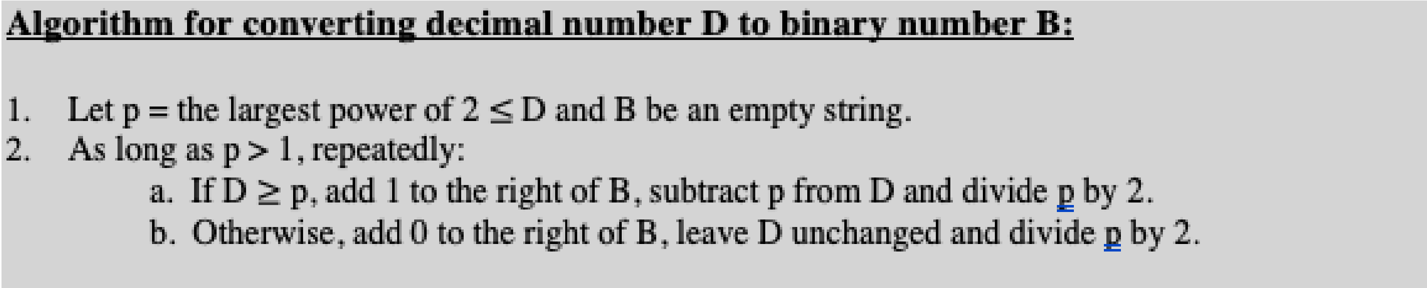
Algorithm for converting binary number B to decimal number D:

```
1. Let D = 0.
2. For each index i in B:
      a. Add $b_i*2^i$ to D, where $b_i$ is the value of the bit at index i.
```



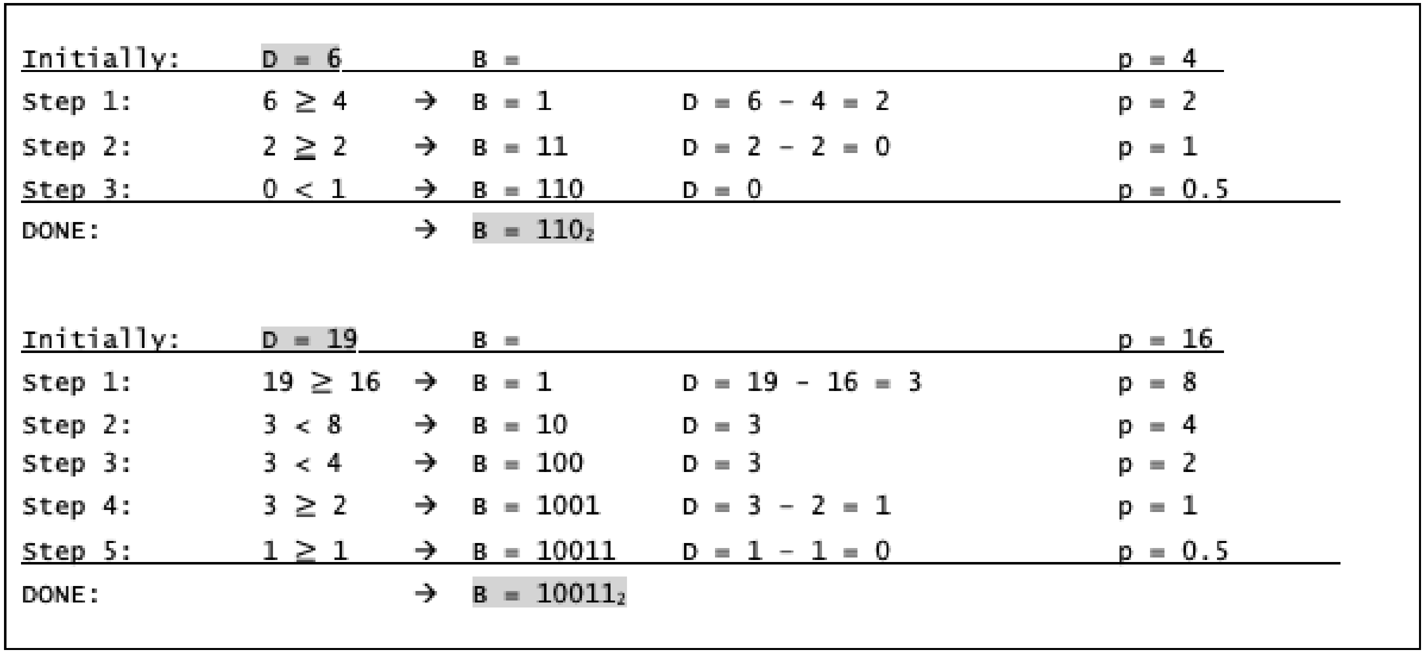
```
Initially:  B = 110          D = 0
Step 1:     i = 0     →      D = 0 + b_0*2^0 = 0 + 0*1 = 0 + 0 = 0
Step 2:     i = 1     →      D = 0 + b_1*2^1 = 0 + 1*2 = 0 + 2 = 2
Step 3:     i = 2     →      D = 2 + b_2*2^2 = 2 + 1*4 = 2 + 4 = 6
DONE:                        D = 6


Initially:  B = 10011        D = 0
Step 1:     i = 0     →      D = 0 + b_0*2^0 = 0 + 1*1 = 0 + 1 = 1
Step 2:     i = 1     →      D = 1 + b_1*2^1 = 1 + 1*2 = 1 + 2 = 3
Step 3:     i = 2     →      D = 3 + b_2*2^2 = 3 + 0*4 = 3 + 0 = 3
Step 4:     i = 3     →      D = 3 + b_3*2^3 = 3 + 0*8 = 3 + 0 = 3
Step 5:     i = 4     →      D = 3 + b_4*2^4 = 3 + 1*16 = 3 + 16 = 19
DONE:                        D = 19
```

Decimal → Binary



Algorithm for converting decimal number D to binary number B:

1. Let p = the largest power of 2 ≤ D and B be an empty string.
2. As long as p > 1, repeatedly:
   a. If D ≥ p, add 1 to the right of B, subtract p from D and divide p by 2.
   b. Otherwise, add 0 to the right of B, leave D unchanged and divide p by 2.



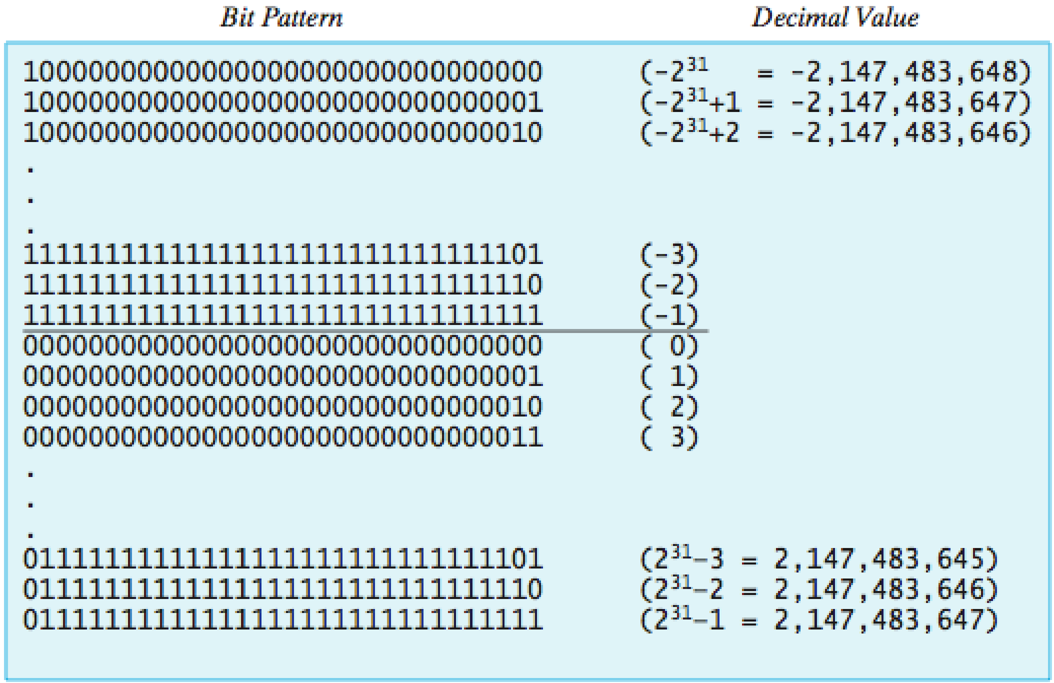
```
Initially:    D = 6              B =                                   p = 4
Step 1:       6 ≥ 4      →      B = 1         D = 6 - 4 = 2           p = 2
Step 2:       2 ≥ 2      →      B = 11        D = 2 - 2 = 0           p = 1
Step 3:       0 < 1      →      B = 110       D = 0                   p = 0.5
DONE:                    →      B = 110₂


Initially:    D = 19             B =                                   p = 16
Step 1:       19 ≥ 16    →      B = 1         D = 19 - 16 = 3         p = 8
Step 2:       3 < 8      →      B = 10        D = 3                   p = 4
Step 3:       3 < 4      →      B = 100       D = 3                   p = 2
Step 4:       3 ≥ 2      →      B = 1001      D = 3 - 2 = 1           p = 1
Step 5:       1 ≥ 1      →      B = 10011     D = 1 - 1 = 0           p = 0.5
DONE:                    →      B = 10011₂
```

Representing Integers

when an integer value must be saved on a computer, its binary equivalent can be encoded as a bit pattern and stored digitally

usually, a fixed size (e.g., 32 bits) is used for each integer so that the computer knows where one integer ends and another begins

- the initial bit in each pattern acts as the *sign bit* (0=positive, 1=negative)
- negative numbers are represented in *two's complement notation*
    - the "largest" bit pattern corresponds to the smallest absolute value (-1)



Bit Pattern	Decimal Value
10000000000000000000000000000000	(-2^{31} = -2,147,483,648)
10000000000000000000000000000001	(-2^{31}+1 = -2,147,483,647)
10000000000000000000000000000010	(-2^{31}+2 = -2,147,483,646)
.	
.	
.	
11111111111111111111111111111101	(-3)
11111111111111111111111111111110	(-2)
11111111111111111111111111111111	(-1)
00000000000000000000000000000000	(0)
00000000000000000000000000000001	(1)
00000000000000000000000000000010	(2)
00000000000000000000000000000011	(3)
.	
.	
.	
01111111111111111111111111111101	(2^{31}-3 = 2,147,483,645)
01111111111111111111111111111110	(2^{31}-2 = 2,147,483,646)
01111111111111111111111111111111	(2^{31}-1 = 2,147,483,647)

Representing Real Numbers

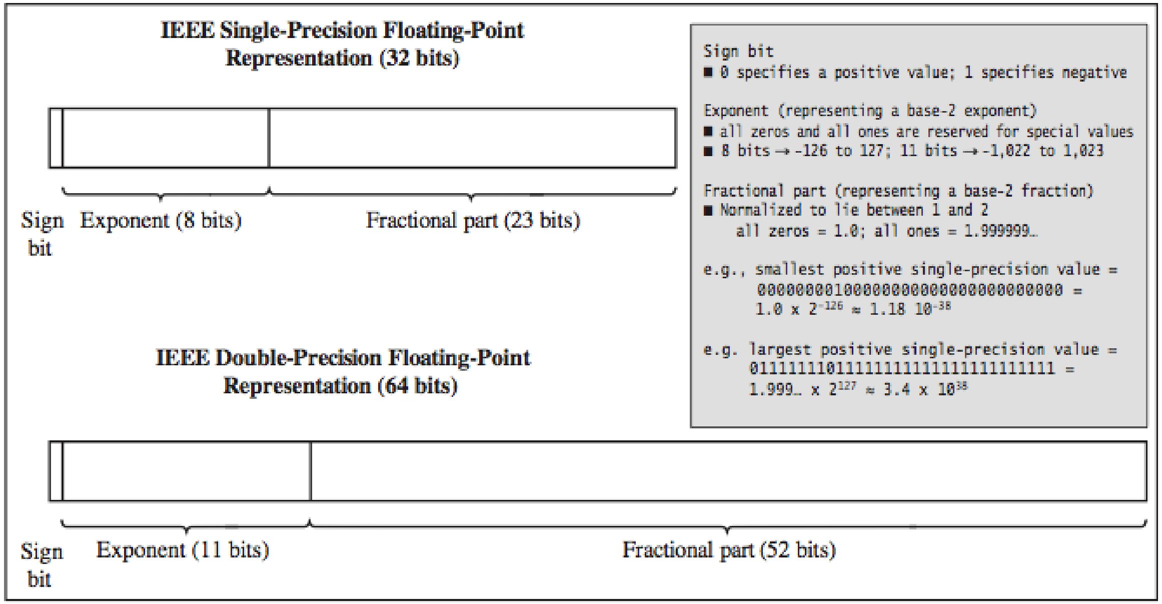
a real number can be uniquely identified by the two components of its scientific notation (fractional part and the exponent)

123.456 = 0.123456 x 10^3 0.0099 = 0.99 x 10^{-2}

thus, any real number can be stored as a pair of integers

- real numbers stored in this format are known as *floating point numbers*, since the decimal point moves (floats) to normalize the fraction

standard formats exist for storing real numbers, using either 32 or 64bits



IEEE Single-Precision Floating-Point Representation (32 bits)

IEEE Double-Precision Floating-Point Representation (64 bits)

```
Sign bit
■ 0 specifies a positive value; 1 specifies negative

Exponent (representing a base-2 exponent)
■ all zeros and all ones are reserved for special values
■ 8 bits → -126 to 127; 11 bits → -1,022 to 1,023

Fractional part (representing a base-2 fraction)
■ Normalized to lie between 1 and 2
   all zeros = 1.0; all ones = 1.999999…

e.g., smallest positive single-precision value =
      00000000100000000000000000000000 =
      1.0 x 2^-126 ≈ 1.18 10^-38

e.g. largest positive single-precision value =
     01111111011111111111111111111111 =
     1.999… x 2^127 ≈ 3.4 x 10^38
```

most programming languages represent integers and reals differently

JavaScript simplifies things by using IEEE double-precision floating point for all numbers

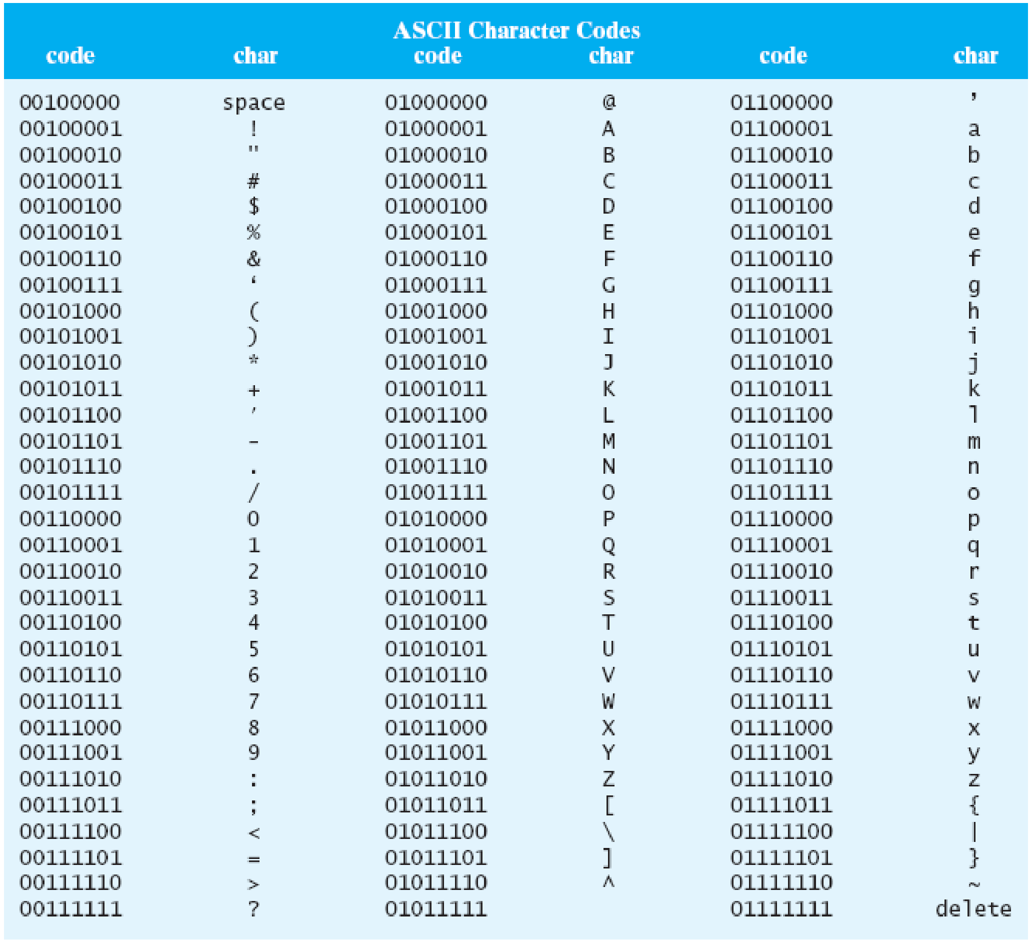
Representing Characters

characters have no natural correspondence to binary numbers

- computer scientists devised an arbitrary system for representing characters as bit patterns
- ASCII (American Standard Code for Information Interchange)
    - maps each character to a specific 8-bit pattern
    - note that all digits are contiguous, as are lower- and upper-case letters

```
'0' < '1' < … < '9'
'A' < 'B' < … < 'Z'
'a' < 'b' < … < 'z'
```

- Unicode is a 16-bit extension to ASCII that supports other languages

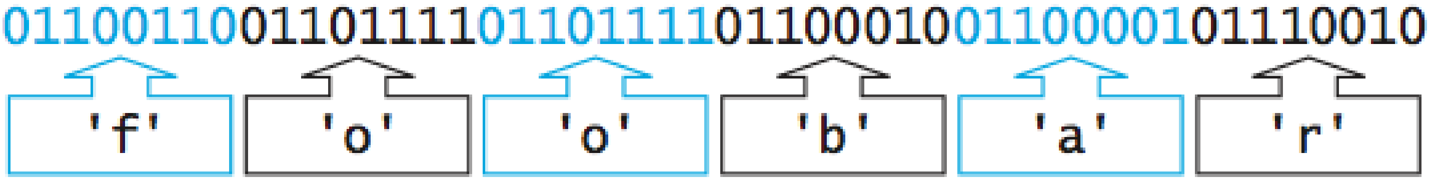


ASCII Character Codes

code	char	code	char	code	char
00100000	space	01000000	@	01100000	’
00100001	!	01000001	A	01100001	a
00100010	"	01000010	B	01100010	b
00100011	#	01000011	C	01100011	c
00100100	$	01000100	D	01100100	d
00100101	%	01000101	E	01100101	e
00100110	&	01000110	F	01100110	f
00100111	‘	01000111	G	01100111	g
00101000	(	01001000	H	01101000	h
00101001	)	01001001	I	01101001	i
00101010	*	01001010	J	01101010	j
00101011	+	01001011	K	01101011	k
00101100	,	01001100	L	01101100	l
00101101	-	01001101	M	01101101	m
00101110	.	01001110	N	01101110	n
00101111	/	01001111	O	01101111	o
00110000	0	01010000	P	01110000	p
00110001	1	01010001	Q	01110001	q
00110010	2	01010010	R	01110010	r
00110011	3	01010011	S	01110011	s
00110100	4	01010100	T	01110100	t
00110101	5	01010101	U	01110101	u
00110110	6	01010110	V	01110110	v
00110111	7	01010111	W	01110111	w
00111000	8	01011000	X	01111000	x
00111001	9	01011001	Y	01111001	y
00111010	:	01011010	Z	01111010	z
00111011	;	01011011	[	01111011	{
00111100	<	01011100	\	01111100	|
00111101	=	01011101	]	01111101	}
00111110	>	01011110	^	01111110	~
00111111	?	01011111		01111111	delete

Representing Text

strings can be represented as sequences of ASCII/Unicode codes, one for each character in the string



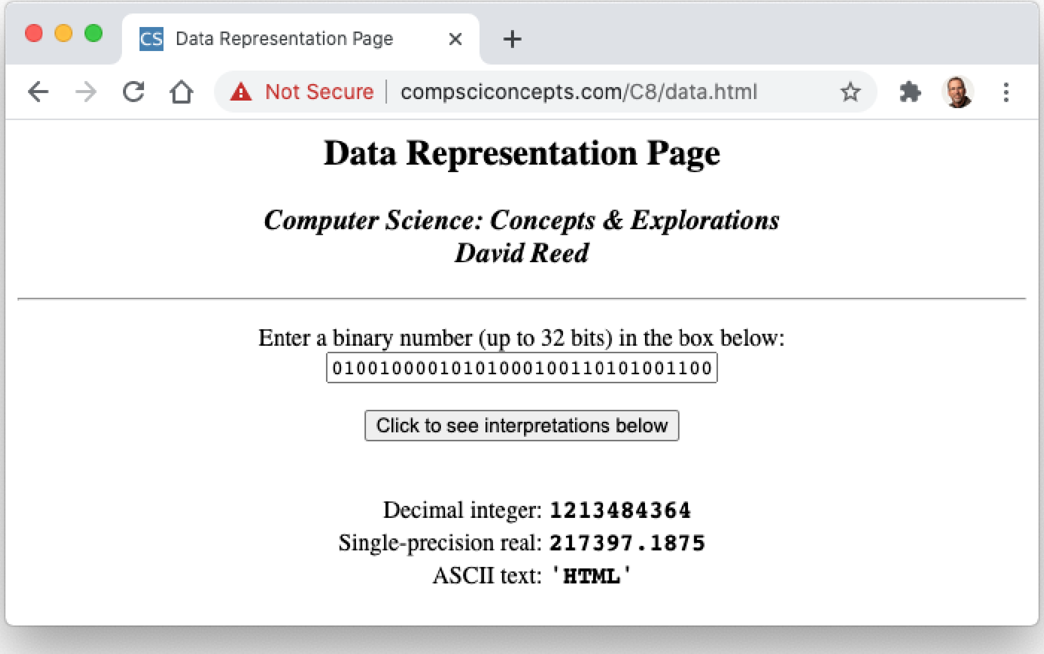
specific programs may store additional information along with the ASCII codes

- e.g. programming languages will often store the number of characters along with the ASCII/Unicode codes
- e.g., word processing programs will insert special character symbols to denote formatting (analogous to HTML tags in a Web page)

Distinguishing Data Types

how does a computer know what type of value is stored in a particular piece of memory?

- short answer: it doesn't
- when a program stores data in memory, it must store additional information as to what type of data the bit pattern represents
- thus, the same bit pattern might represent different values in different contexts


Data Representation Page
Not Secure
compsciconcepts.com/C8/data.html
Click to see interpretations below

Data Representation Page

Computer Science: Concepts & Explorations
David Reed

Enter a binary number (up to 32 bits) in the box below:

01001000010101000100110101001100

Decimal integer: **1213484364**
Single-precision real: **217397.1875**
ASCII text: **'HTML'**

Conditional Execution

so far, all of the code you have written has been *unconditionally executed*

- the browser carried out statements in the same set order

in contrast, many programming tasks require code that reacts differently under varying circumstances or conditions

e.g., a student's course grade depends upon his/her average
e.g., an ESP test requires recognizing when a subject guessed right
e.g., the outcome of a game depends upon die rolls or player moves

conditional execution refers to a program’s ability to execute a statement or sequence of statements only if some condition holds true

If Statements

in JavaScript, the simplest form of conditional statement is the *if statement*

- one action is taken if some condition is true, but a different action is taken if the condition is not true (called the *else case*)
- the else case is optional

general form of the if statement:

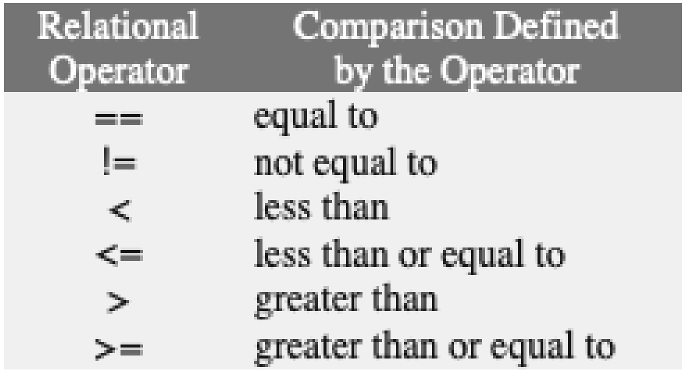
```
if (BOOLEAN_TEST) {
    STATEMENTS_EXECUTED_IF_TRUE
}
else {
    STATEMENTS_EXECUTED_IF_FALSE
}
```

note: indentation is not required, but it is STRONGLY RECOMMENDED to make an if statement readable

Boolean Tests

the test that controls an if statement can be any *Boolean expression* (i.e., an expression that evaluates to either `true` or `false`)

- Boolean tests are formed using *relational operators* because they test the relationships between values



Relational Operator	Comparison Defined by the Operator
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

NOTE:

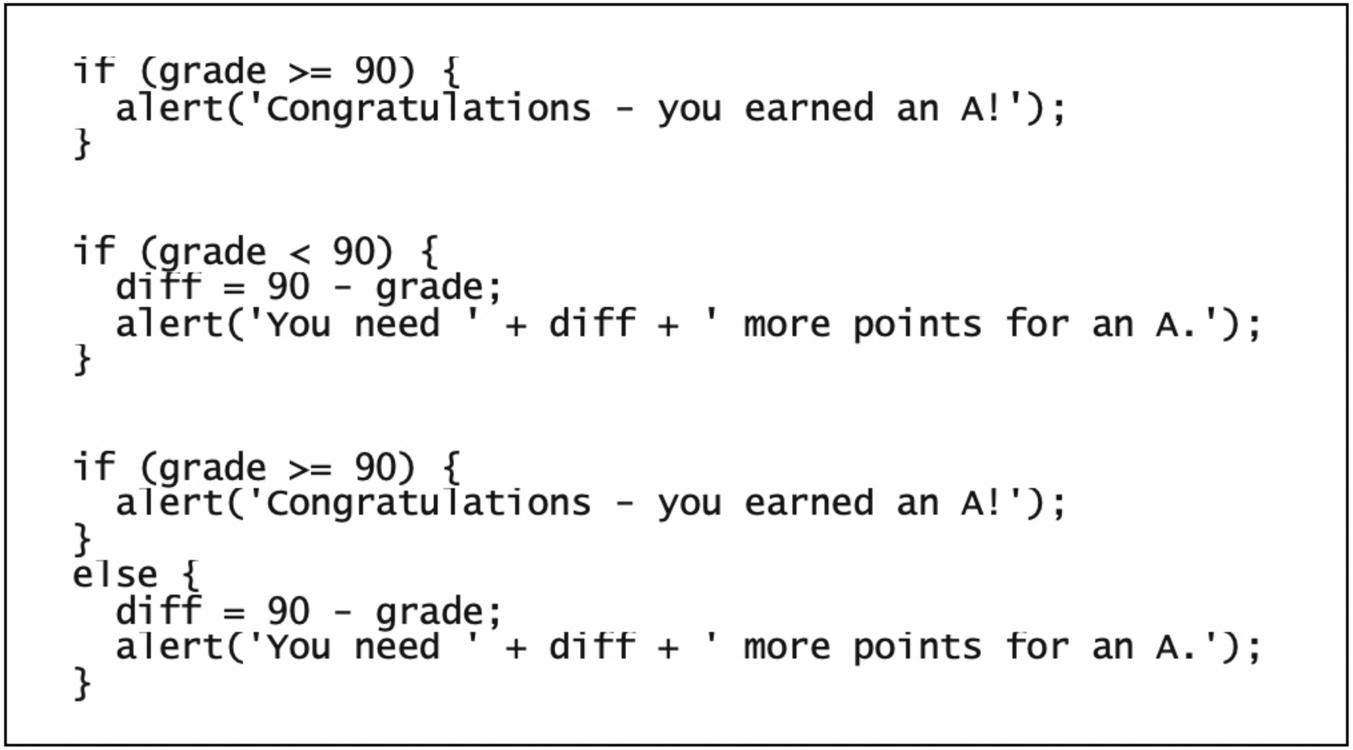
== is for comparisons

= is for assignments

the Boolean test in an if statement determines the code that will be executed

- if the test succeeds (evaluate to true), then the code inside the subsequent curly braces will execute
- if the test fails (evaluates to false), then the code inside the curly braces following the else will execute
- note that if the test fails and there is no else case, the program moves on to the statement directly after the if

If Statement Examples

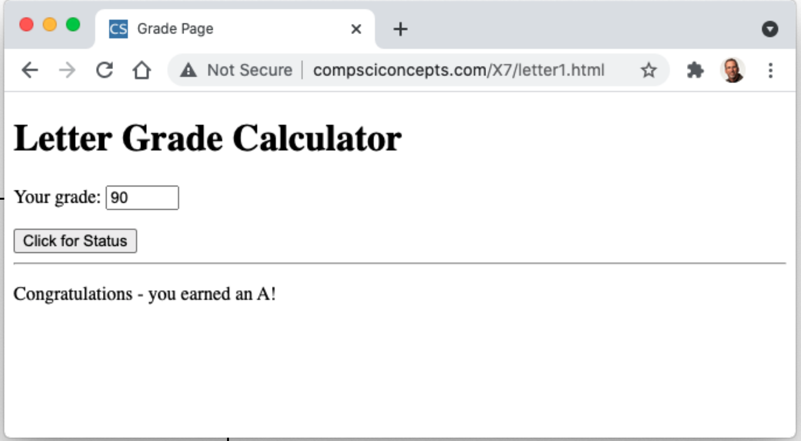


```
if (grade >= 90) {
  alert('Congratulations - you earned an A!');
}

if (grade < 90) {
  diff = 90 - grade;
  alert('You need ' + diff + ' more points for an A.');
}

if (grade >= 90) {
  alert('Congratulations - you earned an A!');
}
else {
  diff = 90 - grade;
  alert('You need ' + diff + ' more points for an A.');
}
```

an if statement is known as a *control statement*, since its purpose is to control the execution of other statements

Letter Grade Page

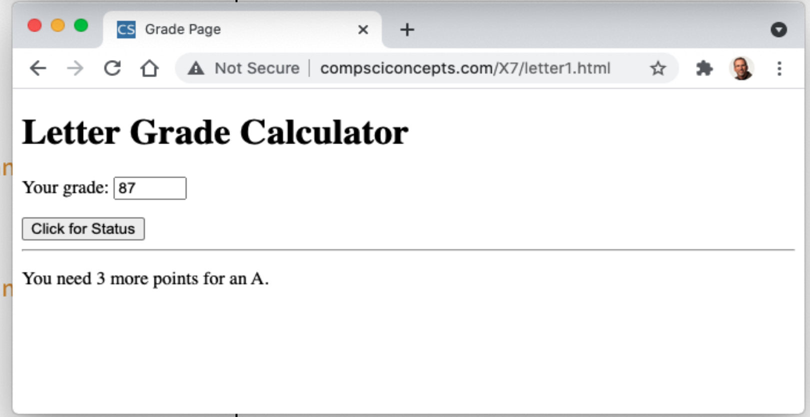
```
<!doctype html>
<!-- letter1.html                          Dave Reed -->
<!-- This page displays a student's letter grade. -->
<!-- ============================================ -->

<html>
 <head>
   <title> Grade Page </title>
   <script>
     function ShowStatus() {
       grade = gradeBox.valueAsNumber;

       if (grade >= 90) {
           outputP.innerHTML = 'Congratulations - you earned an
       }
       else {
           diff = 90 - grade;
           outputP.innerHTML = 'You need ' + diff + ' more poin
       }
     }
   </script>
 </head>

 <body>
   <h1>Letter Grade Calculator</h1>
   <p>Your grade: <input type="number" id="gradeBox" min=0 max=100.0></p>
   <button onclick="ShowStatus();">Click for Status</button>
   <hr>
   <p id="outputP"> </p>
 </body>
</html>
```



Letter Grade Calculator

Your grade: 90

Congratulations - you earned an A!



Letter Grade Calculator

Your grade: 87

You need 3 more points for an A.

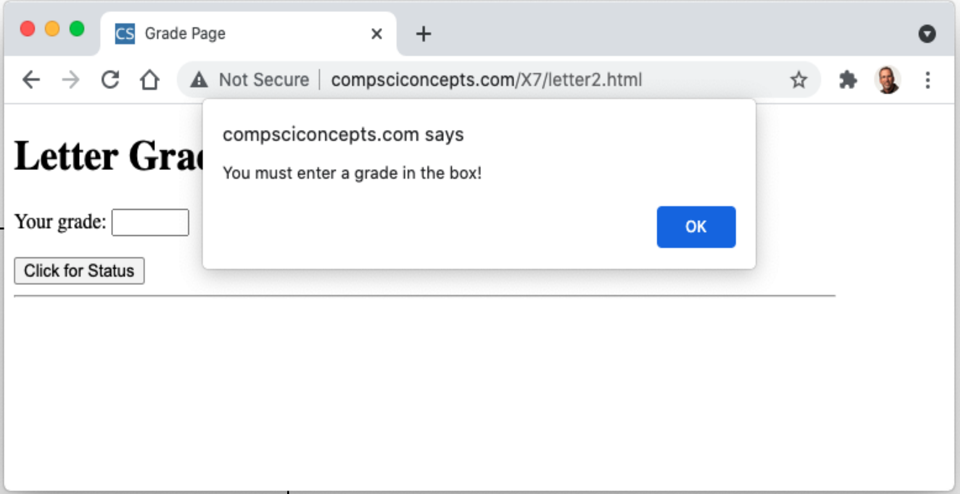
since else case is present, it will display one message or the other

Input Verification

common use of if statement:

- verify that user input has been entered
- warn the user if not

```
<!doctype html>
<!-- letter2.html                          Dave Reed -->
<!-- This page displays a student's letter grade. -->
<!-- ============================================ -->

<html>
 <head>
   <title> Grade Page </title>
   <script>
     function ShowStatus() {
       if (gradeBox.value == '') {
           alert('You must enter a grade in the box!');
       }
       else {
           grade = gradeBox.valueAsNumber;

           if (grade >= 90) {
               outputP.innerHTML = 'Congratulations - you earned an A!';
           }
           else {
               diff = 90 - grade;
               outputP.innerHTML = 'You need ' + diff + ' more points for an A.';
           }
       }
     }
   </script>
 </head>

 <body>
   <h1>Letter Grade Calculator</h1>
   <p>Your grade: <input type="number" id="gradeBox" min=0 max=100.0></p>
   <button onclick="ShowStatus();">Click for Status</button>
   <hr>
   <p id="outputP"> </p>
 </body>
</html>
```



compsciconcepts.com says

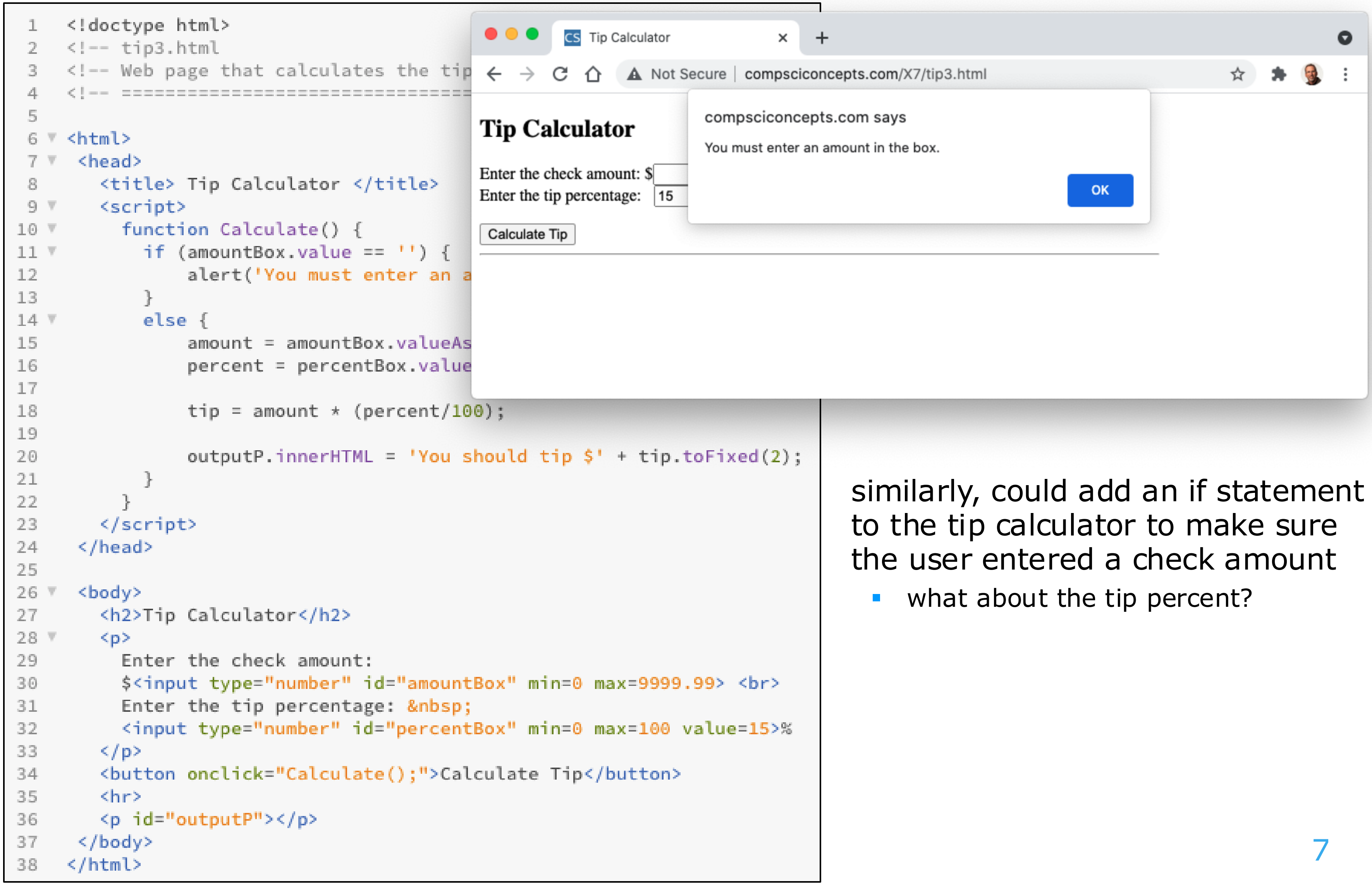
You must enter a grade in the box!

here, an if statement checks to see if number box is empty (i.e., its value attribute is '')

- if empty, displays an alert warning
- if not, calculates & displays letter grade as before

note: an if statement can be nested inside another if statement

Input Verification



```
<!doctype html>
<!-- tip3.html
<!-- Web page that calculates the tip
<!-- ================================

<html>
 <head>
   <title> Tip Calculator </title>
   <script>
     function Calculate() {
       if (amountBox.value == '') {
           alert('You must enter an a
       }
       else {
           amount = amountBox.valueAs
           percent = percentBox.value

           tip = amount * (percent/100);

           outputP.innerHTML = 'You should tip $' + tip.toFixed(2);
       }
     }
   </script>
 </head>

 <body>
   <h2>Tip Calculator</h2>
   <p>
     Enter the check amount:
     $<input type="number" id="amountBox" min=0 max=9999.99> <br>
     Enter the tip percentage: &nbsp;
     <input type="number" id="percentBox" min=0 max=100 value=15>%
   </p>
   <button onclick="Calculate();">Calculate Tip</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```

Tip Calculator

Enter the check amount: $

Enter the tip percentage: 15

compsciconcepts.com says

You must enter an amount in the box.

similarly, could add an if statement to the tip calculator to make sure the user entered a check amount

- what about the tip percent?

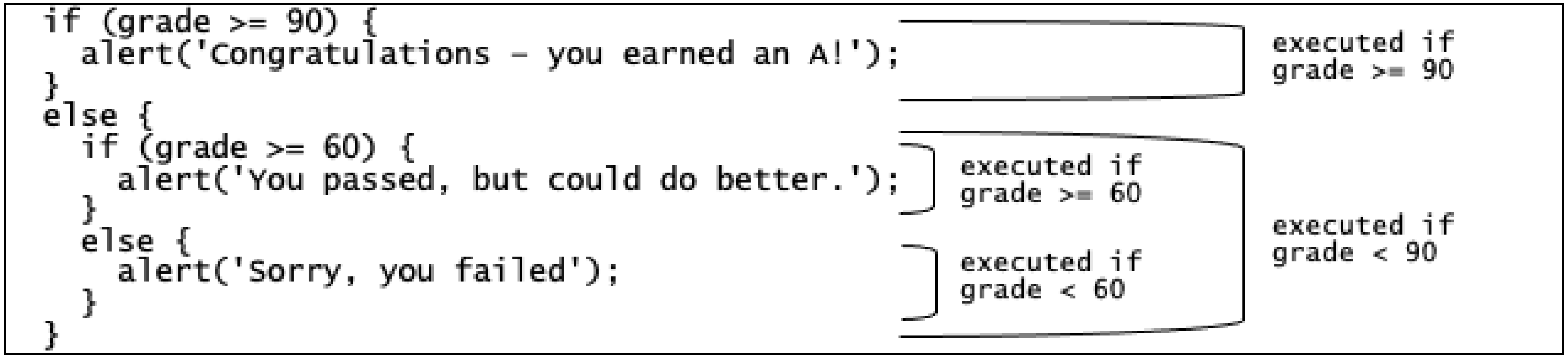
Cascading If-Else

programming tasks often require code that responds to more than one condition

- this can be accomplished by nesting one if statement inside of another

example: three different grade levels

- A-level (grade ≥ 90), passing (60 ≤ grade < 90), failing (grade < 60)
- the outer if-else distinguishes A from non-A grades
- the nested if-else further separates non-A grades into passing and failing

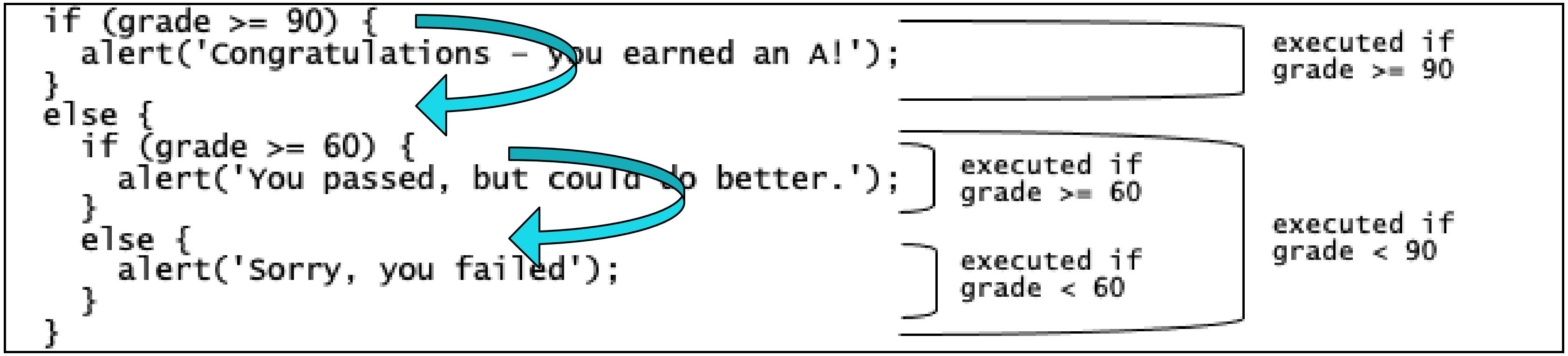


```
if (grade >= 90) {
  alert('Congratulations - you earned an A!');
}
else {
  if (grade >= 60) {
    alert('You passed, but could do better.');
  }
  else {
    alert('Sorry, you failed');
  }
}
```

Cascading If-else Structure

nested if-else statements are known as *cascading if-else structures* because control cascades down the branches

- the topmost level is evaluated first
- if the test succeeds, then the corresponding statements are executed and control moves to the next statement following the cascading if-else structure
- if the test fails, then control cascades down to the next if test
- in general, control cascades down the structure from one test to another until one succeeds or the end of the statement is reached



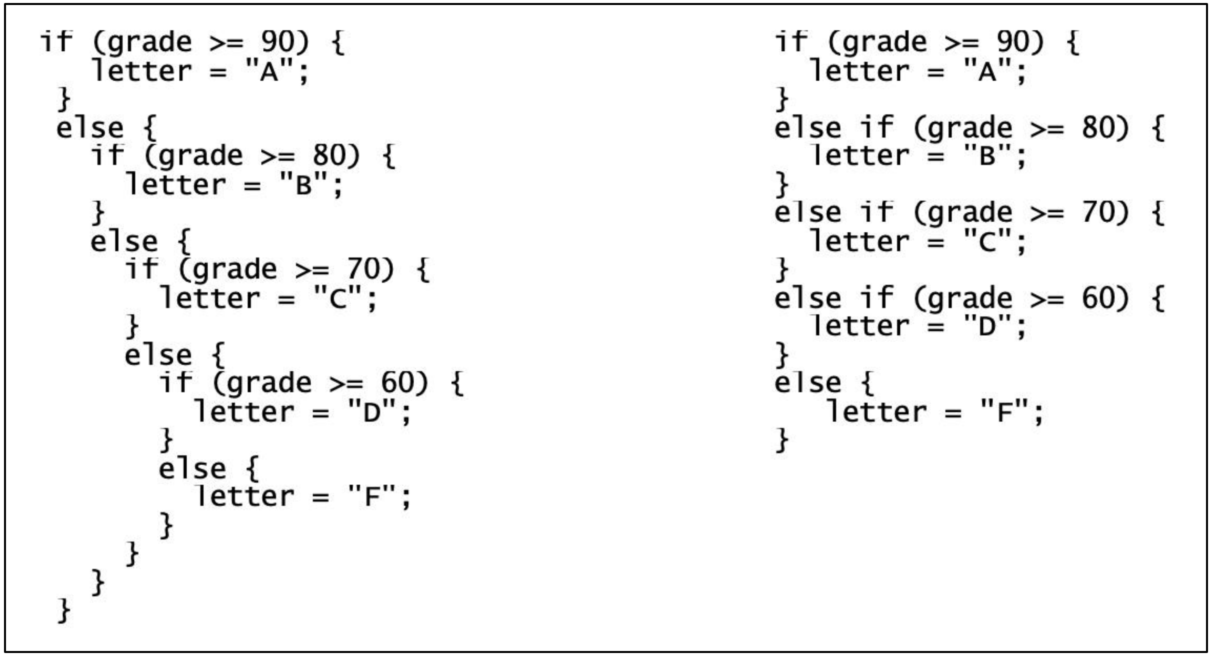
```
if (grade >= 90) {
  alert('Congratulations - you earned an A!');
}
else {
  if (grade >= 60) {
    alert('You passed, but could do better.');
  }
  else {
    alert('Sorry, you failed');
  }
}
```

A Cleaner Notation

when it is necessary to handle a large number of alternatives, nested if-else structures can become unwieldy

- multiple levels of indentation and curly braces cause the code to look cluttered make it harder to read/understand
- can simplify by removing some unnecessary curly braces & aligning each case to the left

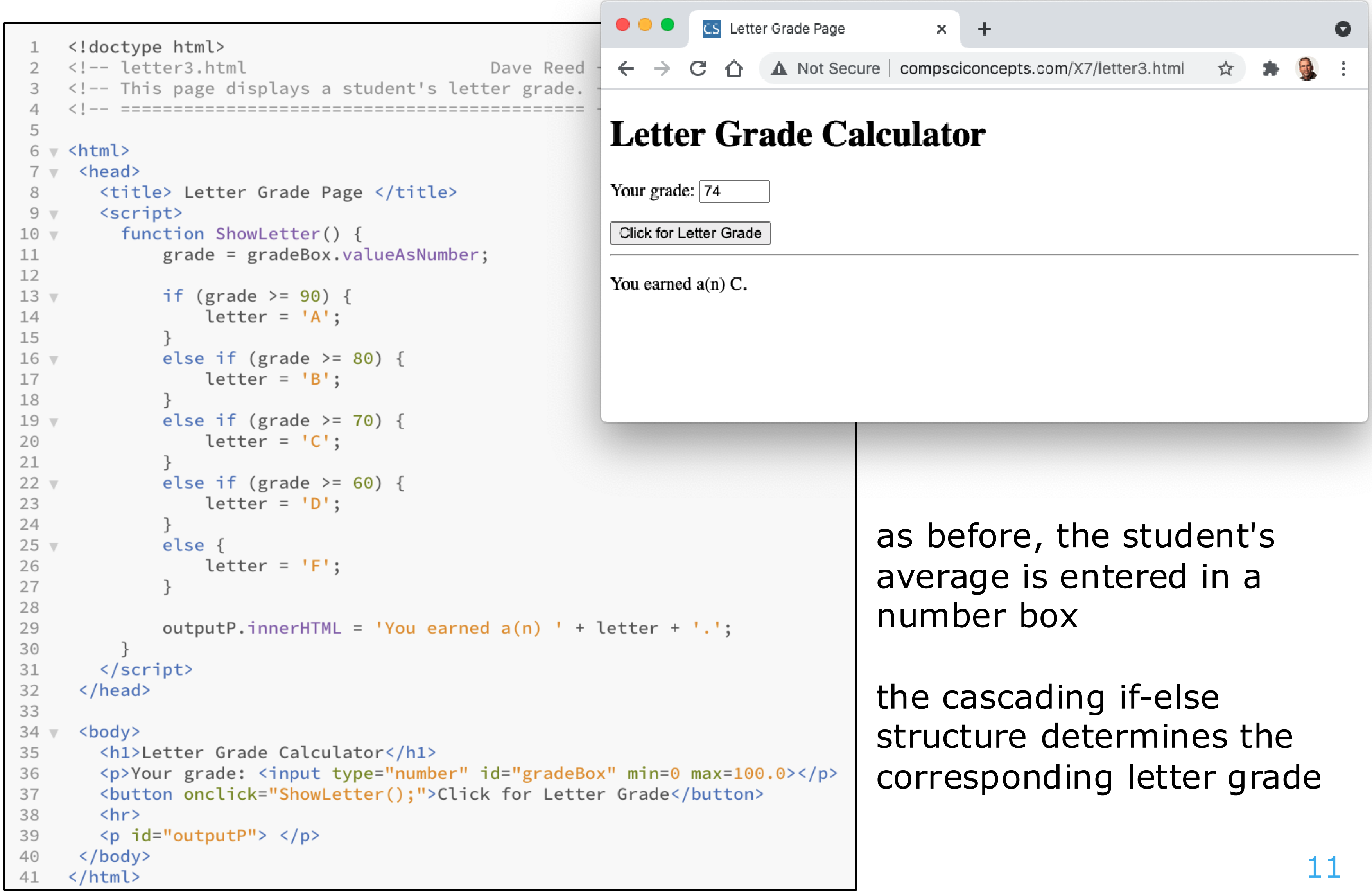
nested if statements vs. more readable if-else



```
if (grade >= 90) {
   letter = "A";
 }
 else {
   if (grade >= 80) {
     letter = "B";
   }
   else {
     if (grade >= 70) {
       letter = "C";
     }
     else {
       if (grade >= 60) {
         letter = "D";
       }
       else {
         letter = "F";
       }
     }
   }
 }
```

```
if (grade >= 90) {
  letter = "A";
}
else if (grade >= 80) {
  letter = "B";
}
else if (grade >= 70) {
  letter = "C";
}
else if (grade >= 60) {
  letter = "D";
}
else {
   letter = "F";
}
```

Letter Grade (cont.)



```
<!doctype html>
<!-- letter3.html                         Dave Reed -
<!-- This page displays a student's letter grade. -
<!-- ============================================ -

<html>
 <head>
   <title> Letter Grade Page </title>
   <script>
     function ShowLetter() {
         grade = gradeBox.valueAsNumber;

         if (grade >= 90) {
             letter = 'A';
         }
         else if (grade >= 80) {
             letter = 'B';
         }
         else if (grade >= 70) {
             letter = 'C';
         }
         else if (grade >= 60) {
             letter = 'D';
         }
         else {
             letter = 'F';
         }

         outputP.innerHTML = 'You earned a(n) ' + letter + '.';
     }
   </script>
 </head>

 <body>
   <h1>Letter Grade Calculator</h1>
   <p>Your grade: <input type="number" id="gradeBox" min=0 max=100.0></p>
   <button onclick="ShowLetter();">Click for Letter Grade</button>
   <hr>
   <p id="outputP"> </p>
 </body>
</html>
```

Letter Grade Page

Not Secure | compsciconcepts.com/X7/letter3.html

Letter Grade Calculator

Your grade: 74

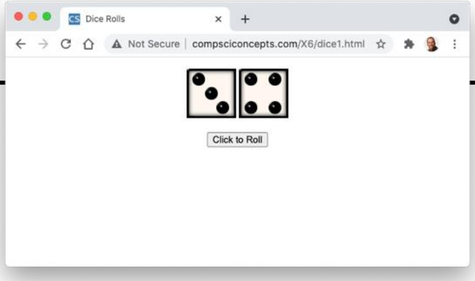
Click for Letter Grade

You earned a(n) C.

as before, the student's average is entered in a number box

the cascading if-else structure determines the corresponding letter grade

Embedded Counters



recall the dice pages from Chapter X6

suppose we wanted to keep a count of the number of rolls

- we could use a span element to store the count (initially 0)

```
Number of rolls: <span id="rollSpan">0</span>.
```

- for each roll, access the count, add 1, and reassign

```
rollSpan.innerHTML = rollSpan.innerHTML + 1;        // DOES NOT WORK
```

problem: `innerHTML` always evaluates to a string

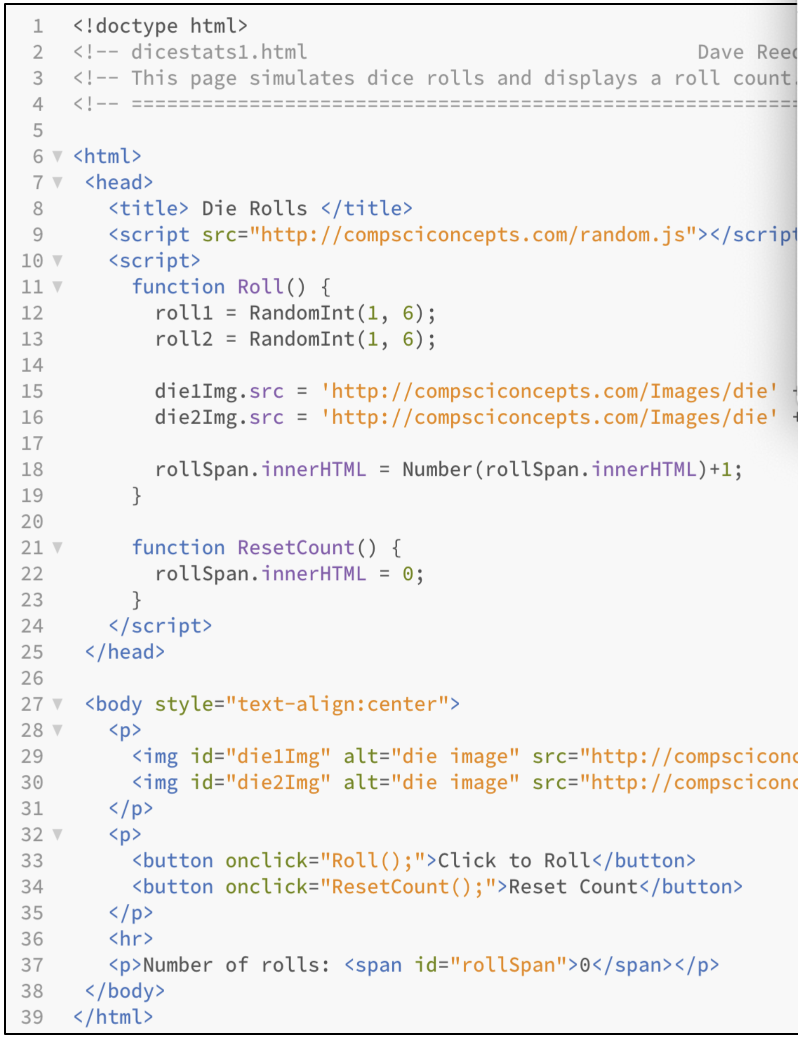
- `rollSpan.innerHTML + 1 = '0' + 1 = '0' + '1' = '01'`   (see Ch. X5)

we avoided this problem with number boxes using `valueAsNumber`

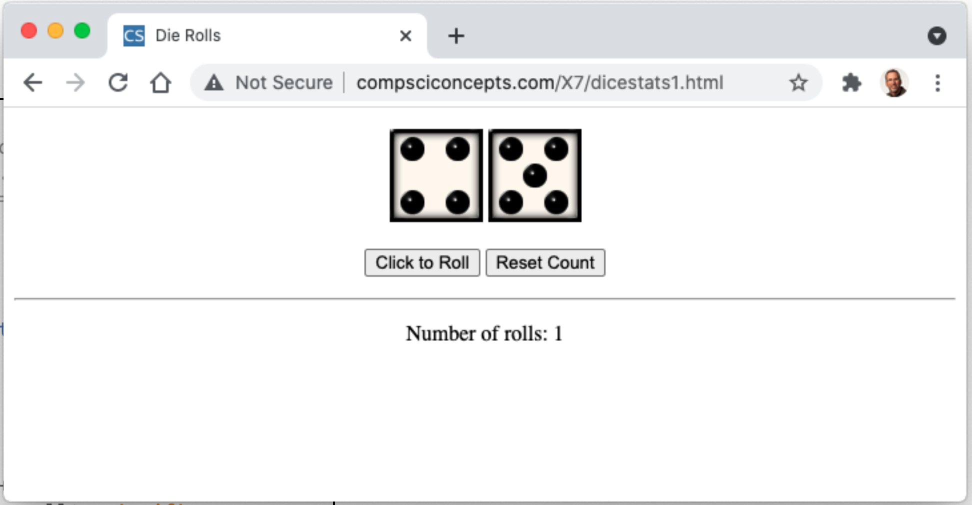
- unfortunately, there is no equivalent attribute for a span
- fortunately, there is a function that will convert a string into its equivalent number value

```
rollSpan.innerHTML = Number(rollSpan.innerHTML) + 1;      // THIS WORKS!
```

Dice Stats (v.1)



```
<!doctype html>
<!-- dicestats1.html                                 Dave Reed
<!-- This page simulates dice rolls and displays a roll count.
<!-- ==========================================================

<html>
 <head>
   <title> Die Rolls </title>
   <script src="http://compsciconcepts.com/random.js"></script
   <script>
     function Roll() {
       roll1 = RandomInt(1, 6);
       roll2 = RandomInt(1, 6);

       die1Img.src = 'http://compsciconcepts.com/Images/die' +
       die2Img.src = 'http://compsciconcepts.com/Images/die' + roll2 + '.gif';

       rollSpan.innerHTML = Number(rollSpan.innerHTML)+1;
     }

     function ResetCount() {
       rollSpan.innerHTML = 0;
     }
   </script>
 </head>

 <body style="text-align:center">
   <p>
     <img id="die1Img" alt="die image" src="http://compsciconcepts.com/Images/die3.gif">
     <img id="die2Img" alt="die image" src="http://compsciconcepts.com/Images/die4.gif">
   </p>
   <p>
     <button onclick="Roll();">Click to Roll</button>
     <button onclick="ResetCount();">Reset Count</button>
   </p>
   <hr>
   <p>Number of rolls: <span id="rollSpan">0</span></p>
 </body>
</html>
```



Number of rolls: 1

recall the pages from Ch X6 that simulated dice rolls

add an embedded counter

- initially, the span contains 0
- its contents are incremented each time function is called

also add a function to reset the counter

Conditional Counters

counters can be combined with if statements to count *conditional events*

- e.g., to count doubles

```
if (roll1 == roll2) {
  doubleSpan.innerHTML = Number(doubleSpan.innerHTML) + 1;
}
```

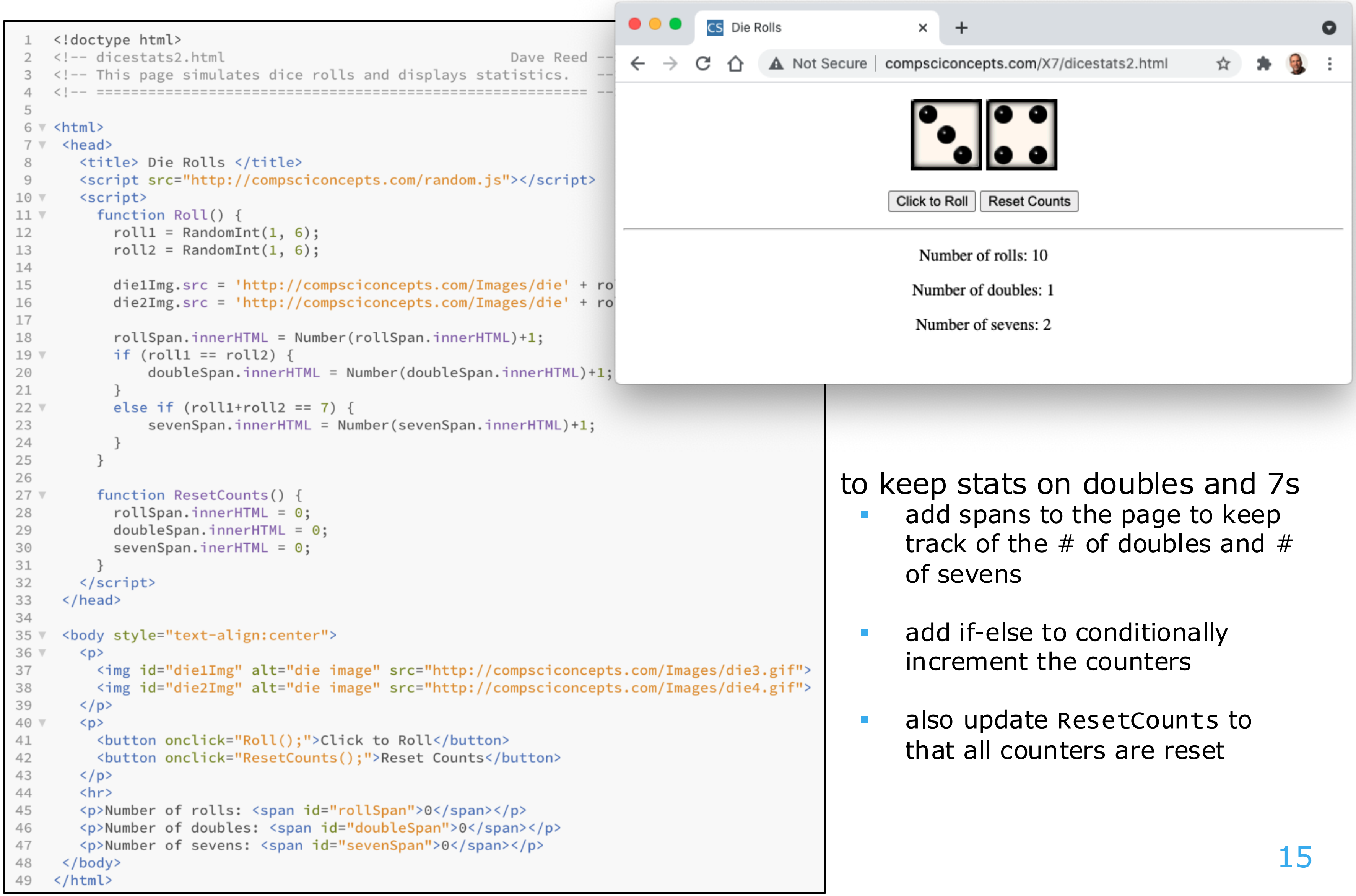
- e.g., to count sevens

```
if (roll1+roll2 == 7) {
  sevenSpan.innerHTML = Number(sevenSpan.innerHTML) + 1;
}
```

since doubles and sevens are mutually exclusive, could even combine

```
if (roll1 == roll2) {
  doubleSpan.innerHTML = Number(doubleSpan.innerHTML) + 1;
}
else if (roll1+roll2 == 7) {
  sevenSpan.innerHTML = Number(sevenSpan.innerHTML) + 1;
}
```

Dice Stats (v.2)



```
<!doctype html>
<!-- dicestats2.html                                  Dave Reed --
<!-- This page simulates dice rolls and displays statistics.   --
<!-- ========================================================= --

<html>
 <head>
   <title> Die Rolls </title>
   <script src="http://compsciconcepts.com/random.js"></script>
   <script>
     function Roll() {
       roll1 = RandomInt(1, 6);
       roll2 = RandomInt(1, 6);

       die1Img.src = 'http://compsciconcepts.com/Images/die' + ro
       die2Img.src = 'http://compsciconcepts.com/Images/die' + ro

       rollSpan.innerHTML = Number(rollSpan.innerHTML)+1;
       if (roll1 == roll2) {
           doubleSpan.innerHTML = Number(doubleSpan.innerHTML)+1;
       }
       else if (roll1+roll2 == 7) {
           sevenSpan.innerHTML = Number(sevenSpan.innerHTML)+1;
       }
     }

     function ResetCounts() {
       rollSpan.innerHTML = 0;
       doubleSpan.innerHTML = 0;
       sevenSpan.inerHTML = 0;
     }
   </script>
 </head>

 <body style="text-align:center">
   <p>
     <img id="die1Img" alt="die image" src="http://compsciconcepts.com/Images/die3.gif">
     <img id="die2Img" alt="die image" src="http://compsciconcepts.com/Images/die4.gif">
   </p>
   <p>
     <button onclick="Roll();">Click to Roll</button>
     <button onclick="ResetCounts();">Reset Counts</button>
   </p>
   <hr>
   <p>Number of rolls: <span id="rollSpan">0</span></p>
   <p>Number of doubles: <span id="doubleSpan">0</span></p>
   <p>Number of sevens: <span id="sevenSpan">0</span></p>
 </body>
</html>
```

Number of rolls: 10

Number of doubles: 1

Number of sevens: 2

to keep stats on doubles and 7s

- add spans to the page to keep track of the # of doubles and # of sevens
- add if-else to conditionally increment the counters
- also update `ResetCounts` to that all counters are reset

Boolean Expressions

sometimes, simple comparisons between two values may not be adequate to express the conditions under which code should execute

complex Boolean expressions can be built using logical connectives

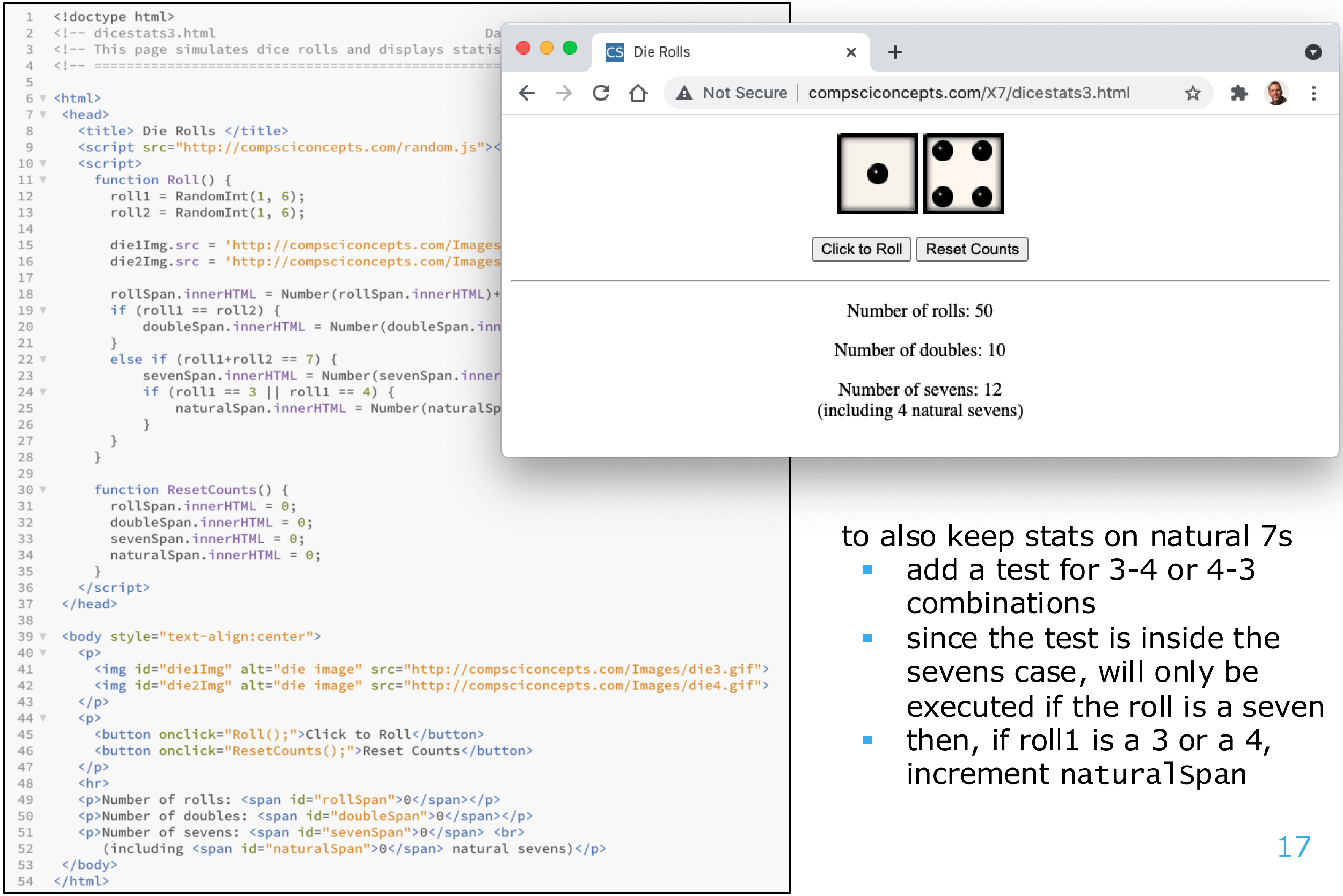
TEST1 && TEST2 evaluates to true if TEST1 AND TEST2 are true

TEST1 || TEST2 evaluates to true if TEST1 OR TEST2 arise true

! TEST evaluates to true if TEST is NOT true

```
if (roll1 == 4 && roll2 == 4) {
  CODE_TO_BE_EXECUTED_IF_4-4_COMBINATION_IS_ROLLED
}

if (roll1 == 4 || roll2 == 4) {
  CODE_TO_BE_EXECUTED_IF_EITHER_ROLL_IS_4
}

if (!(roll1 == 4 && roll2 == 4)) {
  CODE_TO_BE_EXECUTED_ IF_4-4_COMBINATION_IS_NOT_ROLLED
}
```

Dice Stats (v.3)



```
<!doctype html>
<!-- dicestats3.html                                      Da
<!-- This page simulates dice rolls and displays statis
<!-- ==================================================

<html>
 <head>
   <title> Die Rolls </title>
   <script src="http://compsciconcepts.com/random.js"><
   <script>
     function Roll() {
       roll1 = RandomInt(1, 6);
       roll2 = RandomInt(1, 6);

       die1Img.src = 'http://compsciconcepts.com/Images
       die2Img.src = 'http://compsciconcepts.com/Images

       rollSpan.innerHTML = Number(rollSpan.innerHTML)+
       if (roll1 == roll2) {
           doubleSpan.innerHTML = Number(doubleSpan.inn
       }
       else if (roll1+roll2 == 7) {
           sevenSpan.innerHTML = Number(sevenSpan.inner
           if (roll1 == 3 || roll1 == 4) {
               naturalSpan.innerHTML = Number(naturalSp
           }
       }
     }

     function ResetCounts() {
       rollSpan.innerHTML = 0;
       doubleSpan.innerHTML = 0;
       sevenSpan.innerHTML = 0;
       naturalSpan.innerHTML = 0;
     }
   </script>
 </head>

 <body style="text-align:center">
   <p>
     <img id="die1Img" alt="die image" src="http://compsciconcepts.com/Images/die3.gif">
     <img id="die2Img" alt="die image" src="http://compsciconcepts.com/Images/die4.gif">
   </p>
   <p>
     <button onclick="Roll();">Click to Roll</button>
     <button onclick="ResetCounts();">Reset Counts</button>
   </p>
   <hr>
   <p>Number of rolls: <span id="rollSpan">0</span></p>
   <p>Number of doubles: <span id="doubleSpan">0</span></p>
   <p>Number of sevens: <span id="sevenSpan">0</span> <br>
      (including <span id="naturalSpan">0</span> natural sevens)</p>
 </body>
</html>
```

Number of rolls: 50

Number of doubles: 10

Number of sevens: 12
(including 4 natural sevens)

to also keep stats on natural 7s

- add a test for 3-4 or 4-3 combinations
- since the test is inside the sevens case, will only be executed if the roll is a seven
- then, if roll1 is a 3 or a 4, increment `naturalSpan`

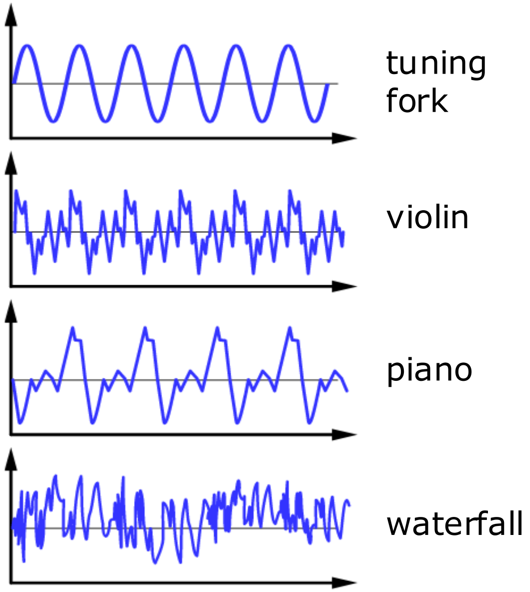
Representing Sounds

computers are capable of representing much more than numbers and text

- complex data requires additional techniques and algorithms

EXAMPLE: representing sounds

- sounds are inherently analog values with a specific amplitudes and frequencies
- when sound waves reach your ear, they cause your eardrum to vibrate, and your brain interprets the vibration as sound
- e.g. telephones translate a waveform into electrical signals, which are then sent over a wire and converted back to sound
- e.g. phonographs interpret waveforms stored on on grooves of a disk (similar to audio cassettes)
- analog values cannot be reproduced exactly, but this is not usually a problem since the human ear is unlikely to notice small inconsistencies

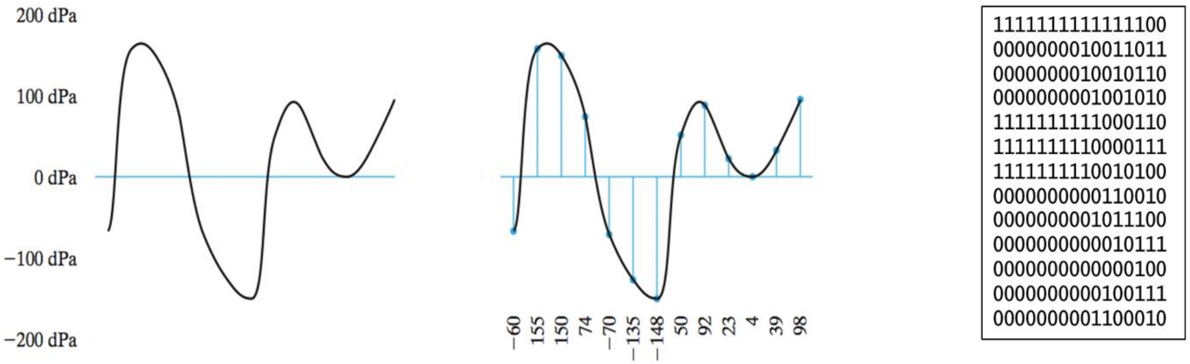


Representing Sounds (cont.)

when analog recordings are repeatedly duplicated, small errors that were originally unnoticed begin to propagate

digital recordings can be reproduced exactly without any deterioration in sound quality

- analog waveforms must be converted to a sequence of discrete values
- *digital sampling* is the process in which the amplitude of a wave is measured at regular intervals, and stored as discrete measurements



```
1111111111111100
0000000010011011
0000000010010110
0000000001001010
1111111111000110
1111111110000111
1111111110010100
0000000000110010
0000000001011100
0000000000010111
0000000000000100
0000000000100111
0000000001100010
```

start with an analog signal (sound wave, measured in dPa of pressure)

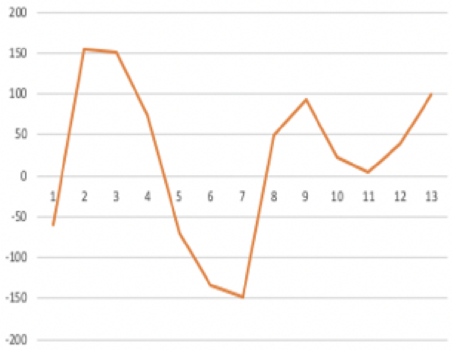
measure the wave amplitude at regular intervals (CD = 44,100/sec)

store the numbers in a file (using 16 bits or 2 bytes per number)

Representing Sounds (cont.)

to play a digital recording, a music player must extract the numbers and reconstruct the analog waveform

```
1111111111111100
0000000010011011
0000000010010110
0000000001001010
1111111111000110
1111111110000111
1111111110010100
0000000000110010
0000000001011100
0000000000010111
0000000000000100
0000000000100111
0000000001100010
```



your music player (e.g., phone) extracts the measurement numbers from the digital file.

it attempts to reconstruct the continuous waveform (not perfectly, but good enough)

since there are gaps between measurements, the reconstruction will not be an exact reproduction of the original

- CD takes 44,100 measurements/second, so gaps are small and the resulting sound is high-quality

Sound compression

CD-quality sound requires significant storage

consider a 3-minute, stereo recording (meaning two separate tracks)

- assume each measurement is stored using 2 bytes

3 min x 60 sec/min x 2 tracks x 44,100 nums/sec x 2 bytes/num = 31.752 MB

since the typical music album contains 10-20 songs, CD storage requires:
317-634 MB of space

- this will fit on a 700 MB disk, but would quickly overwhelm smartphones and portable music players

also, file size makes downloading or streaming music problematic

4G wireless connection can typically download up to 1.5 MB/second

- to download 3 min song: 31.752 MB / 1.5 MB/sec = 21 seconds
- to download 20 song album: 634 MB / 1.5 MB/sec = 7 minutes

Sound compression (cont.)

fortunately, techniques have been developed to reduce file size

- e.g., filter out sounds beyond the range of human hearing
- e.g., recognize when one track masks another
- e.g., possibly even simplify the waveform

the MP3 format (introduced in 1993) uses techniques such as these to reduce file size by 75-95%

- 3 minute song: 31.752 x 0.05 = 1.6 MB = 1 second using G4
- 20 song album = 634 MB x 0.05 = 31 MB = 20 seconds using G4

MP3 is a ***lossy*** format

- to achieve that level of compression, it makes simplifications and loses some of the information in the waveform
- this results in lower sound quality when played

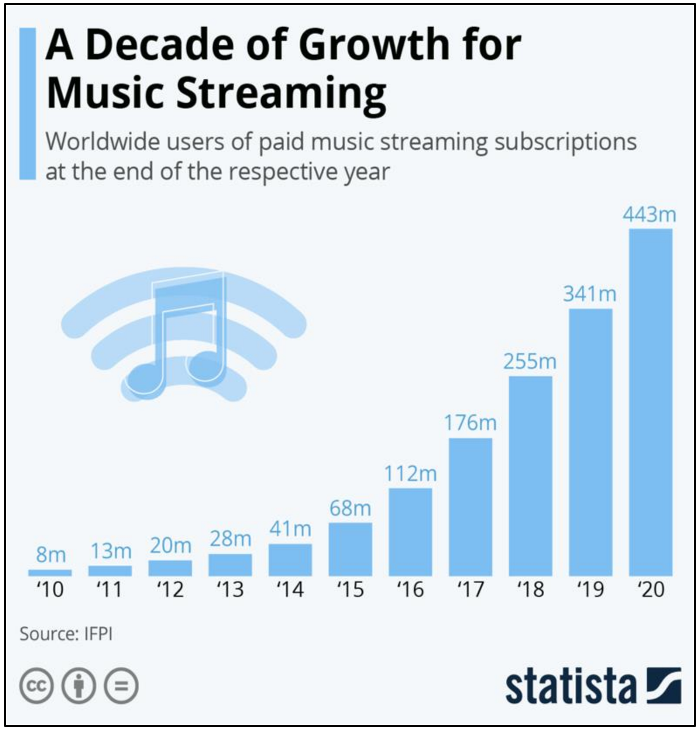
Impact on the music industry

before the CD in 1982, music piracy was almost non-existent

- few people had the technology to copy a phonograph record
- recording a cassette was easy, but the sound quality was bad
- CDs changed the game
  - cheap devices for making perfect, digital copies were readily available
  - this panicked the music industry, led to MANY piracy prosecutions

before the MP3 format in 1993,
streaming music was not feasible

- now, phones & portable players are the primary music source for many
- other digital audio formats have followed, e.g., WAV, AIFF, M4P.



A Decade of Growth for Music Streaming

Worldwide users of paid music streaming subscriptions at the end of the respective year

Source: IFPI

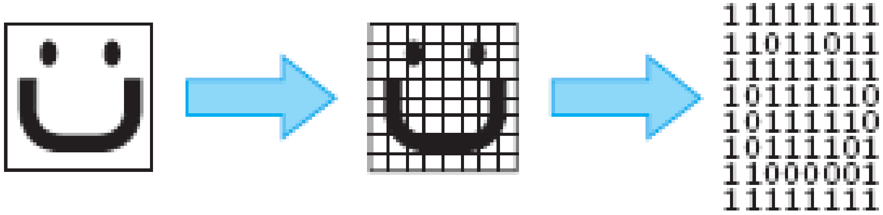
Representing Images

like sound, images are inherently analog

- real-world colors come in an infinite variety of shades
- film photography creates an analog representation using light-sensitive chemicals

like sound, techniques exist for digitizing images

- the simplest involves partitioning the image into a grid of picture elements (*pixels)* and then converting each pixel into a bit pattern
- the digital representation is known as a *bitmap*



```
11111111
11011011
11111111
10111110
10111110
10111101
11000001
11111111
```

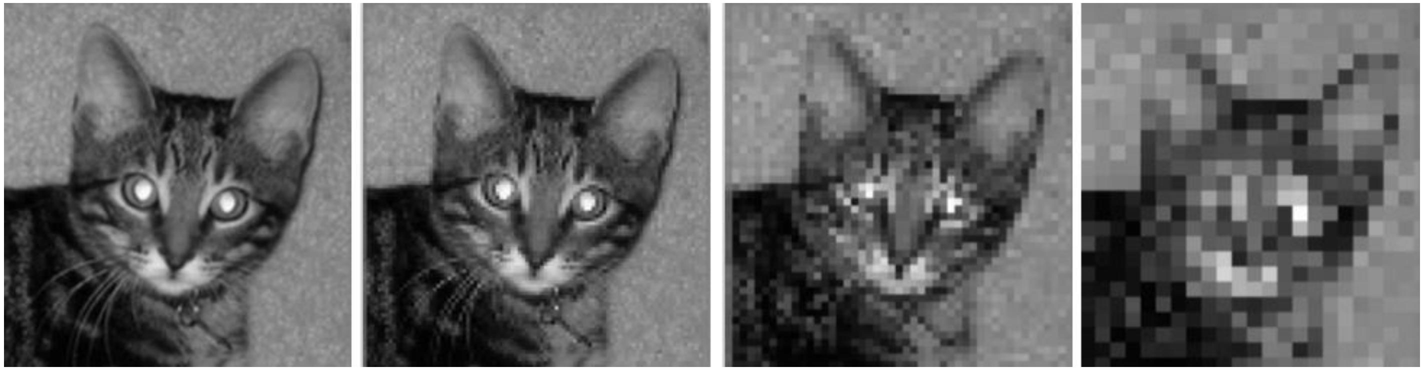
1. To generate a bitmap, the image is first partitioned into a grid of pixels, here 8 × 8.

2. For a black-and-white bitmap, a black pixel is represented with a 0 bit, a white pixel with a 1 bit.

Image resolution

resolution refers to the sharpness or clarity of an image

- bitmaps that are divided into smaller pixels will yield higher resolution images
- the left image is stored using 72 pixels per square inch, each subsequent image has half the resolution



this is a *grayscale* image

- each pixel is a shade of gray, somewhere between black and white
- most image formats use 8 bits for a grayscale pixel → 256 shades of gray
- as a result, grayscale requires 8x storage of black-and-white

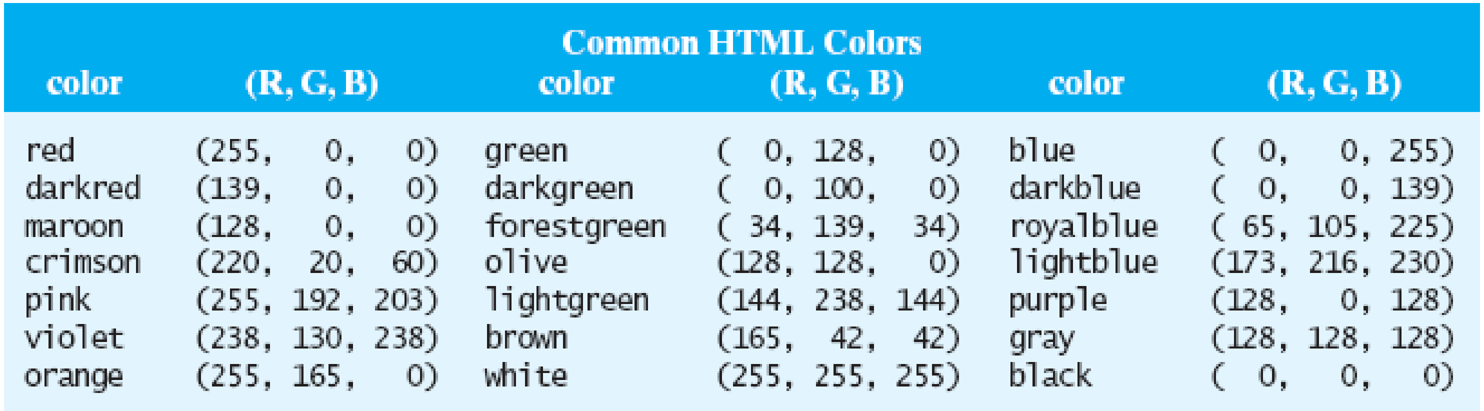
Color images

for color images, can break each color into its red, green & blue components

- RGB value is a triple, listing the intensities of red, green & blue on a 0-255 scale
- 256^3 = 16,777.216 different color combinations can be represented

- each component requires 8 bits, so a total of 24 bits per pixel

→ color images require 3x storage of grayscale, 24x the storage of black-and-white



Common HTML Colors

color	(R, G, B)	color	(R, G, B)	color	(R, G, B)
red	(255, 0, 0)	green	(0, 128, 0)	blue	(0, 0, 255)
darkred	(139, 0, 0)	darkgreen	(0, 100, 0)	darkblue	(0, 0, 139)
maroon	(128, 0, 0)	forestgreen	(34, 139, 34)	royalblue	(65, 105, 225)
crimson	(220, 20, 60)	olive	(128, 128, 0)	lightblue	(173, 216, 230)
pink	(255, 192, 203)	lightgreen	(144, 238, 144)	purple	(128, 0, 128)
violet	(238, 130, 238)	brown	(165, 42, 42)	gray	(128, 128, 128)
orange	(255, 165, 0)	white	(255, 255, 255)	black	(0, 0, 0)

experiment with http://compsciconcepts.com/C9/rgb.html

Image compression

color bitmaps can be extremely large

- 12 megapixel image → 12 million pixels → 36 MB of storage

common image formats implement various compression techniques to reduce storage size

- GIF (Graphics Interchange Format)
  - a *lossless* format, meaning no information is lost in the compression
  - commonly used for precise pictures, such as line drawings
  - 25-50% reduction possible
- PNG(Portable Network Graphics)
  - more modern, lossless alternative to GIF - more colors
  - 10-50% smaller than GIF (so 33-75% reduction from bitmap)
- JPEG (Joint Photographic Experts Group)
  - a *lossy* format, so the compression is not fully reversible (but more efficient)
  - commonly used for photographs
  - 90-95% reduction possible

image formats also embed metadata (date, location, source, …) in images

- can be useful for tasks such as organizing a photo gallery by data or geography

Steganography

an interesting side topic related to images is steganography

- the practice of hiding a secret message in plain sight (e.g., within another message or object)

1. Take your secret message an encode it as bits (using ASCII/Unicode codes): $b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8\ b_9$ ...
2. Take an image and break it into its RGB pixels: (R_1,G_1,B_1) (R_2,G_2,B_2) (R_3,G_3,B_3) …
3. For each bit b_i in the message, possibly change the B_i component of the corresponding pixel so that:
   - if b_i is even, then B_i is also even (add 1 to B_i if necessary)
   - if b_i is odd, then B_i is also odd (add 1 to B_i if necessary)

the resulting image will look the same to the human eye

- e.g., most people can't distinguish between (20, 50, 100) and (20, 50, 101)

but, if you know to look, you can extract the message bits from the pixels

Example:

suppose the message starts with 'M' (whose ASCII value is 01001101)
and the image starts with pixels:

(100,200,50) (100,200,50) (100,202,52) (98,203,53) (88,190,48) (88,188,47) (86,180,40) (80,160,43)

then, embed the message bits in the B components of the pixels:

(100,200,50) (100,200,51) (100,202,52) (98,203,54) (88,190,49) (88,188,47) (86,180,40) (80,160,43)

to an unsuspecting viewer, the image will look normal
to a person who knows to look, the message bits can easily be extracted

(100,200,50)	(100,200,51)	(100,202,52)	(98,203,54)	(88,190,49)	(88,188,47)	(86,180,40)	(80,160,43)
⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
0	1	0	0	1	1	0	1

Representing movies

in principle, a movie is a sequence of images (frames) that are displayed in sequence to produce the effect of motion

- typically, 24 frames/sec

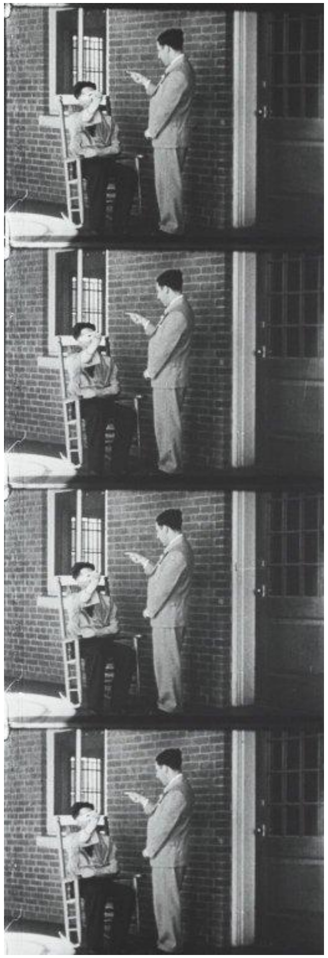
MPEG or MP4 format uses a variety of techniques to compress video

- individual frames use techniques similar to JPEG
- since much of successive frames are same, need only store changes from frame to frame

elements of MPEG are included in the ATSC (Advanced Television Systems Committee) standard for digital TV

- individual frames use techniques similar to JPEG

other related formats are DVD & Blu-Ray



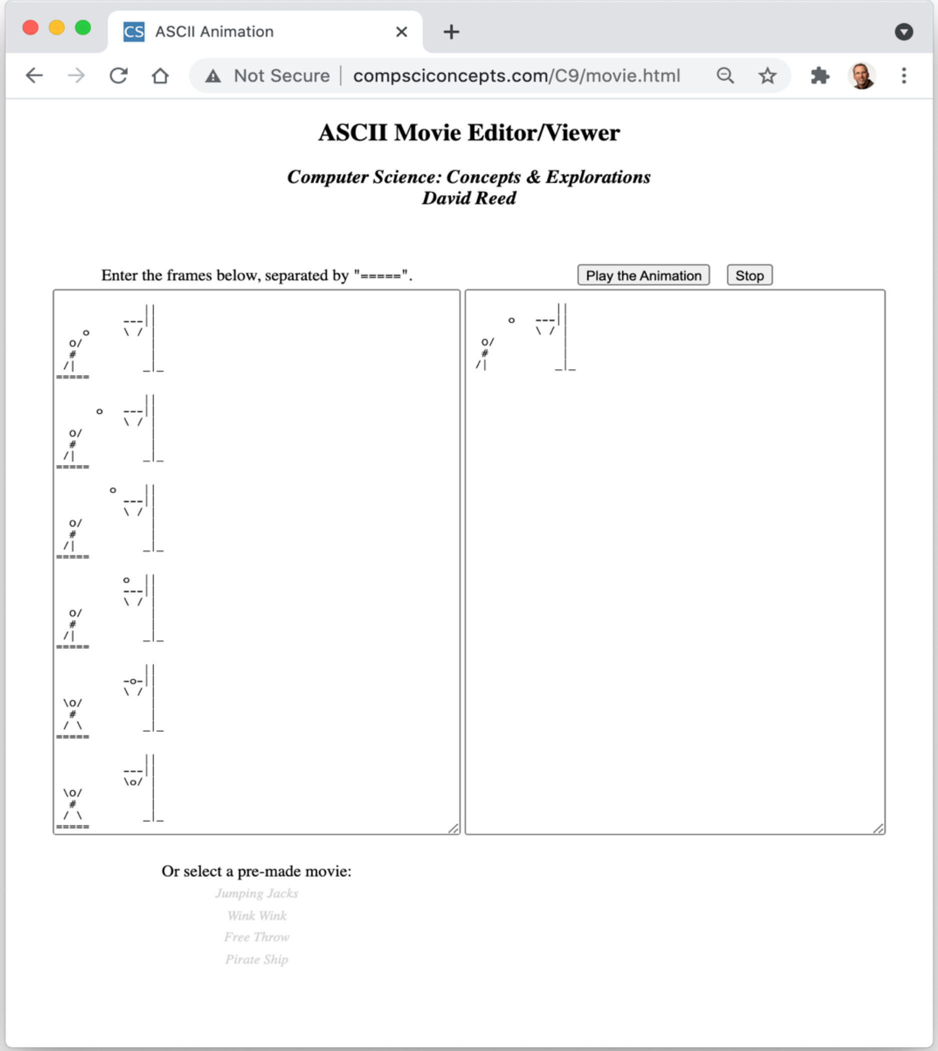
ASCII movies

for a fun and creative exercise, make your own movies

each frame is "drawn" using characters from the keyboard

frames are separated using =====

the button will "play" the movie, 5 frames/second



ASCII Movie Editor/Viewer

Computer Science: Concepts & Explorations
David Reed

Enter the frames below, separated by "=====".

```
             ||
          ---||
    o     \ / |
  o/          |
  #           |
 /|         _|_
=====
             ||
      o   ---||
          \ / |
  o/          |
  #           |
 /|         _|_
=====
        o    ||
          ---||
          \ / |
  o/          |
  #           |
 /|         _|_
=====
          o  ||
          ---||
          \ / |
  o/          |
  #           |
 /|         _|_
=====
             ||
          -o-||
          \ / |
 \o/          |
  #           |
 / \        _|_
=====
             ||
          ---||
          \o/ |
 \o/          |
  #           |
 / \        _|_
=====
```

```
             ||
      o   ---||
          \ / |
  o/          |
  #           |
 /|         _|_
```

Or select a pre-made movie:
Jumping Jacks
Wink Wink
Free Throw
Pirate Ship

Software Models

when studying complex systems (e.g., weather, stock markets, voting),
software models provide a fast & cost-effective tool for gaining insights

- computer simulations can be much faster than real-world counterparts
  e.g., can simulate centuries of climate change in seconds
- can be much cheaper to simulate than to study the real-world system
  e.g., can try different investment strategies without risking real money
- can control for parameters that are difficult to obtain reliably in real-world
  e.g., can speculate on how changes in demographics would affect an election

we will consider 2 examples of modeling real-world systems

1. disease spread
2. volleyball game

software models always make simplifying assumptions

- must always be careful to examine the model and make sure that the simplifications do not invalidate conclusions drawn from the simulations

Disease Spread

when there is a disease outbreak (e.g., COVID-19), the CDC and other health organizations build software models to

- better understand and warn the public about the disease
- study potential interventions (e.g., social distancing, vaccines)

these models can be very complex, taking into account:

- transmission method, patient incubation period, population density, …

a very simple model uses the *basic reproduction number*, R_0: the number of people that a patient will infect over the course of their infection

- if $R_0 < 1$, then each patient infects fewer than 1 other (on average), so the disease will eventually die out
- if $R_0 = 1$, then each patient infects exactly 1 other (on average), so the disease spreads steadily (known as an *endemic*)
- if $R_0 > 1$, then each patient infects more than 1 other (on average), so the disease spreads exponentially (known as an *epidemic* or *pandemic*)

R_0 Example

suppose a disease has $R_0 = 2$, 100 people infected:

1. in 1st wave, those 100 patients infect 200 (2 per person on average)
2. in 2nd wave, the 200 patients from wave 1 infect 400
3. in 3rd wave, the 400 patients from wave 2 infect 800

   . . .

10. in 10th wave, the 51,200 patients from wave 9 infect 102,400

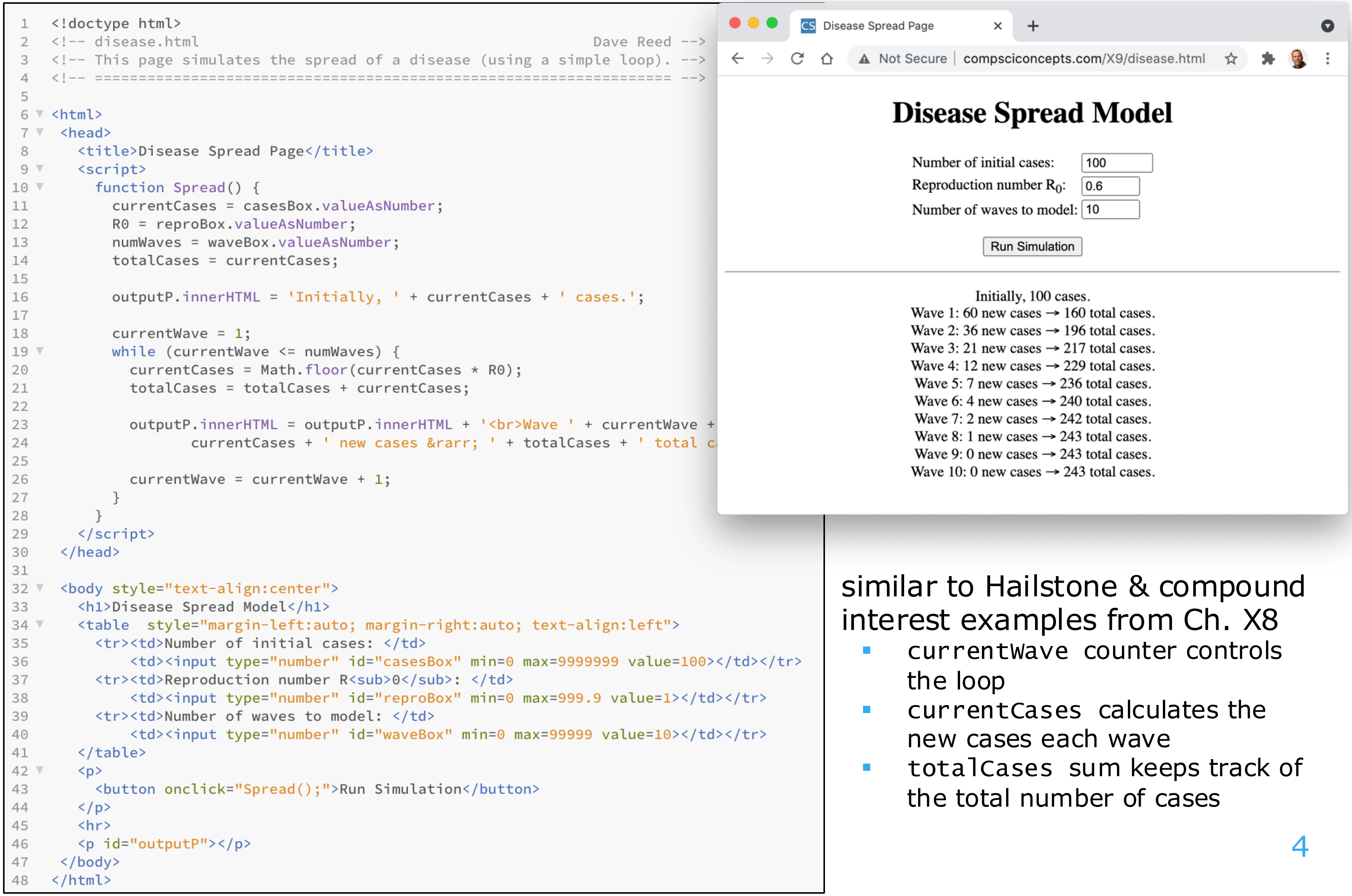
estimated R_0 for common diseases:

seasonal flu	$R_0 \approx$ 0.9-2.1
Smallpox	$R_0 \approx 7$
Polio	$R_0 \approx 7$
Measles	$R_0 \approx 18$
COVID-19 in U.S., March 2020	$R_0 \approx 4$
COVID-19 in U.S., October 2020	$R_0 \approx 2$
COVID-19 in U.S., March 2021	$R_0 \approx 0.9$

fortunately, vaccines have been developed for many highly contagious diseases

note: R_0 is not fixed – medical treatments and behavior changes can affect it

Disease Spread Example



```
<!doctype html>
<!-- disease.html                                      Dave Reed -->
<!-- This page simulates the spread of a disease (using a simple loop). -->
<!-- ================================================================== -->

<html>
 <head>
   <title>Disease Spread Page</title>
   <script>
     function Spread() {
       currentCases = casesBox.valueAsNumber;
       R0 = reproBox.valueAsNumber;
       numWaves = waveBox.valueAsNumber;
       totalCases = currentCases;

       outputP.innerHTML = 'Initially, ' + currentCases + ' cases.';

       currentWave = 1;
       while (currentWave <= numWaves) {
         currentCases = Math.floor(currentCases * R0);
         totalCases = totalCases + currentCases;

         outputP.innerHTML = outputP.innerHTML + '<br>Wave ' + currentWave +
                currentCases + ' new cases &rarr; ' + totalCases + ' total c

         currentWave = currentWave + 1;
       }
     }
   </script>
 </head>

 <body style="text-align:center">
   <h1>Disease Spread Model</h1>
   <table  style="margin-left:auto; margin-right:auto; text-align:left">
     <tr><td>Number of initial cases: </td>
         <td><input type="number" id="casesBox" min=0 max=9999999 value=100></td></tr>
     <tr><td>Reproduction number R<sub>0</sub>: </td>
         <td><input type="number" id="reproBox" min=0 max=999.9 value=1></td></tr>
     <tr><td>Number of waves to model: </td>
         <td><input type="number" id="waveBox" min=0 max=99999 value=10></td></tr>
   </table>
   <p>
     <button onclick="Spread();">Run Simulation</button>
   </p>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```

Disease Spread Model

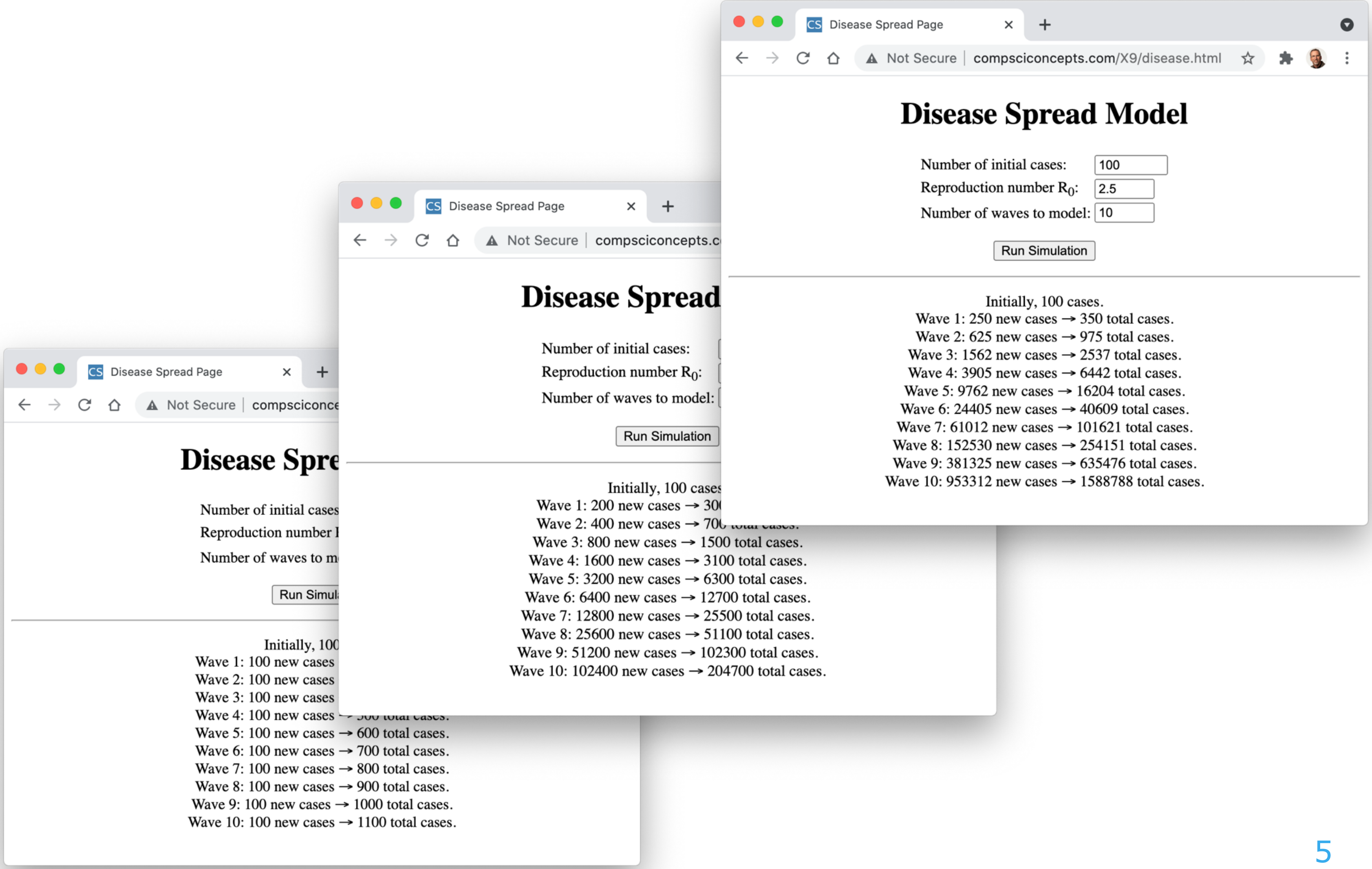
Number of initial cases:	100
Reproduction number R_0:	0.6
Number of waves to model:	10

```
Initially, 100 cases.
Wave 1: 60 new cases → 160 total cases.
Wave 2: 36 new cases → 196 total cases.
Wave 3: 21 new cases → 217 total cases.
Wave 4: 12 new cases → 229 total cases.
Wave 5: 7 new cases → 236 total cases.
Wave 6: 4 new cases → 240 total cases.
Wave 7: 2 new cases → 242 total cases.
Wave 8: 1 new cases → 243 total cases.
Wave 9: 0 new cases → 243 total cases.
Wave 10: 0 new cases → 243 total cases.
```

similar to Hailstone & compound interest examples from Ch. X8

- `currentWave` counter controls the loop
- `currentCases` calculates the new cases each wave
- `totalCases` sum keeps track of the total number of cases

Model Results



Disease Spre

Number of initial cases
Reproduction number I
Number of waves to m

```
Initially, 100
Wave 1: 100 new cases
Wave 2: 100 new cases
Wave 3: 100 new cases
Wave 4: 100 new cases → 500 total cases.
Wave 5: 100 new cases → 600 total cases.
Wave 6: 100 new cases → 700 total cases.
Wave 7: 100 new cases → 800 total cases.
Wave 8: 100 new cases → 900 total cases.
Wave 9: 100 new cases → 1000 total cases.
Wave 10: 100 new cases → 1100 total cases.
```

Disease Spread

Number of initial cases:
Reproduction number R_0:
Number of waves to model:

```
Initially, 100 cases
Wave 1: 200 new cases → 30
Wave 2: 400 new cases → 700 total cases.
Wave 3: 800 new cases → 1500 total cases.
Wave 4: 1600 new cases → 3100 total cases.
Wave 5: 3200 new cases → 6300 total cases.
Wave 6: 6400 new cases → 12700 total cases.
Wave 7: 12800 new cases → 25500 total cases.
Wave 8: 25600 new cases → 51100 total cases.
Wave 9: 51200 new cases → 102300 total cases.
Wave 10: 102400 new cases → 204700 total cases.
```

Disease Spread Model

Number of initial cases: 100
Reproduction number R_0: 2.5
Number of waves to model: 10

```
Initially, 100 cases.
Wave 1: 250 new cases → 350 total cases.
Wave 2: 625 new cases → 975 total cases.
Wave 3: 1562 new cases → 2537 total cases.
Wave 4: 3905 new cases → 6442 total cases.
Wave 5: 9762 new cases → 16204 total cases.
Wave 6: 24405 new cases → 40609 total cases.
Wave 7: 61012 new cases → 101621 total cases.
Wave 8: 152530 new cases → 254151 total cases.
Wave 9: 381325 new cases → 635476 total cases.
Wave 10: 953312 new cases → 1588788 total cases.
```

Modeling Tradeoffs

advantages:

- **CONVENIENCE:** software models can be executed anywhere sufficient computer power is available
- **SAFETY:** since software models run on a computer, they pose no risk to patients or researchers
- **SPEED:** the speed at which software models can be executed is only limited by the complexity of the model and the processing power of the computer
- **CUSTOMIZABILITY:** software models are typically built with parameters that can be adjusted to test different conditions
- **REPRODUCABILITY:** as they are fast and customizable, software models can be repeated to confirm results and compare outcomes under different conditions

disadvantages:

- **OVERSIMPLIFICATION:** by their very definition, software models simplify the systems they are modeling, potentially ignoring factors that may be important
- **INCORRECTNESS:** as with any software system, software models can contain errors that impact the results
- **OVERCONFIDENCE:** due to flaws in design or implementation, model can produce results that look reasonable but are incorrect or misleading

Nondeterministic Models

the R_0 model of disease spread is *deterministic*

- it follows rules that unambiguously determine the output
- i.e., same inputs will always produce the same output

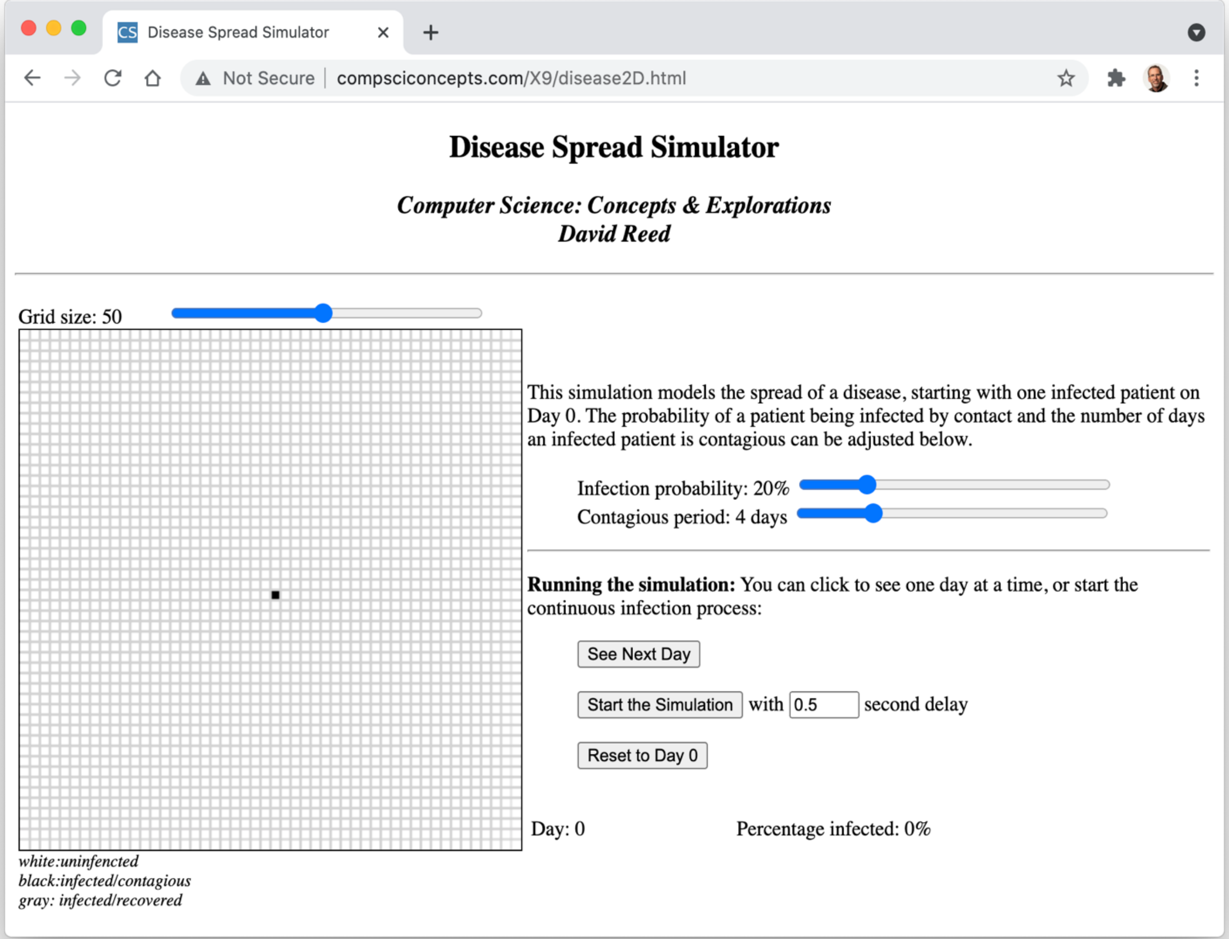
many real-world systems are too complex to model deterministically

- instead, they utilize probabilities to capture unpredictable features
- i.e., same input should produce similar but not necessarily identical results

consider a nondeterministic model of disease spread

- each day, an infectious patient has a certain probability of infecting those with whom they come in contact
- allows us to introduce more complex variables: locality & infectious period
- e.g., once infected, a patient will be contagious for 4 days & have 20% probability of infecting those with whom they come in contact

2-D Disease Spread



Disease Spread Simulator

Computer Science: Concepts & Explorations
David Reed

Grid size: 50

This simulation models the spread of a disease, starting with one infected patient on Day 0. The probability of a patient being infected by contact and the number of days an infected patient is contagious can be adjusted below.

Infection probability: 20%
Contagious period: 4 days

Running the simulation: You can click to see one day at a time, or start the continuous infection process:

with 0.5 second delay

Day: 0

Percentage infected: 0%

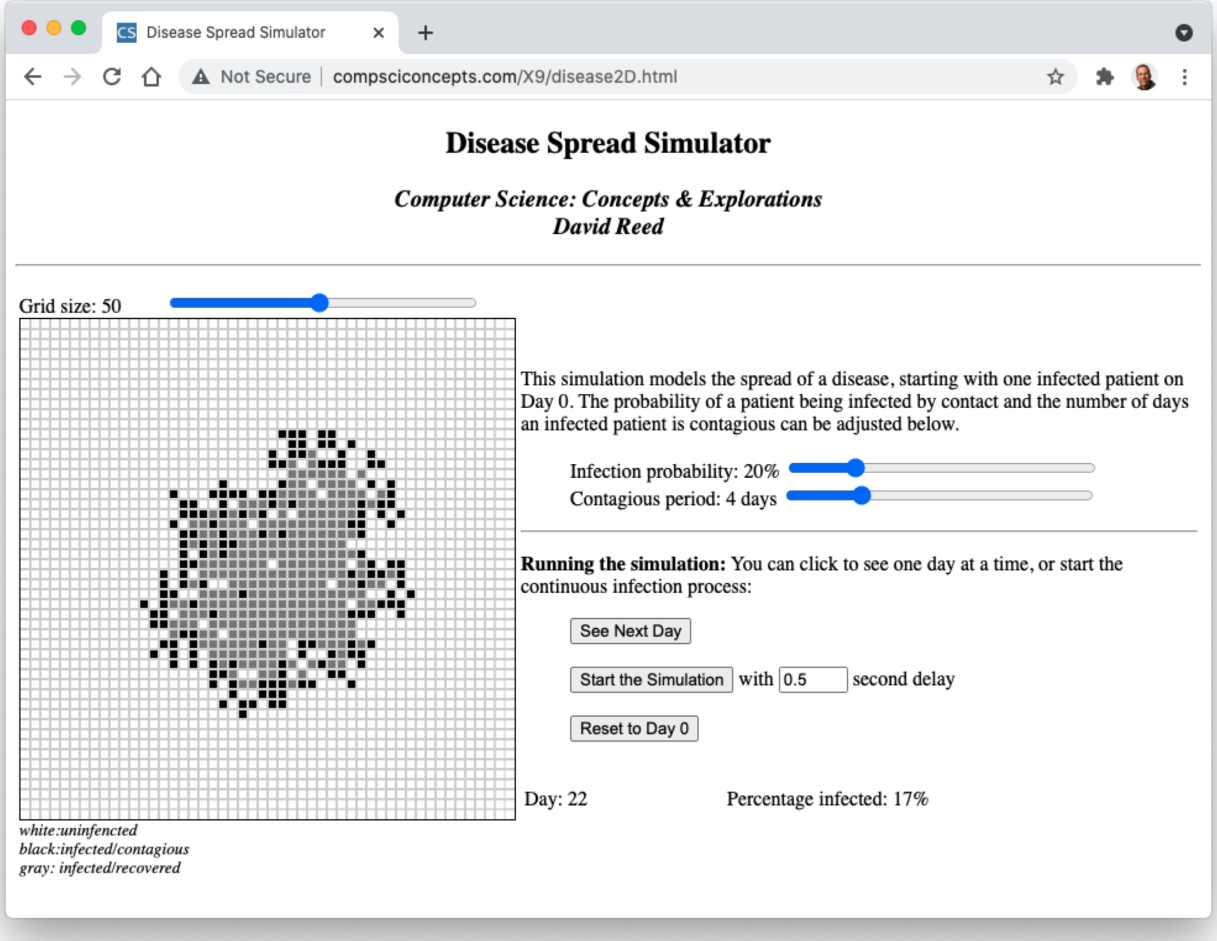
white:uninfencted
black:infected/contagious
gray: infected/recovered

the user can control
- the size of the grid
- infection probability
- contagious period

initially, there is a single infected person at the center of the grid

can click a button to see:
- a single day
- a series of days (with delays in between)

2-D Disease Spread



Disease Spread Simulator

Computer Science: Concepts & Explorations
David Reed

Grid size: 50

This simulation models the spread of a disease, starting with one infected patient on Day 0. The probability of a patient being infected by contact and the number of days an infected patient is contagious can be adjusted below.

Infection probability: 20%
Contagious period: 4 days

Running the simulation: You can click to see one day at a time, or start the continuous infection process:

with 0.5 second delay

Day: 22 Percentage infected: 17%

white:uninfected
black:infected/contagious
gray: infected/recovered

white square: uninfected person

black square: infect & contagious person

gray square: infected but no longer contagious

the simulation is highly sensitive to changes in the infection rate & contagious period

Volleyball Simulations

in 1998, FIVB switched the scoring system for international volleyball

- old system: *sideout scoring* – only serving team can win a point
- new system: *rally scoring* – a point is awarded on every rally

- the goal was to make games more exciting and more consistent in length

in conjunction, they extended games from 15 points to 25 points

- the new game length was intended to maintain competitive balance
  HOW DID THEY DETERMINE THE NEW GAME LENGTH?

for our model:

- assume each team has a ranking (1-100) that quantifies their strength
- by comparing team strengths, can determine the likelihood of winning a rally
  e.g., if team1 is 80 and team2 is 40, team1 twice as likely to win a given rally
- to simulate a game, repeatedly simulate points and keep score
  the game is over when a team reaches 25 points (must win by 2)

Simulation Details

the following is a general layout of the simulation

```
score1 = 0;
score2 = 0;

while (GAME_NOT_OVER) {
  DETERMINE_WINNER_OF_RALLY
  if (TEAM1_WON_RALLY) {
    score1 = score1 + 1;
  }
  else {
    score2 = score2 + 1;
  }

  DISPLAY_SCORE
}
```

need to figure out the loop condition

also need to figure out how to simulate a rally

Simulation Details

if we ignore the win-by-2 rule, the loop test is straightforward

```
while (score1 < 25 && score2 < 25) {
```

be careful with loop tests

- they are not stopping conditions, they are continuing conditions
- the game continues as long as both teams are under 25 points

the win-by-2 condition adds another possibility of continuing

- continue if both teams are under 25 points OR within 1 point of each other

```
while ((score1 < 25 && score2 < 25) ||
                (Math.abs(score1-score2) < 2)) {
```

Simulation Details

to simulate a single rally, we need to pick the winner probabilistically

- if team1 is X% better, they should be X% more likely to win any given rally

- we can simulate this using `RandomInt` (from the `random.js` library)

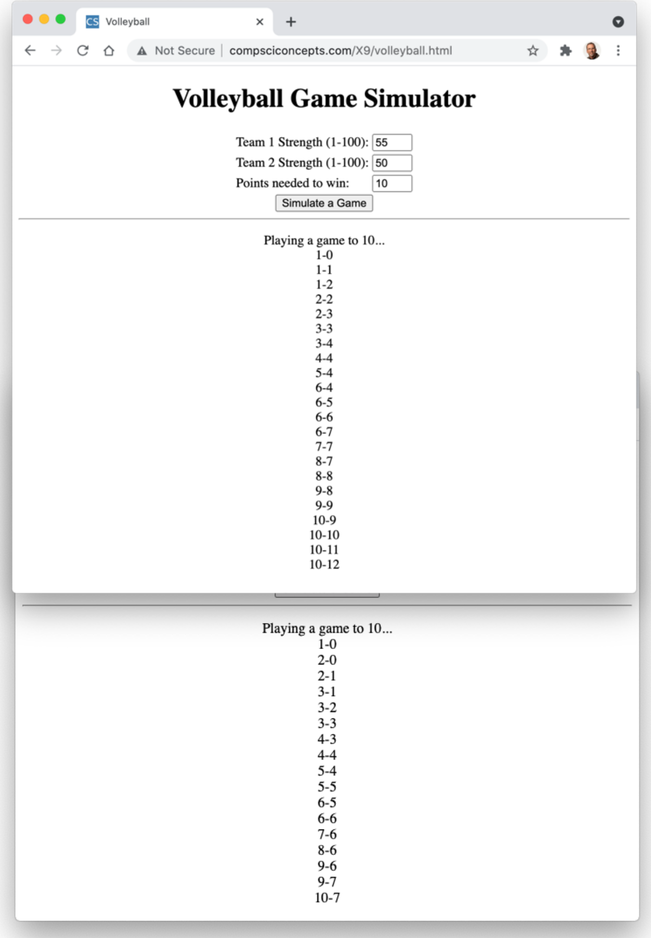
  1. generate a random integer in the range 1 to (strength1+strength2)
  2. if that random integer <= strength1, then team1 wins the rally
  3. otherwise, team2 wins the rally

- e.g., suppose team1 has strength 50 and team2 has strength 50
  if pick a random number in 1…100, it is equally likely to be 1..50 as 51..100

- e.g., suppose team1 has strength 80 and team2 has strength 40
  if pick a random number in 1…120, it is twice as likely to be 1..80 as 81-120

Volleyball

```html
<!doctype html>
<!-- volleyball.html                                     Dave Reed -->
<!-- This page simulates a game of volleyball between two ranked teams. -->
<!-- ================================================================= -->

<html>
  <head>
    <title> Volleyball </title>
    <script src="http://compsciconcepts.com/random.js"></script>
    <script>
      function PlayGame() {
        team1 = team1Box.valueAsNumber;
        team2 = team2Box.valueAsNumber;
        needed = pointsBox.valueAsNumber;

        score1 = 0;
        score2 = 0;

        outputP.innerHTML = 'Playing a game to ' + needed + '...<br>';

        while ((score1 < needed && score2 < needed) || Math.abs(score1-score2) < 2) {
          roll = RandomInt(1, team1+team2);
          if (roll <= team1) {
            score1 = score1 + 1;
          }
          else {
            score2 = score2 + 1;
          }

          outputP.innerHTML = outputP.innerHTML + score1 + '-' + score2 + '<br>';
        }
      }
    </script>
  </head>

  <body style="text-align:center">
    <h1 >Volleyball Game Simulator</h1>
    <table style="margin-left:auto; margin-right:auto; text-align:left">
      <tr><td>Team 1 Strength (1-100): </td>
        <td><input type="number" id="team1Box" min= 0 max=100 value=50></td></tr>
      <tr><td>Team 2 Strength (1-100):</td>
          <td><input type="number" id="team2Box" min=0 max=100 value=50></td></tr>
      <tr><td>Points needed to win:</td>
          <td><input type="number" id="pointsBox" min=1 max=999 value=25></td></tr>
    </table>
    <button onclick="PlayGame();">Simulate a Game</button>
    <hr>
    <p id="outputP"></p>
  </body>
</html>
```



Volleyball Game Simulator

Team 1 Strength (1-100): 55
Team 2 Strength (1-100): 50
Points needed to win: 10

Simulate a Game

```
Playing a game to 10...
1-0
1-1
1-2
2-2
2-3
3-3
3-4
4-4
5-4
6-4
6-5
6-6
6-7
7-7
8-7
8-8
9-8
9-9
10-9
10-10
10-11
10-12
```

```
Playing a game to 10...
1-0
2-0
2-1
3-1
3-2
3-3
4-3
4-4
5-4
5-5
6-5
6-6
7-6
8-6
9-6
9-7
10-7
```

Simulating Many Games

with any nondeterministic model, need to perform a large number of simulations to obtain statistically meaningful results

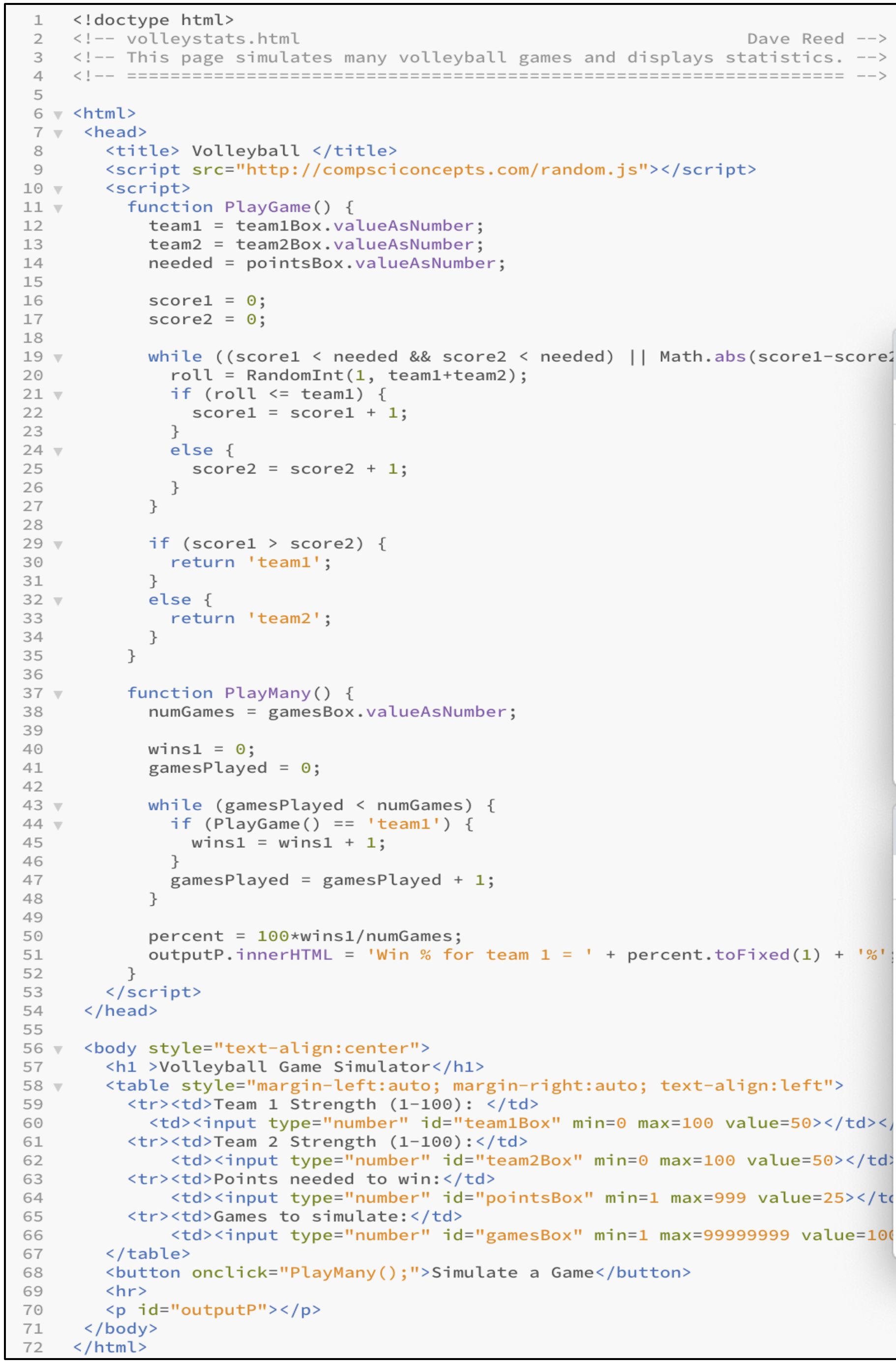
- with only a small number, lucky/unlucky streaks can greatly skew the results
- e.g., 7 HEADS out of 10 coin flips would not shock you, 700 out of 1,000 should

if we want to simulate thousands of games, we don't need to see point-by-point scores

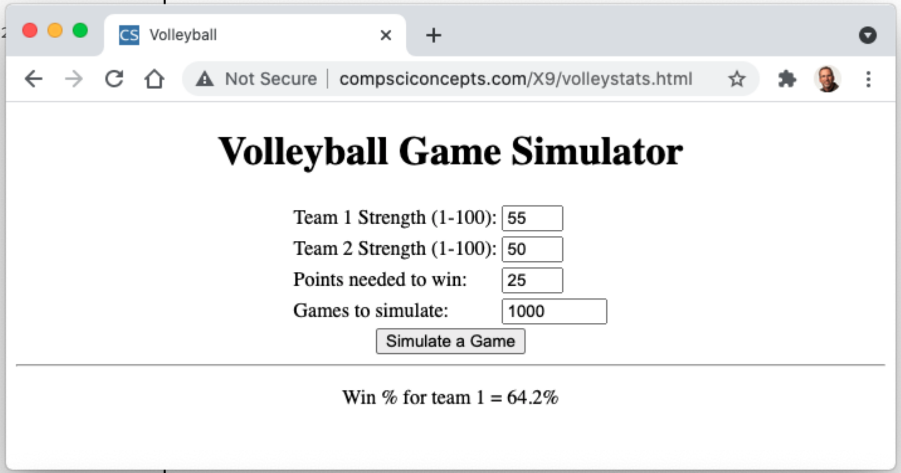
- similar to roll stats example from Ch. X8, simply keep a counter of wins/losses and display the final result when done

1. remove the statements from `PlayGame` that display the score
2. instead, add an if statement at the end of the function that determines the winner and returns either 'team1' or 'team2'
3. define a function that simulates a specified number of games (by calling `PlayGame` inside a loop) and keeps count of wins by team1

Volleyball



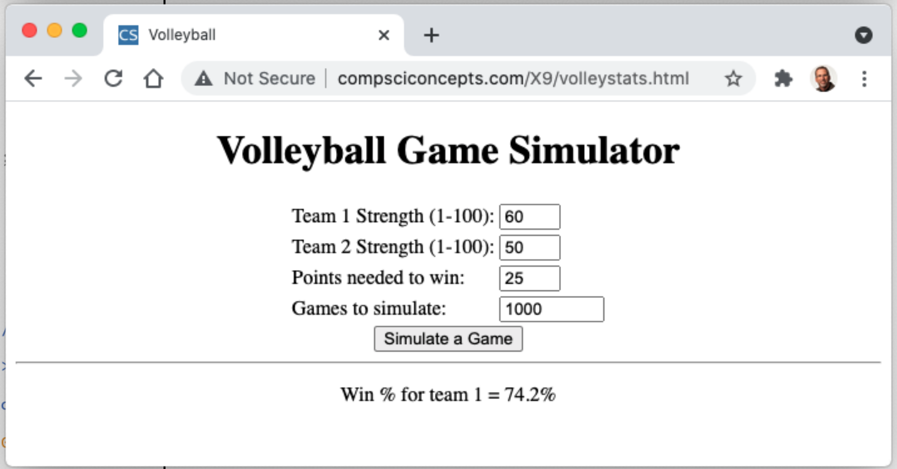
```
<!doctype html>
<!-- volleystats.html                                         Dave Reed -->
<!-- This page simulates many volleyball games and displays statistics. -->
<!-- ================================================================== -->

<html>
 <head>
   <title> Volleyball </title>
   <script src="http://compsciconcepts.com/random.js"></script>
   <script>
     function PlayGame() {
       team1 = team1Box.valueAsNumber;
       team2 = team2Box.valueAsNumber;
       needed = pointsBox.valueAsNumber;

       score1 = 0;
       score2 = 0;

       while ((score1 < needed && score2 < needed) || Math.abs(score1-score2
         roll = RandomInt(1, team1+team2);
         if (roll <= team1) {
           score1 = score1 + 1;
         }
         else {
           score2 = score2 + 1;
         }
       }

       if (score1 > score2) {
         return 'team1';
       }
       else {
         return 'team2';
       }
     }

     function PlayMany() {
       numGames = gamesBox.valueAsNumber;

       wins1 = 0;
       gamesPlayed = 0;

       while (gamesPlayed < numGames) {
         if (PlayGame() == 'team1') {
           wins1 = wins1 + 1;
         }
         gamesPlayed = gamesPlayed + 1;
       }

       percent = 100*wins1/numGames;
       outputP.innerHTML = 'Win % for team 1 = ' + percent.toFixed(1) + '%';
     }
   </script>
 </head>

 <body style="text-align:center">
   <h1 >Volleyball Game Simulator</h1>
   <table style="margin-left:auto; margin-right:auto; text-align:left">
     <tr><td>Team 1 Strength (1-100): </td>
       <td><input type="number" id="team1Box" min=0 max=100 value=50></td></
     <tr><td>Team 2 Strength (1-100):</td>
         <td><input type="number" id="team2Box" min=0 max=100 value=50></td>
     <tr><td>Points needed to win:</td>
         <td><input type="number" id="pointsBox" min=1 max=999 value=25></t
     <tr><td>Games to simulate:</td>
         <td><input type="number" id="gamesBox" min=1 max=99999999 value=10
   </table>
   <button onclick="PlayMany();">Simulate a Game</button>
   <hr>
   <p id="outputP"></p>
 </body>
</html>
```



Volleyball Game Simulator

Team 1 Strength (1-100): 55
Team 2 Strength (1-100): 50
Points needed to win: 25
Games to simulate: 1000

Win % for team 1 = 64.2%



Volleyball Game Simulator

Team 1 Strength (1-100): 60
Team 2 Strength (1-100): 50
Points needed to win: 25
Games to simulate: 1000

Win % for team 1 = 74.2%

Electricity and Switches

modern computers are powered by electricity, using electrical signals to store and manipulate information

the components of a computer require electrical power to carry out their assigned task

- electricity generates the light that shines through a computer screen, illuminating the individual pixels that make up images and letters
- electricity runs the motor that spins the hard-drive disk, allowing information to be accessed
- main memory and CPU employ electrical signals to store and manipulate data
- bit patterns are represented by the presence or absence of electrical current along a wire

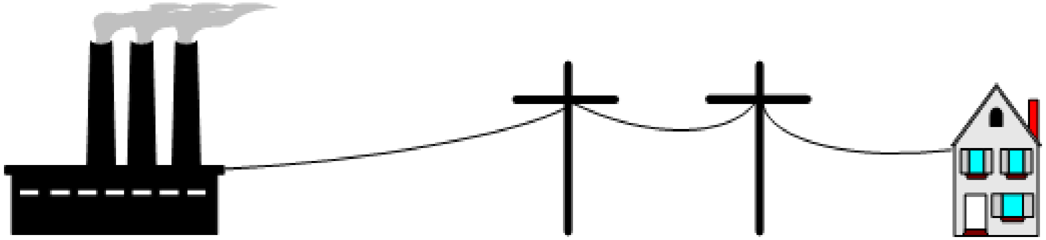
Electricity Basics

electricity is a flow of *electrons,* the negatively charged particles in atoms, through a medium

- good conductors of electricity allow for the flow of electrons with little resistance (e.g., copper, silver, gold)
- other elements, especially nonmetals, are poor conductors (e.g., carbon, oxygen)

electricity can be quantified in *amperes* or *voltage*

- *amperes* gauge electron flow: 1 amp is equal to 6.24 quintillion electrons flowing past a given point each second
- *voltage* measures the physical force produced by the flow of electrons: standard household in United States has 110 to 120 volt outlets



1. Electricity is generated at a power plant.
For example,
(a) Coal or oil is burned to produce steam.
(b) The steam turns a turbine.
(c) The blades of the turbine pass through a magnetic field, producing electricity.

2. The electrical current travels through copper power lines to reach cities and neihborhoods.

3. Transformers distribute the electricity to individual homes and businesses, where it provides power for lights and appliances.

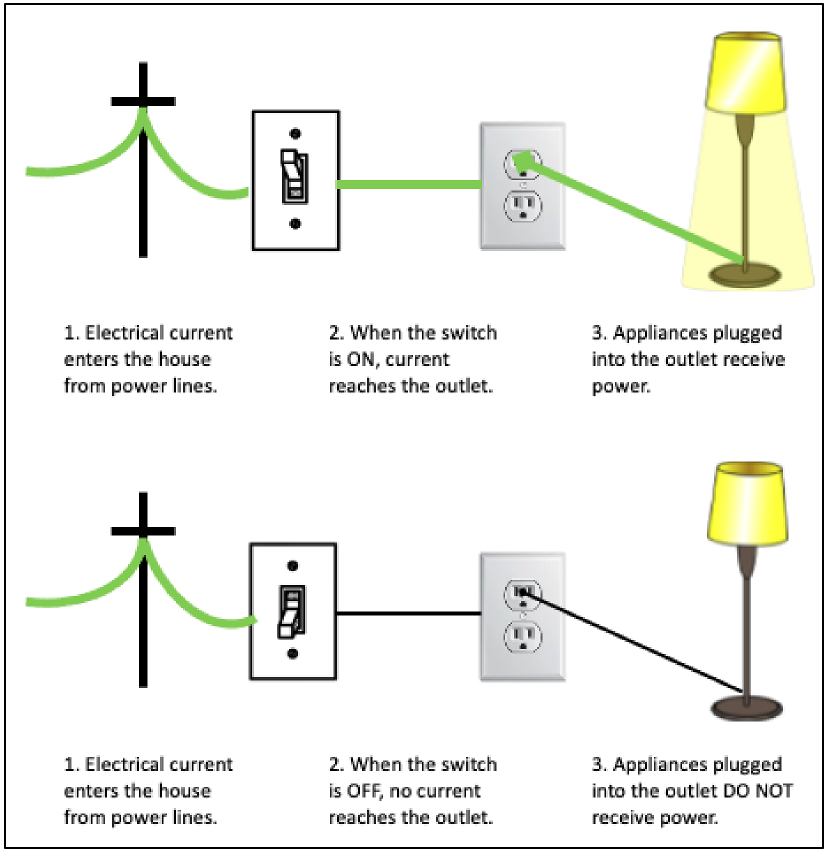
Switches

the most basic tool for controlling the flow of electricity is a *switch*

- a switch can be flipped to connect or disconnect two wires, thus regulating the flow of electricity between them

example: a light switch on a wall serves as an intermediary between the power line entering your home and the outlet that operates a lighting fixture

- if the switch is turned on, then the wires that link the outlet to the power line are connected, and the lighting fixture receives electricity
- if the switch is turned off, then the connection is interrupted, and no power reaches the outlet


1. Electrical current enters the house from power lines.
2. When the switch is ON, current reaches the outlet.
3. Appliances plugged into the outlet receive power.
1. Electrical current enters the house from power lines.
2. When the switch is OFF, no current reaches the outlet.
3. Appliances plugged into the outlet DO NOT receive power.

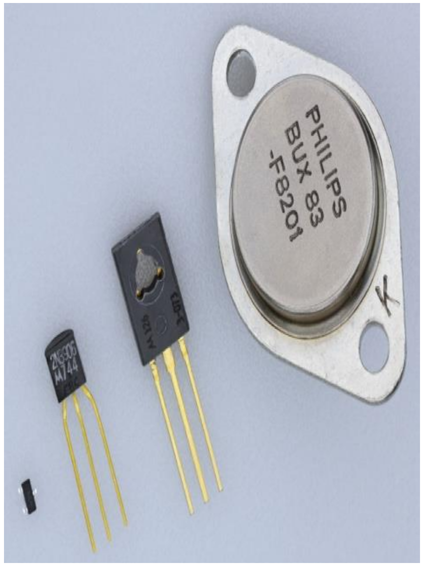
Transistors

as we saw in Chapter C4, advances in switching technology have defined the generations of computers

- 1930's – electromagnetic relays served as physical switches, with on/off positions controlled by the voltage to a magnet
- 1940's – vacuum tubes replaced relays, which were faster (since no moving parts) but tended to overheat and burn out frequently
- 1948 – the transistor was developed by Bardeen, Brattain, and Shockley
  - a transistor is a solid piece of metal attached to a wire that serves as a switch by alternatively conducting or resisting electricity
  - transistors allowed for the development of smaller, faster machines at a lower cost

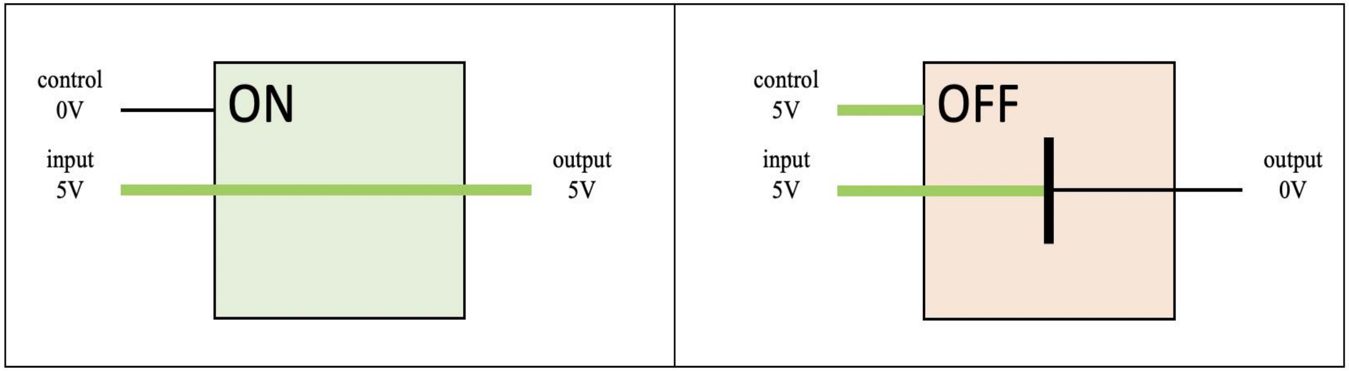
semiconductors are metals that can be manipulated to be either good or bad conductors of electricity

- the first transistors were made of germanium and gold, but modern transistors are constructed from silicon
- through a process known as *doping*, impurities are added to a slab of silicon, causing the metal to act as an electrical switch



Transistors as Switches

a *PMOS transistor* is positively doped, so that the switch is ON (or closed) when there is no current on the control wire, but OFF (or open) when current is applied



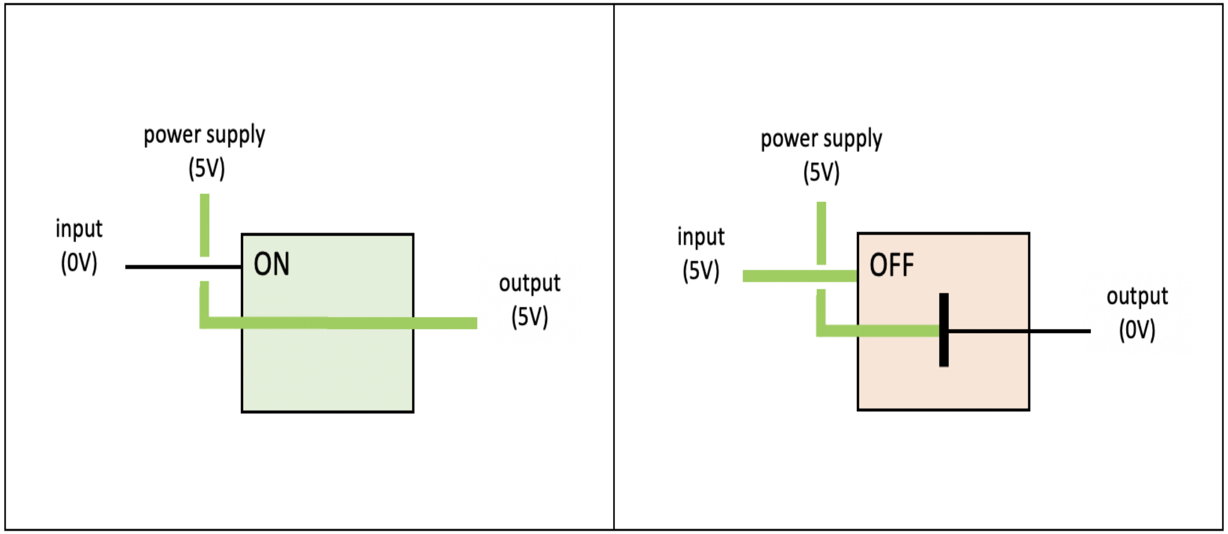
transistors can be combined with other electronic components to form *circuits,* which control the flow of electricity in order to produce a particular behavior

From Transistors to Gates

consider the following circuit combining a PMOS transistor with a power supply

- note: the power supply is connected to the transistor input wire, the circuit input is connected to the control wire

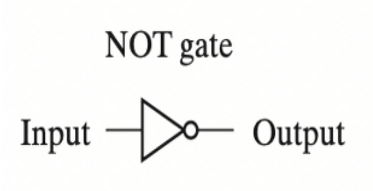
- if no current (0 volts) is applied to the circuit input wire, the transistor will be ON to allow current to travel on the output wire
- if current (5 volts) is applied to the circuit input wire, the transistor will be OFF so no current reaches the output wire
- the result is that the output is the opposite of the input
- this circuit known as a *NOT gate*



Gates and Binary Logic

the term “gate” suggests a simple circuit that controls the flow of electricity

- in the case of a NOT gate, the flow of electricity is manipulated so that the output signal is always opposite of the input signal
- we can think of a gate as computing a function of binary values
  - 0 represents no current; 1 represents current



input	NOT output
0	1
1	0

- the symbol to the left (triangle w/ circle) is often used to denote a NOT gate
- the *truth table* to the right describes the mapping of input to output

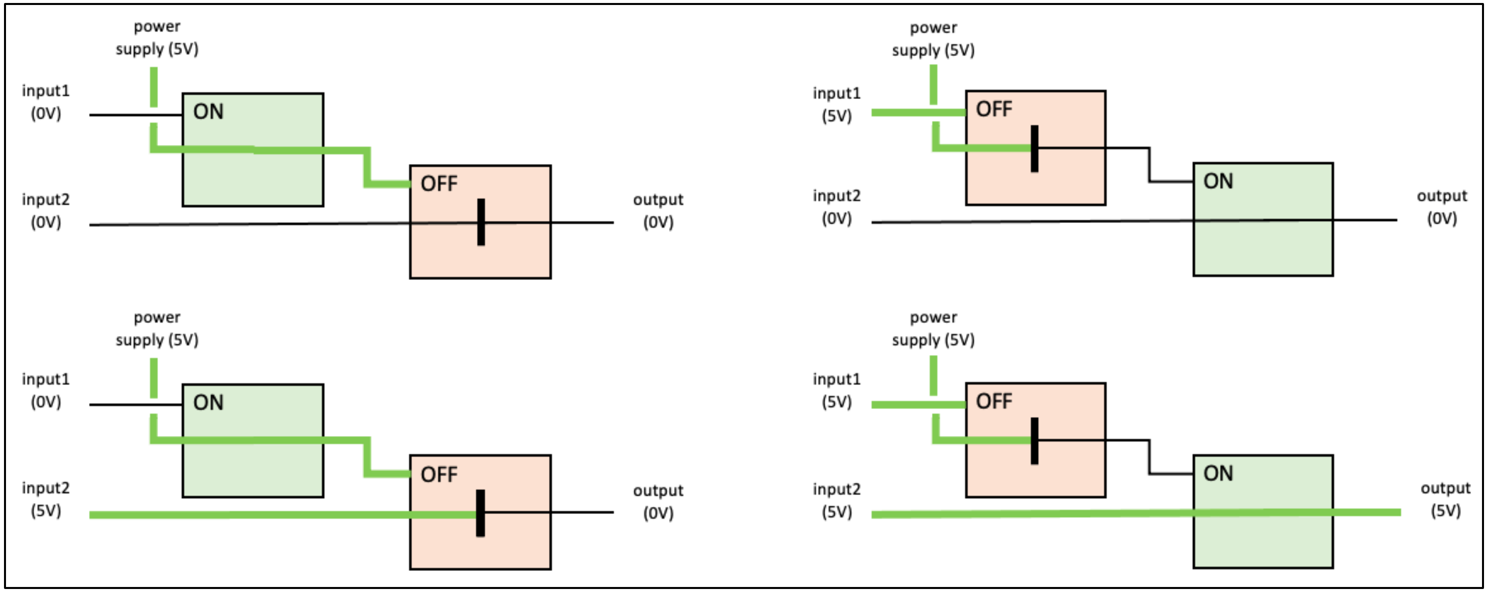
note: NOT gates invert voltages in the same way that the JavaScript NOT operator (!) inverts Boolean values

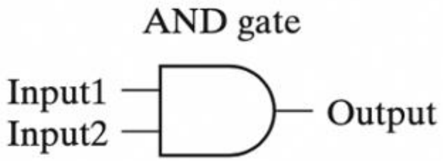
- 0 corresponds to false; 1 corresponds to true

Gates and Binary Logic

many other simple circuits can be defined to perform useful tasks

- AND gate – produces voltage on its output wire if both input wires carry voltage

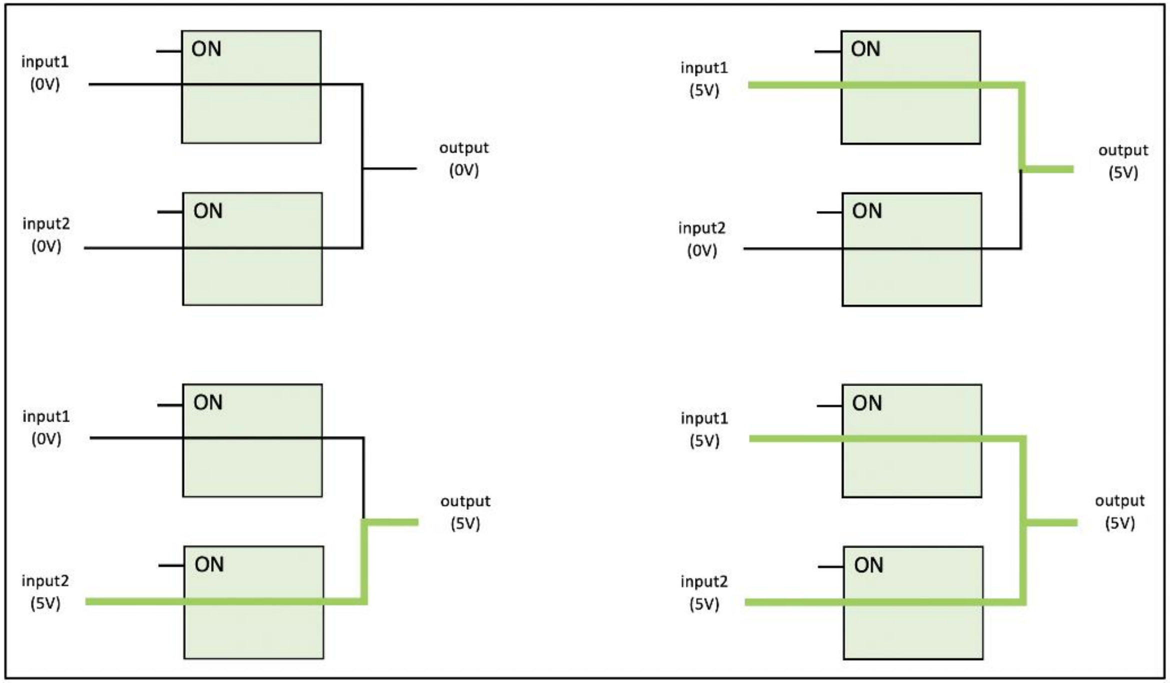


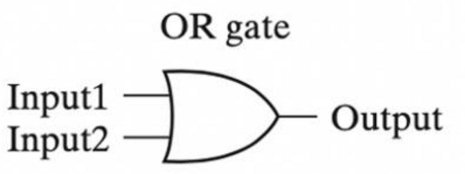


input 1	input 2	AND output
0	0	0
0	1	0
1	0	0
1	1	1

Gates and Binary Logic

- OR gate – produces voltage on its output wire if either input wire carries voltage





input 1	input 2	OR output
0	0	0
0	1	1
1	0	1
1	1	1

From Gates to Circuits

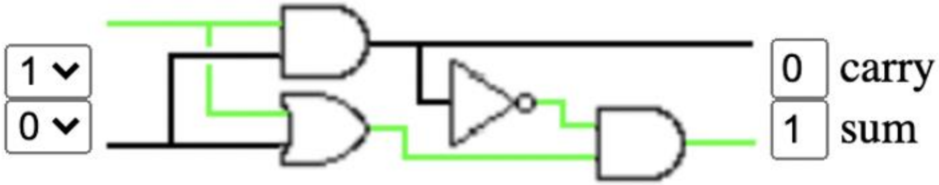
basic logic gates can be combined to build more advanced circuitry

example: adding two binary numbers

$$\begin{array}{ccccc} & & ^{1} & ^{1} & \\ & 1 & 0 & 1 & 1_2 \\ + & & & & 1_2 \\ \hline & 1 & 1 & 0 & 0_2 \end{array} \qquad \begin{array}{cccccc} & ^{1} & ^{1} & & & \\ & & 1 & 1 & 0 & 0_2 \\ + & & & 1 & 0 & 1_2 \\ \hline & 1 & 0 & 0 & 0 & 1_2 \end{array}$$

although binary addition is relatively straightforward, designing a circuit for adding binary numbers is quite complex

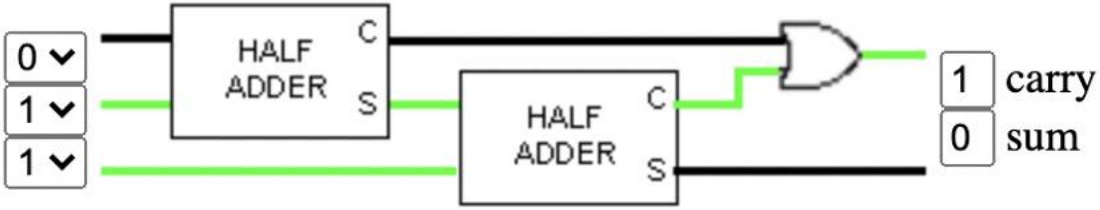
- instead of starting at the transistor level, we can use AND, OR, and NOT gates
- focus first on the addition of 2 bits
  - requires two input lines, two output lines (sum of inputs and possible carry)
  - the circuit consist of four gates (known as a *half-adder*)



Full-adder Circuit

the term “half-adder” refers to the fact that when you add binary numbers containing more than one bit, summing the corresponding bit pairs by column is only half the job

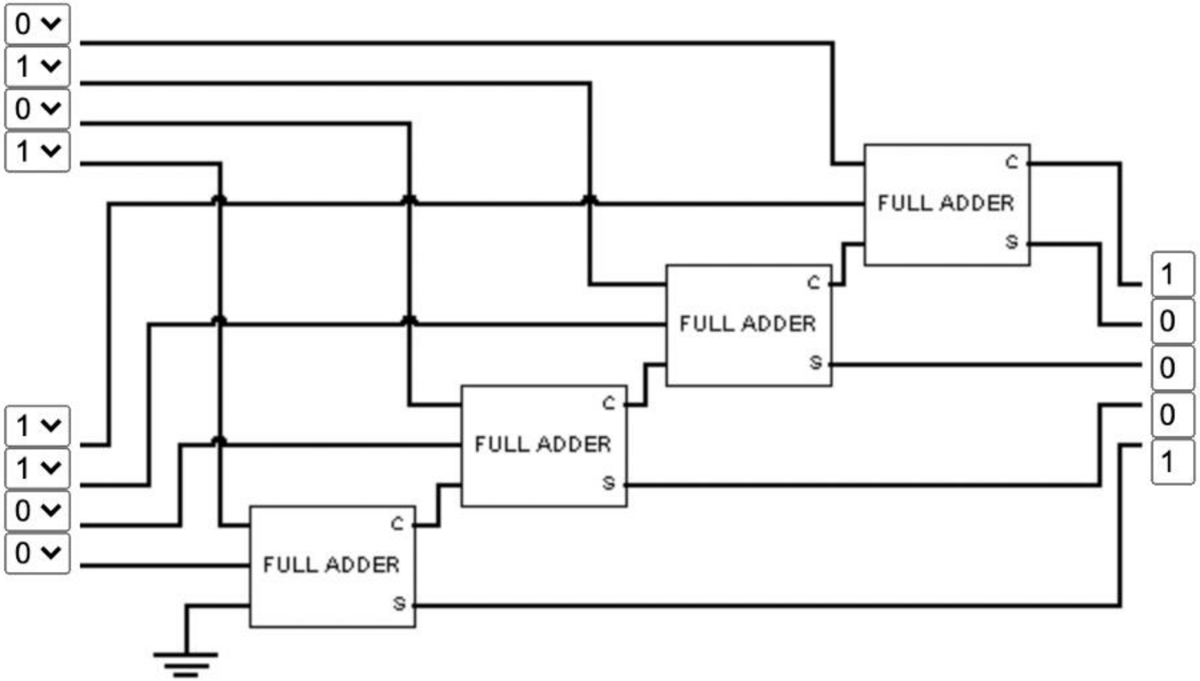
- you must also consider that a bit might be carried over from the previous addition
- using half-adders and logical gates as building blocks, we can design a circuit that takes this into account (known as a *full-adder*)



4-bit Adder Circuit

using full-adders as building blocks, we can design a more complex circuit that sums two 4-bit numbers

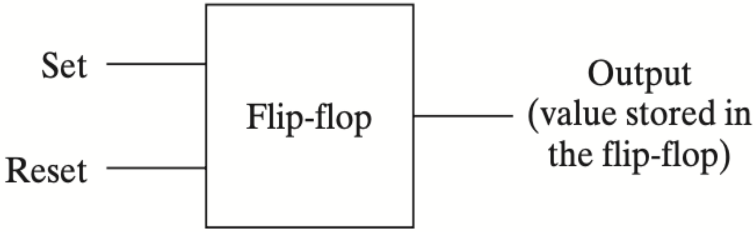
- since a full-adder is required to add each corresponding bit pair together (along with possible carry), the circuit will need four full-adders wired together



Designing Memory Circuitry

main memory and registers within the CPU are composed of circuitry

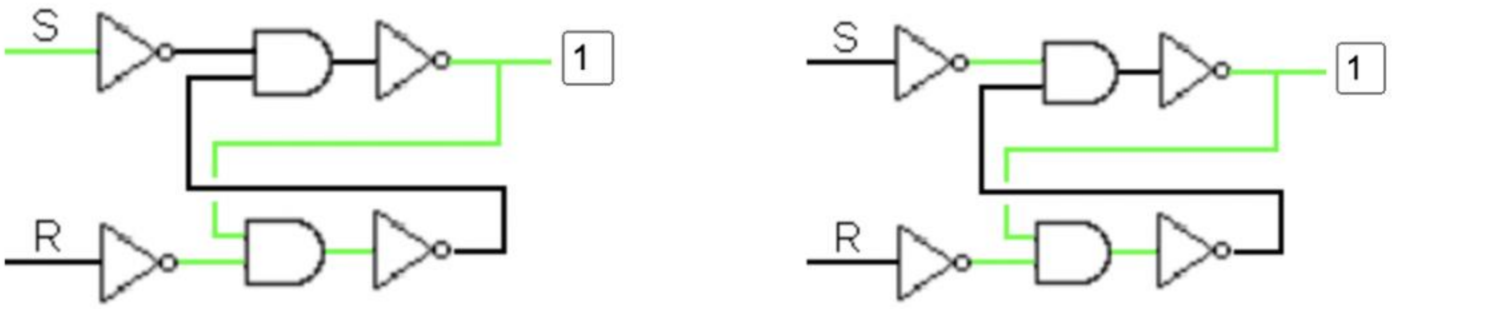
- whereas adders manipulate inputs to produce outputs, memory circuits must maintain values over time
- the simplest circuit for storing a value is known as a *flip-flop*
  - it can be set to store a 1 by applying current on an input wire
  - it can be reset to store a 0 by applying current on another input wire



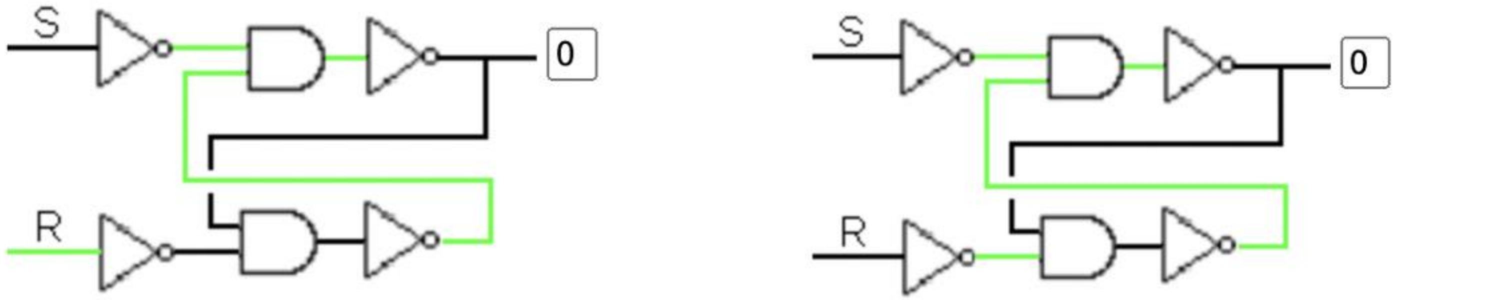
Flip-flop Circuit

a flip-flop stores a value by feeding the output currents back into the circuit

- the value is maintained by current flowing around and around the circuit
- a current burst on the Set wire produces output current, which then cycles



- a current burst on the Reset wire produces no output current



From Circuits to Microchips

initially, circuits were built by wiring together individual transistors

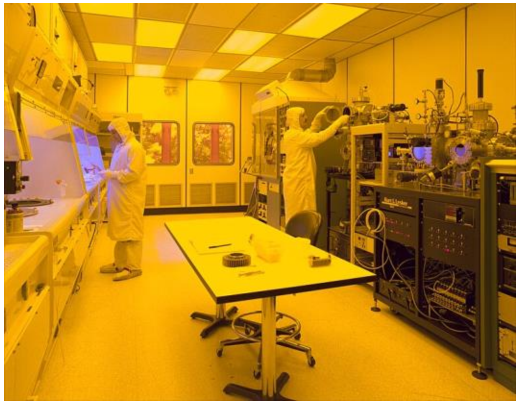
- this did not lend itself to mass production
- even simple circuits consisting of 10s or 100s of transistors were quite large

in 1958, two researchers (Jack Kilby and Robert Noyce) independently developed techniques that allowed for the mass-production of circuitry

- circuitry (transistors + connections) is layered onto a single wafer of silicon, known as a *microchip* or *chip*
- since every component is integrated onto the same microchip, these circuits became known as *integrated circuits*

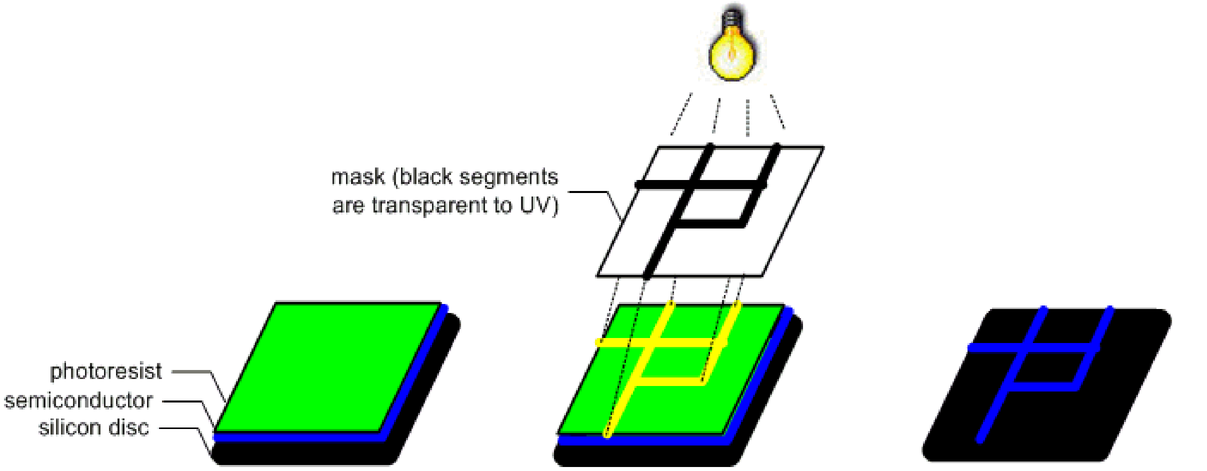
the production of integrated circuits is one of the most complex engineering processes in the world

- transistors can be as small as 5 nanometers (~ 1/16,000th the width of a human hair)
- since a hair or dust can damage circuitry, chips are created in climate-controlled "clean rooms"



Manufacturing ICs

1. initially, the silicon chip is covered with a semiconductor material, then coated with a layer of photoresist (a chemical sensitive to UV light)
2. transistors are then printed onto a mask (transparent surface on which an opaque coating has been applied to form patterns)
3. UV light is filtered through the mask, passing through the transparent portions and striking the surface of the chip in the specified pattern
4. the photoresist exposed to UV light reacts, hardening the layer of the semiconductor below it
5. the photoresist that was not exposed and the soft layer of semiconductor below are etched away, leaving only the desired pattern of semiconductor material on the surface of the chip
6. the process can be repeated 40-60 times depositing multiple layers



1. The surface of the silicon chip is coated with a semiconductor material, followed by a layer of photoresist.

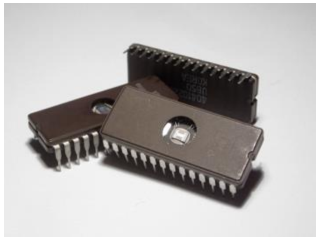
2. Ultraviolet-radiation is sent through a mask, reacting with the photoresist where it passes through the mask to strike the chip.

3. The photresist and unexposed semiconductor material is etched away, leaving the desired pattern on the chip.

Packaging Microchips

since a silicon chip is fragile, the chip is encased in plastic for protection

- metal pins are inserted on both sides of the packaging, facilitating easy connections to other microchips

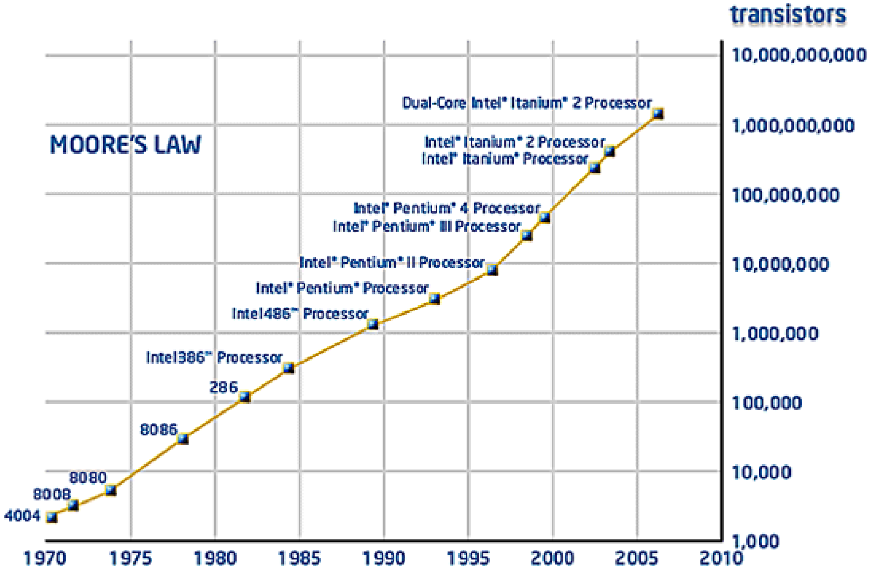


impact of the microchip

- lower cost due to mass production
- faster operation speed due to the close proximity of circuits on chips
- simpler design/construction of computers using prepackaged components

Moore's Law describes the remarkable evolution of manufacturing technology

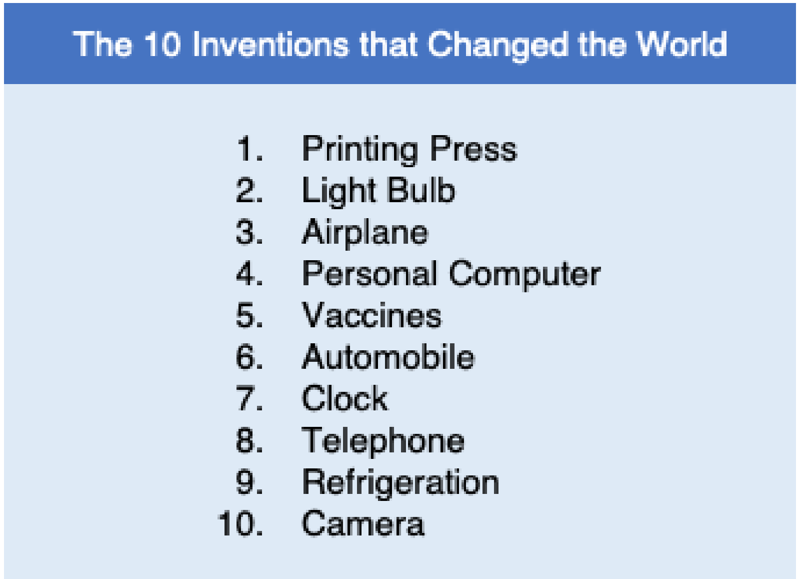
- Moore noted the # of transistors on a chip doubles every 1-2 years
- has held true for 50 years
- technology has slowed, but the pattern continues to hold due to multicore processors



Most impactful inventions

science writer Daniel Stone ranked the 10 most important/impactful inventions

- do you agree with his list?



The 10 Inventions that Changed the World

1. Printing Press
2. Light Bulb
3. Airplane
4. Personal Computer
5. Vaccines
6. Automobile
7. Clock
8. Telephone
9. Refrigeration
10. Camera

(Daniel Stone. "The 10 Inventions that Changed the World." National Geographic Magazine, June 2017.)

more than any other invention, computer technology is still evolving, which means that it continues to impact society in new ways

Impact on Money

40+ years ago, the U.S. was a cash-based society

2 tech developments changed that

1. cheap, portable devices for processing credit card transactions
2. expansion of the Internet
   - banks could treat money as numbers in a file
   - led to expansion of ATMs, debit cards, online banking

many Americans now live virtually cash-free

- in 2019, 39 billion credit card transactions, 10 billion ATM transactions
- in 2020, most individuals carried less than $40

cryptocurrencies like Bitcoin are growing in popularity

- market-based, not backed by government or bank
- offers complete anonymity, potentially tax free
- very volatile, not clear if will be widely accepted

Negatives: debt & security

the convenience of ATMs, debit/credit cards has led to a debt crisis for many

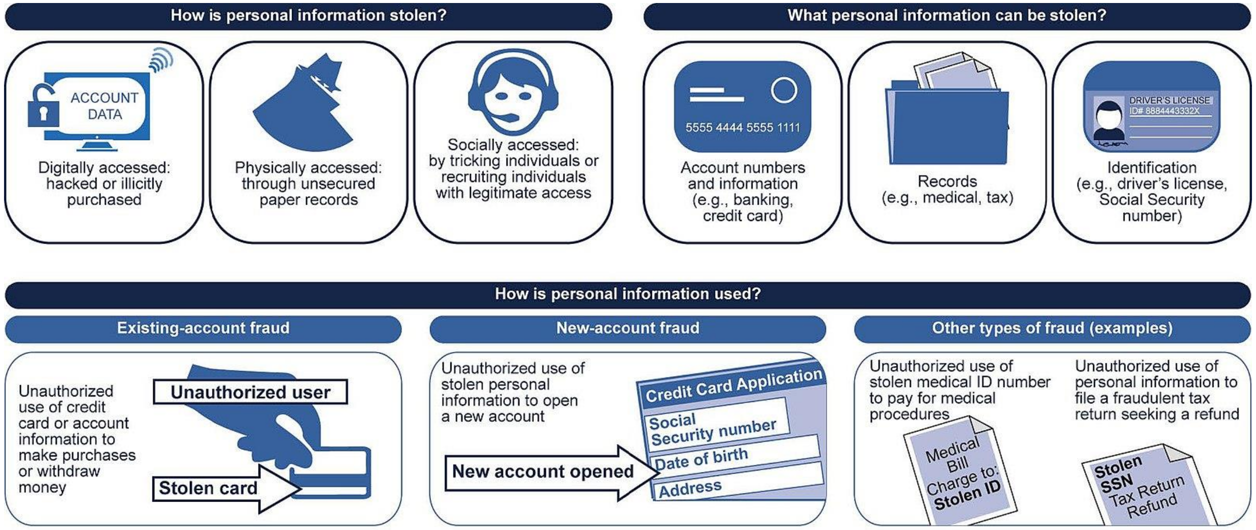
- avg. credit card interest rate: 16%
- by comparison: 30-year mortgage (2.8%), 5-year CD (0.31%)

2021 credit card debt in U.S. totaled $807 billion

- avg. household credit-card debt: $6,270

electronic money meant fewer muggings, but introduced identity theft & fraud

- nearly half of Americans experienced in 2020, total loss of $712 billion



How is personal information stolen?

Digitally accessed: hacked or illicitly purchased

Physically accessed: through unsecured paper records

Socially accessed: by tricking individuals or recruiting individuals with legitimate access

What personal information can be stolen?

Account numbers and information (e.g., banking, credit card)

Records (e.g., medical, tax)

Identification (e.g., driver's license, Social Security number)

How is personal information used?

Existing-account fraud

Unauthorized use of credit card or account information to make purchases or withdraw money

New-account fraud

Unauthorized use of stolen personal information to open a new account

Other types of fraud (examples)

Unauthorized use of stolen medical ID number to pay for medical procedures

Unauthorized use of personal information to file a fraudulent tax return seeking a refund

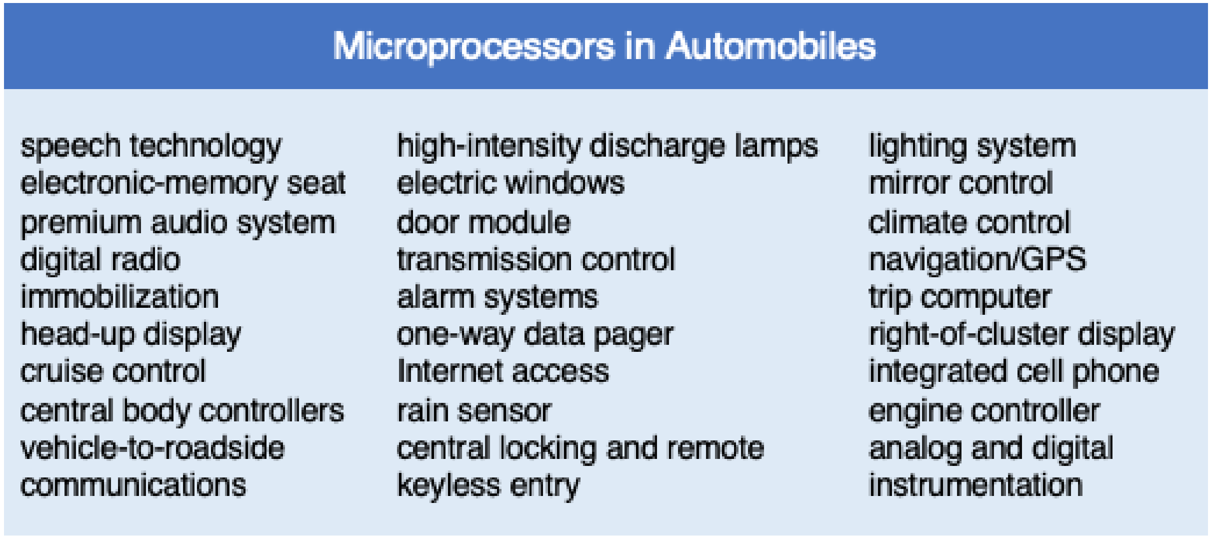
Source: GAO. | GAO-17-254

Computers in Everyday Tasks

modern life also depends on thousands of less obvious, hidden computer applications

embedded processors are computer chips that are built into appliances and machinery to control their workings

- they account for more than 98% of all computer processors
- modern homes contain hundreds of embedded processors
  - in ovens, television remote controls, cordless phones, automatic thermostats, …
- automobiles employ embedded processors to control a wide variety of components



Microprocessors in Automobiles

speech technology
electronic-memory seat
premium audio system
digital radio
immobilization
head-up display
cruise control
central body controllers
vehicle-to-roadside communications

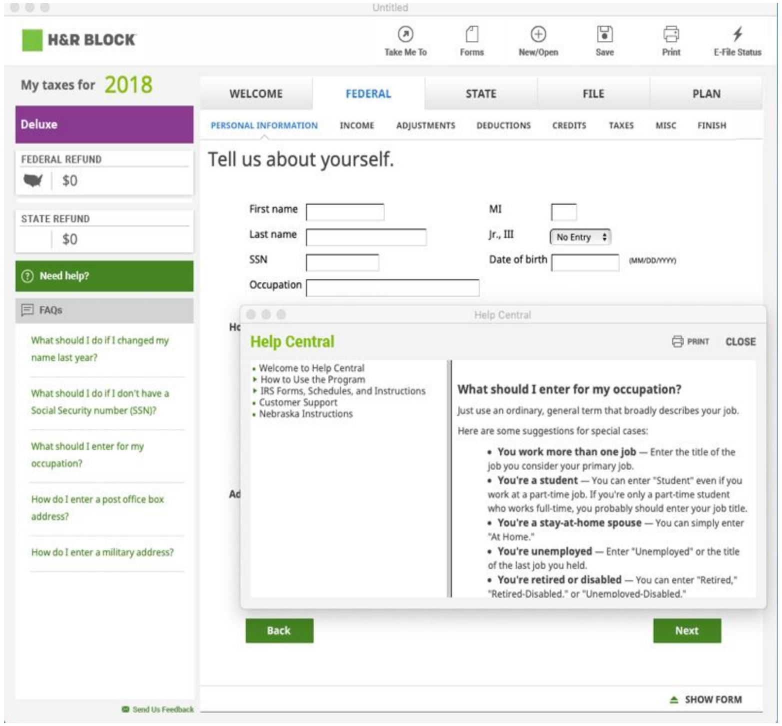
high-intensity discharge lamps
electric windows
door module
transmission control
alarm systems
one-way data pager
Internet access
rain sensor
central locking and remote keyless entry

lighting system
mirror control
climate control
navigation/GPS
trip computer
right-of-cluster display
integrated cell phone
engine controller
analog and digital instrumentation

Computers in Everyday Tasks

society has also been affected by the availability of personal computers and easy-to-use software

- software can enable people to accomplish tasks previously reserved for highly trained professionals, e.g.,
  - word processing and desktop publishing software
  - video editing software
  - tax preparation software



My taxes for 2018

FEDERAL REFUND $0

STATE REFUND $0

What should I do if I changed my name last year?

What should I do if I don't have a Social Security number (SSN)?

What should I enter for my occupation?

How do I enter a post office box address?

How do I enter a military address?

Tell us about yourself.

First name □ MI □

Last name □ Jr., III No Entry

SSN □ Date of birth □ (MM/DD/YYYY)

Occupation □

- Welcome to Help Central
- How to Use the Program
- IRS Forms, Schedules, and Instructions
- Customer Support
- Nebraska Instructions

What should I enter for my occupation?

Just use an ordinary, general term that broadly describes your job.

Here are some suggestions for special cases:

- **You work more than one job** — Enter the title of the job you consider your primary job.
- **You're a student** — You can enter "Student" even if you work at a part-time job. If you're only a part-time student who works full-time, you probably should enter your job title.
- **You're a stay-at-home spouse** — You can simply enter "At Home."
- **You're unemployed** — Enter "Unemployed" or the title of the last job you held.
- **You're retired or disabled** — You can enter "Retired," "Retired-Disabled," or "Unemploved-Disabled."

smart phones and hand-held computers have driven the development of mobile apps

- in June 2021, Apple's App Store listed 2.2 million apps, Google's Play Store listed 3.4 million apps
- Amazon's Kindle & Sony's Reader enable downloading and reading electronic books

Negatives: overreliance

as society becomes dependent on complex, computer-based products and services, the effects of errors or system failures become far-reaching

computer-system bugs can produce dire consequences

- from 1985-1987, 4 cancer patients died from radiation overdoses due to a single coding error in medical equipment software
- in 1991, 28 soldiers were killed by a Scud missile because a software error(involving number roundoff) caused the Patriot missile to miss its target
- in 1999, NASA's Mars Climate Orbiter went off course and was destroyed in the Martian atmosphere (the problem was due to software inconsistencies which used different measurement conversions, e.g., English vs. Metric)
- in 2010, Toyota recalled more than 400,000 hybrid due to faulty anti-lock brake software (estimated cost exceeds $6 billion)
- in 2012, Knight Capital group lost $461 million in 30 minutes due to a bug in their online trading software
- from 2016-2019, self-driving cars have caused 6 fatalities

to avoid errors, various software design and testing methodologies are used

- however, as the size and complexity of the software grows, design and testing become exponentially more difficult
- Windows 2000 – 35 million lines of code, 63,000 known bugs

Internet/Web for Information

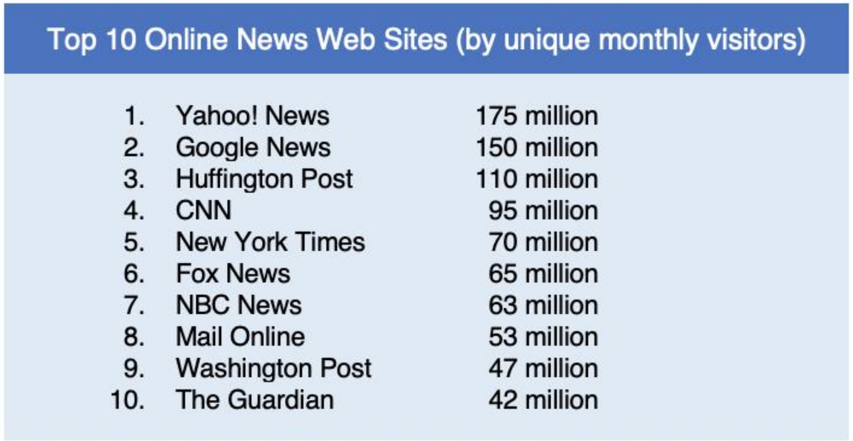
many users utilize the Internet/Web as an information source

online resources are quickly replacing (or complementing) traditional sources of information

- Web sites can be updated 24 hours/day, can report on breaking stories
- text can be integrated with other types of media
- the immediacy of online delivery system is especially appealing

independent media organizations have utilized the Web to present stories and opinions that might not otherwise reach a mainstream audience

in order to compete, many newspapers/magazines now offer online services



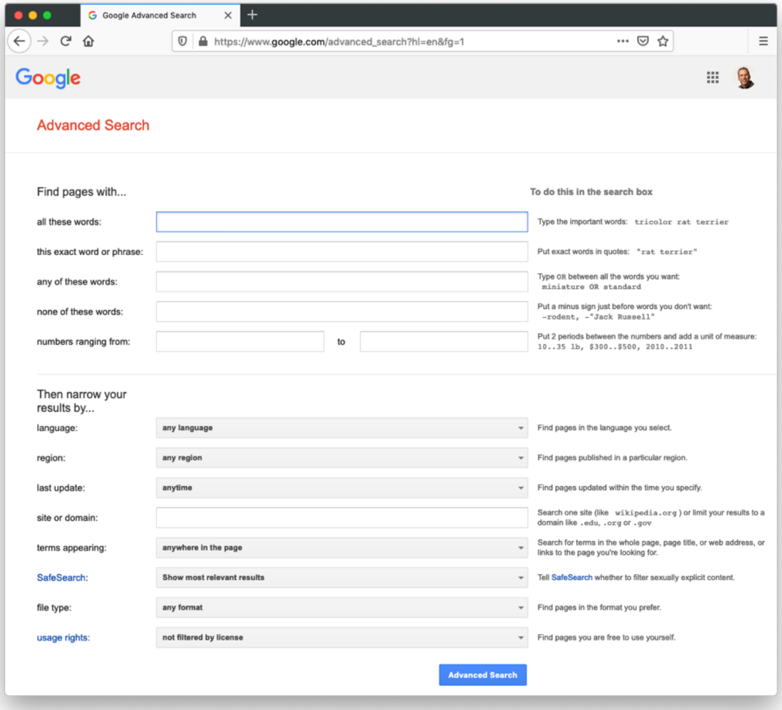
Top 10 Online News Web Sites (by unique monthly visitors)

1.	Yahoo! News	175 million
2.	Google News	150 million
3.	Huffington Post	110 million
4.	CNN	95 million
5.	New York Times	70 million
6.	Fox News	65 million
7.	NBC News	63 million
8.	Mail Online	53 million
9.	Washington Post	47 million
10.	The Guardian	42 million

Internet/Web for Information

the majority of Web pages are unique resources created by individuals and private organizations

- you can find Web content on virtually any topic
- to help navigate the vast sea of information, *search engines* automatically catalog Web pages and allow users to search for data by topic or keywords



Find pages with...

		To do this in the search box
all these words:		Type the important words: `tricolor rat terrier`
this exact word or phrase:		Put exact words in quotes: `"rat terrier"`
any of these words:		Type `OR` between all the words you want: `miniature OR standard`
none of these words:		Put a minus sign just before words you don't want: `-rodent, -"Jack Russell"`
numbers ranging from:	to	Put 2 periods between the numbers and add a unit of measure: `10..35 lb, $300..$500, 2010..2011`

Then narrow your results by...

language:	any language	Find pages in the language you select.
region:	any region	Find pages published in a particular region.
last update:	anytime	Find pages updated within the time you specify.
site or domain:		Search one site (like `wikipedia.org`) or limit your results to a domain like `.edu`, `.org` or `.gov`
terms appearing:	anywhere in the page	Search for terms in the whole page, page title, or web address, or links to the page you're looking for.
SafeSearch:	Show most relevant results	Tell SafeSearch whether to filter sexually explicit content.
file type:	any format	Find pages in the format you prefer.
usage rights:	not filtered by license	Find pages you are free to use yourself.

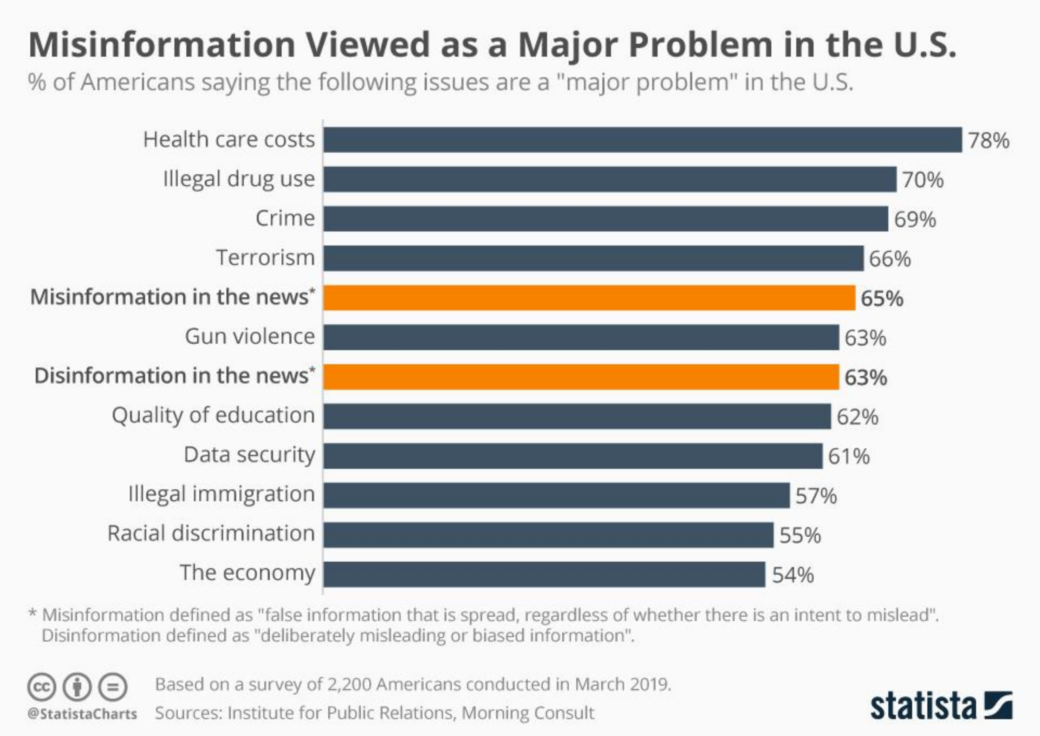
Negatives: fake news

the openness of the Web allows for a diversity of views to be presented

- but, not all views are fact-based or unbiased
- misinformation or disinformation (intentional misinformation) are widespread

Americans rated online misinformation a major problem (2019)

- only 26% were very confident in their ability to recognize fake news



Misinformation Viewed as a Major Problem in the U.S.

% of Americans saying the following issues are a "major problem" in the U.S.

* Misinformation defined as "false information that is spread, regardless of whether there is an intent to mislead". Disinformation defined as "deliberately misleading or biased information".

@StatistaCharts Based on a survey of 2,200 Americans conducted in March 2019.
Sources: Institute for Public Relations, Morning Consult

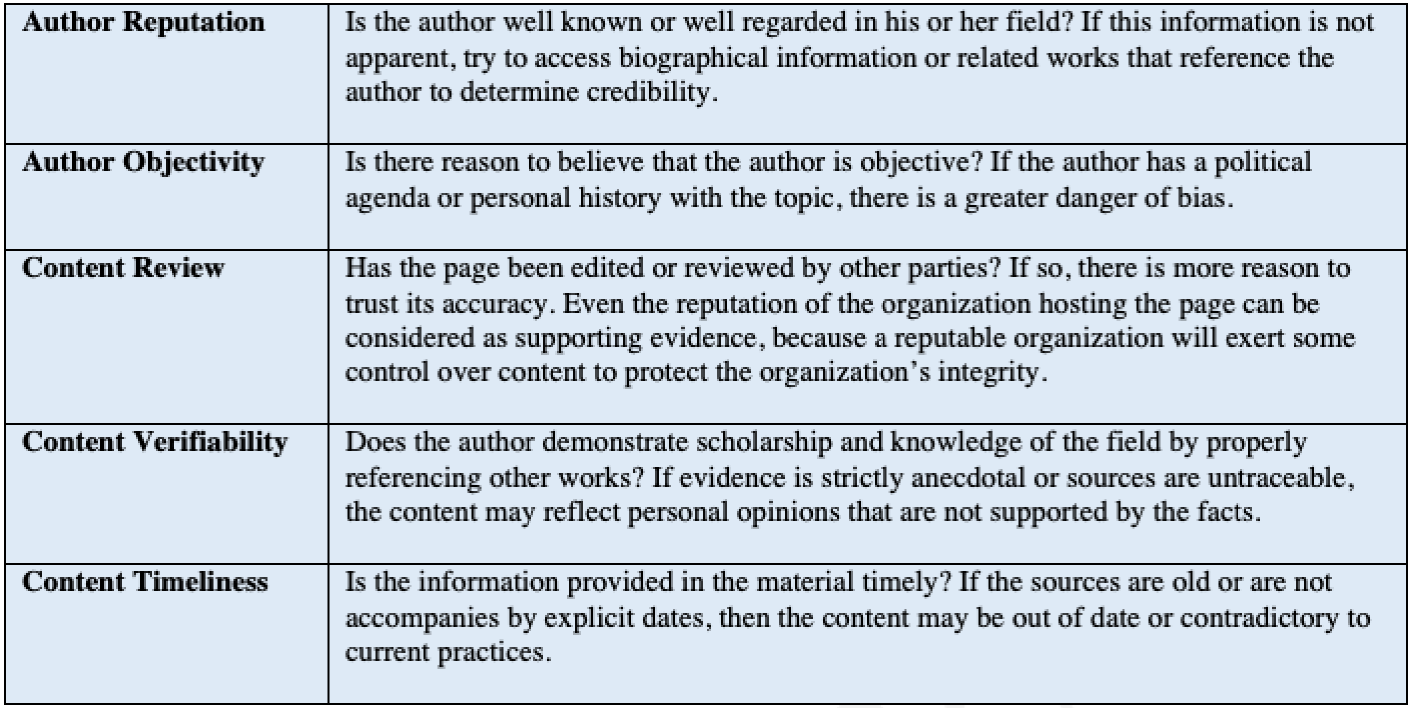
statista

Negatives: info overload

the impressive range of information available online can be viewed as a strength, but it is also one of the greatest weaknesses

- as of 2019, Web estimated in hundreds of billions of pages (maybe much more)
- finding specific info can be hard, even with search engines

since most Internet/Web content lacks editorial review, it is up to the user to evaluate its credibility



Author Reputation	Is the author well known or well regarded in his or her field? If this information is not apparent, try to access biographical information or related works that reference the author to determine credibility.
Author Objectivity	Is there reason to believe that the author is objective? If the author has a political agenda or personal history with the topic, there is a greater danger of bias.
Content Review	Has the page been edited or reviewed by other parties? If so, there is more reason to trust its accuracy. Even the reputation of the organization hosting the page can be considered as supporting evidence, because a reputable organization will exert some control over content to protect the organization's integrity.
Content Verifiability	Does the author demonstrate scholarship and knowledge of the field by properly referencing other works? If evidence is strictly anecdotal or sources are untraceable, the content may reflect personal opinions that are not supported by the facts.
Content Timeliness	Is the information provided in the material timely? If the sources are old or are not accompanies by explicit dates, then the content may be out of date or contradictory to current practices.

Internet/Web for Communication

many users were originally drawn to the Internet by the availability of electronic mail and newsgroups

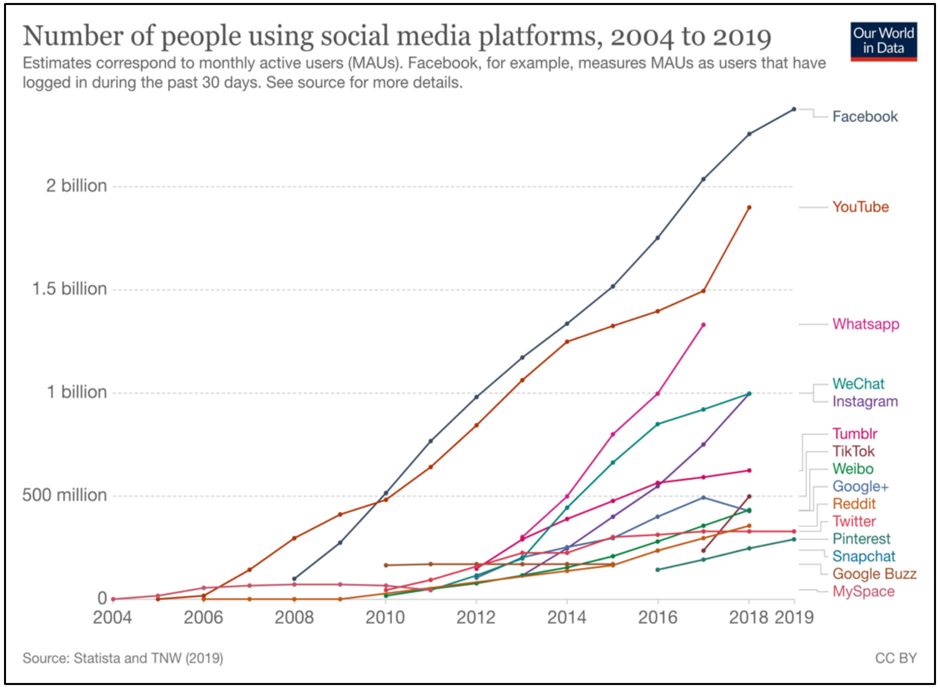
- in 2021, 4.1 billion active email users worldwide (1.8 billion used Gmail)
- 319.6 billion email messages were sent/received each day (~50% spam)

increasingly, the Internet is being used for social networking

- in 2021, 97% of Americans own cell phone, 85% own smartphone
- 23 billion texts sent in 2020

social media sites have grown in popularity

1. Facebook
2. YouTube
3. Whatsapp
4. WeChat
5. Instagram


Number of people using social media platforms, 2004 to 2019

Estimates correspond to monthly active users (MAUs). Facebook, for example, measures MAUs as users that have logged in during the past 30 days. See source for more details.

Source: Statista and TNW (2019)

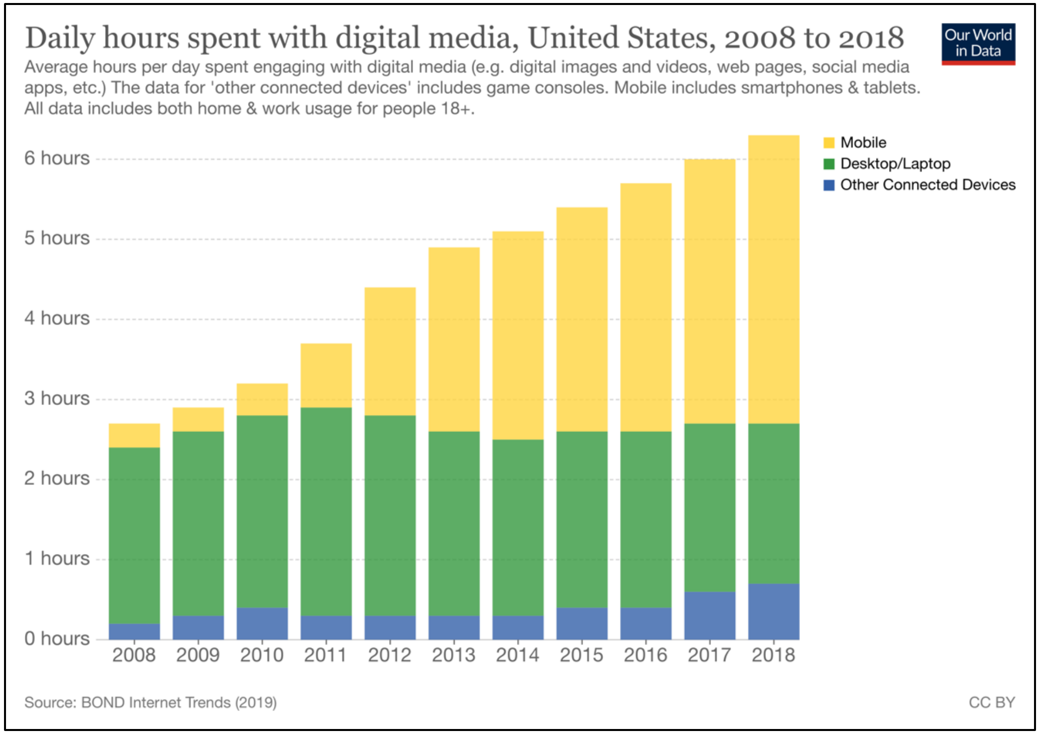
CC BY

Negatives: addiction

studies have shown social media can be physically & emotionally addictive

- in 2018, average adult spent 6 hours/day on connected devices
- assuming 7 hours of sleep, that's 35% of waking day

nomophobia: fear of being detached from mobile phone connectivity



Daily hours spent with digital media, United States, 2008 to 2018

Average hours per day spent engaging with digital media (e.g. digital images and videos, web pages, social media apps, etc.) The data for 'other connected devices' includes game consoles. Mobile includes smartphones & tablets. All data includes both home & work usage for people 18+.

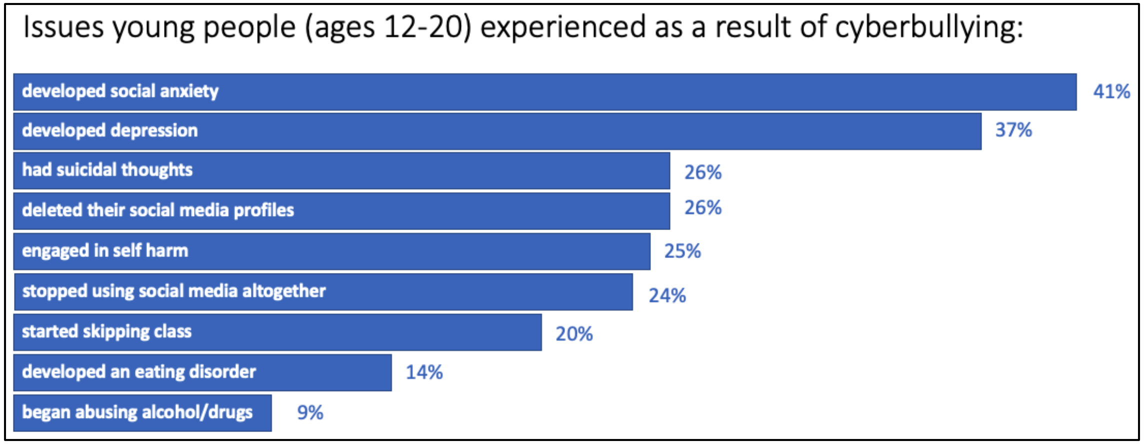
Source: BOND Internet Trends (2019)

CC BY

Negatives: cyberbullying

preteen & teen use of social media has led to bullying via text & social media

- annual survey in U.K. identifies negative consequences



Issues young people (ages 12-20) experienced as a result of cyberbullying:

- suicide rate for teen girls in U.S. increased 65% from 2010-2015, coinciding with rise in smartphone social media use

also, texting while driving is 6x more dangerous than drunk driving

Internet/Web for Commerce

another popular function of the Web is to facilitate *electronic commerce*, or *e-commerce*

- businesses have recognized the Web's potential as an advertising medium, and as a tool for reaching new customers
- some business sites are information-based (providing background on the company or product descriptions)
- other business sites are transaction-based (allowing customers to purchase products or services directly)

online shopping has numerous advantages for the consumer

- you can make purchases from your home at any time
- it is easy to comparison shop
- many online retailers, such as Amazon.com, allow consumers to research products as well as purchase them

Internet/Web for Commerce

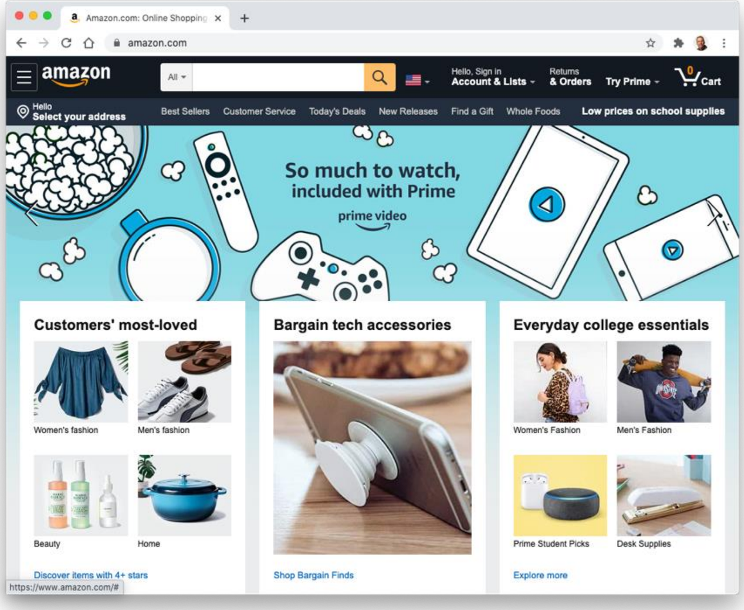
total online sales in 2020: $861 billion

some of the most successful sites are online offshoots of traditional retailers

- Walmart, Best Buy, PetSmart

Amazon is by far the dominant force

- more than 1/3 of ecommerce sector



the Web has provided a new advertising channel for businesses

- e-commerce sites charge fees for hosting advertising banners on Web pages
  - *banner ads* are clickable images that promote a company's product or service
  - users who click on the ad are directed to the company's Web site
- the Web's structure allows for a direct connection between ads and related purchasing interfaces

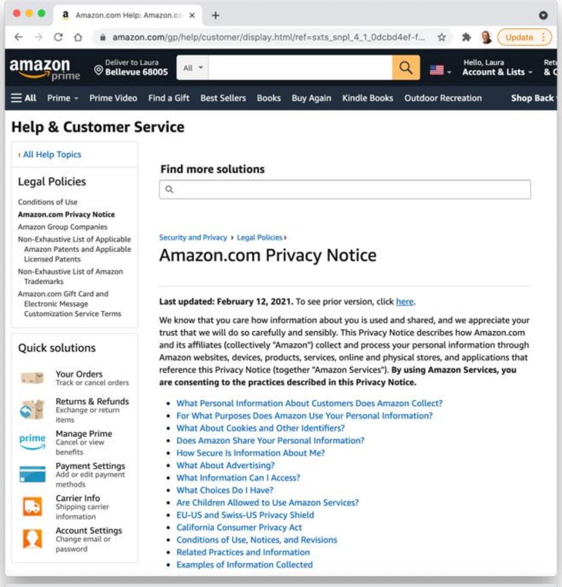
Negatives: privacy

when using credit cards or shopping online, consumers sacrifice privacy for the sake of convenience

- companies maintain records of consumer purchases
- private details can be inferred from shopping patterns
- companies often sell customer profiles to marketing firms

Web users can limit exploitation by interacting only with reputable online businesses with privacy policies

- such policies will explain what information is collected by the business and how that information is to be used (and shared)



Help & Customer Service

‹ All Help Topics

Legal Policies

- Conditions of Use
- **Amazon.com Privacy Notice**
- Amazon Group Companies
- Non-Exhaustive List of Applicable Amazon Patents and Applicable Licensed Patents
- Non-Exhaustive List of Amazon Trademarks
- Amazon.com Gift Card and Electronic Message Customization Service Terms

Quick solutions

- Your Orders — Track or cancel orders
- Returns & Refunds — Exchange or return items
- Manage Prime — Cancel or view benefits
- Payment Settings — Add or edit payment methods
- Carrier Info — Shipping carrier information
- Account Settings — Change email or password

Find more solutions

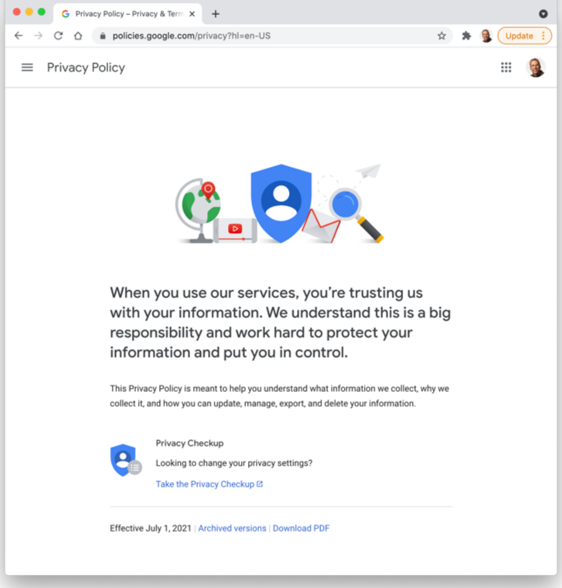
Security and Privacy › Legal Policies ›

Amazon.com Privacy Notice

Last updated: February 12, 2021. To see prior version, click here.

We know that you care how information about you is used and shared, and we appreciate your trust that we will do so carefully and sensibly. This Privacy Notice describes how Amazon.com and its affiliates (collectively "Amazon") collect and process your personal information through Amazon websites, devices, products, services, online and physical stores, and applications that reference this Privacy Notice (together "Amazon Services"). **By using Amazon Services, you are consenting to the practices described in this Privacy Notice.**

- What Personal Information About Customers Does Amazon Collect?
- For What Purposes Does Amazon Use Your Personal Information?
- What About Cookies and Other Identifiers?
- Does Amazon Share Your Personal Information?
- How Secure Is Information About Me?
- What About Advertising?
- What Information Can I Access?
- What Choices Do I Have?
- Are Children Allowed to Use Amazon Services?
- EU-US and Swiss-US Privacy Shield
- California Consumer Privacy Act
- Conditions of Use, Notices, and Revisions
- Related Practices and Information
- Examples of Information Collected


Privacy Policy – Privacy & Term
policies.google.com/privacy?hl=en-US
Update
Privacy Policy

When you use our services, you're trusting us with your information. We understand this is a big responsibility and work hard to protect your information and put you in control.

This Privacy Policy is meant to help you understand what information we collect, why we collect it, and how you can update, manage, export, and delete your information.

Privacy Checkup

Looking to change your privacy settings?

Take the Privacy Checkup

Effective July 1, 2021 | Archived versions | Download PDF

Negatives: security

email also raises privacy concerns

- email messages travel through numerous routers, and each router represents a security risk
- when a message is received it is commonly stored in a file on the recipient’s computer – there is a danger that unauthorized users might get access

few laws apply directly to electronic privacy

- courts overwhelmingly favor employers over employees in privacy suits
- unless explicitly stated, it is generally accepted that employers may access any content on company-owned machines

increasing occurrences of *phishing* attacks, in which people are fooled into surrendering sensitive information via email

privacy is closely linked with security

- with online transactions, credit card numbers or other personal information can be intercepted and subsequently result in identity theft
- encryption methods are commonly used to secure information transmissions, but online fraud is still a continuing problem

In conclusion…

"Technology is, of course, a double edged sword. Fire can cook our food but also burn us." – Jason Silva

overall, changes brought about by computers have been extremely positive

- there is no going back, computers are integrated into our lives

not all change is good, so you need to be informed and diligent about the choices you make

- social media allows people to connect over distances, but also can lead to physical isolation and cyber-bullying
- facial recognition can add security to banking, but can invade privacy and be discriminatory

hopefully, this book has helped you better understand computer technology and its applications

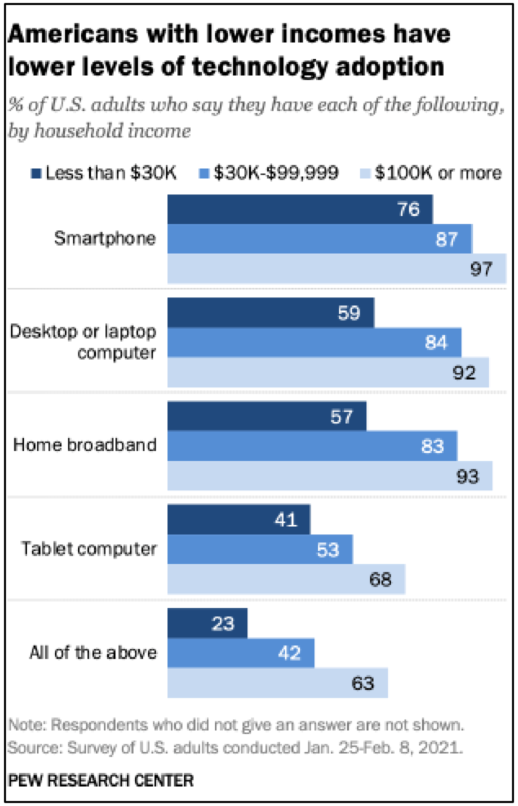
- will enable you to make informed choices about how you adopt technology

The Digital Divide

access to computers & Internet/Web offers numerous advantages

- education, citizenship, employment, …

a troubling aspect of recent technological developments is that the benefits associated with computers are not shared equally by all

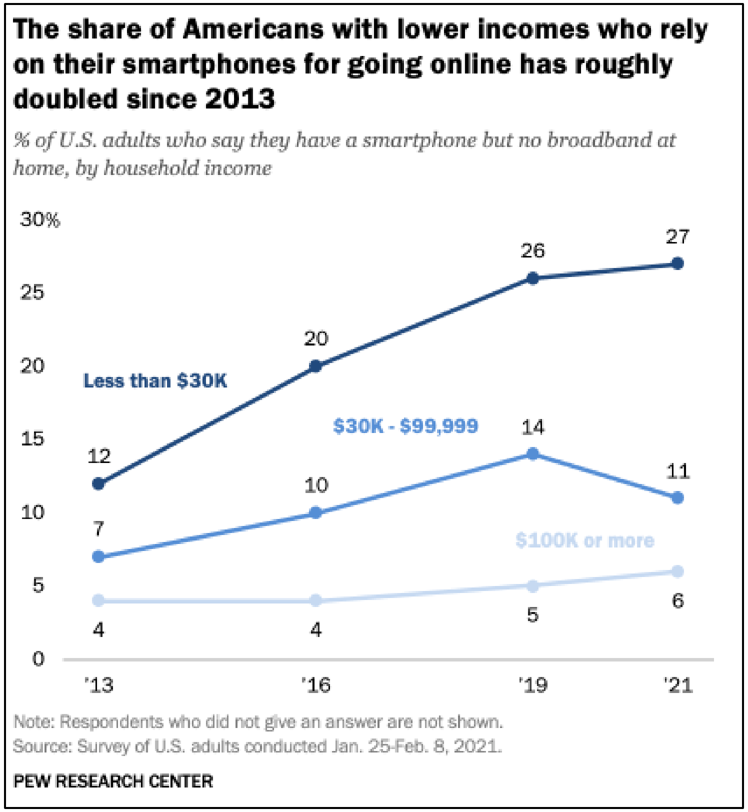


Americans with lower incomes have lower levels of technology adoption

% of U.S. adults who say they have each of the following, by household income

Note: Respondents who did not give an answer are not shown.
Source: Survey of U.S. adults conducted Jan. 25-Feb. 8, 2021.

PEW RESEARCH CENTER



The share of Americans with lower incomes who rely on their smartphones for going online has roughly doubled since 2013

% of U.S. adults who say they have a smartphone but no broadband at home, by household income

Note: Respondents who did not give an answer are not shown.
Source: Survey of U.S. adults conducted Jan. 25-Feb. 8, 2021.

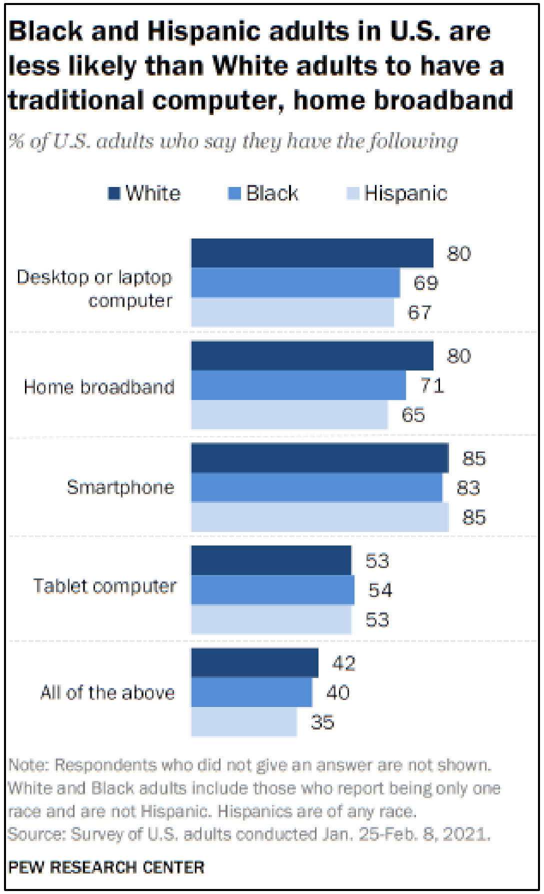
PEW RESEARCH CENTER

lower income Americans are far less likely to be online

many rely on smartphones for access

The Digital Divide (cont.)

similarly, inequities occur for those with minority ancestry or disabilities

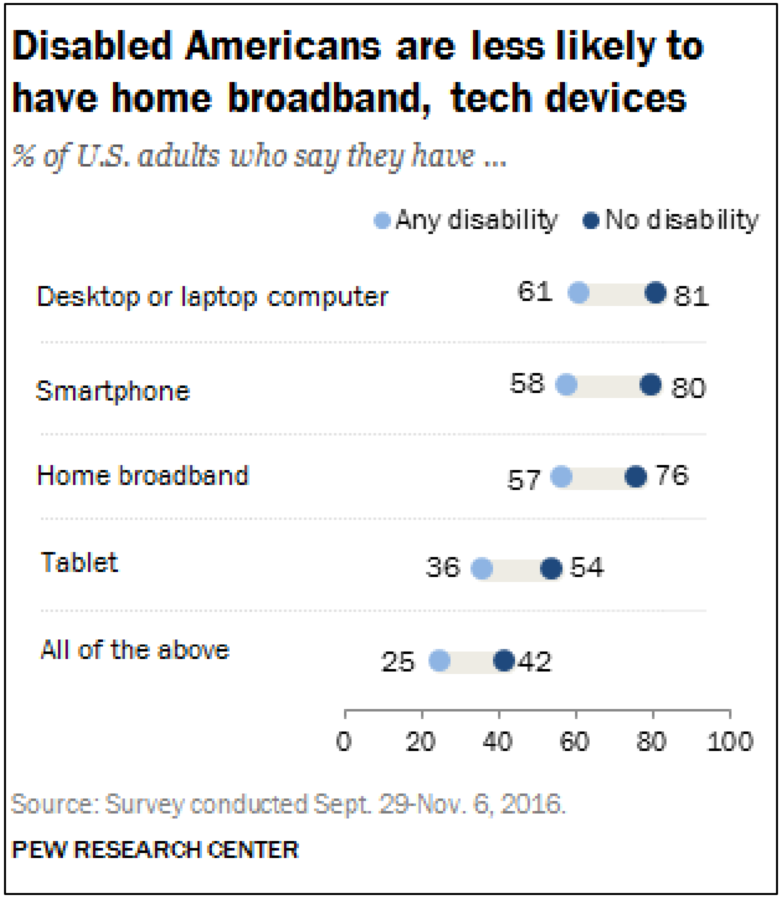


Black and Hispanic adults in U.S. are less likely than White adults to have a traditional computer, home broadband

% of U.S. adults who say they have the following

Note: Respondents who did not give an answer are not shown. White and Black adults include those who report being only one race and are not Hispanic. Hispanics are of any race.
Source: Survey of U.S. adults conducted Jan. 25-Feb. 8, 2021.

PEW RESEARCH CENTER



Disabled Americans are less likely to have home broadband, tech devices

% of U.S. adults who say they have ...

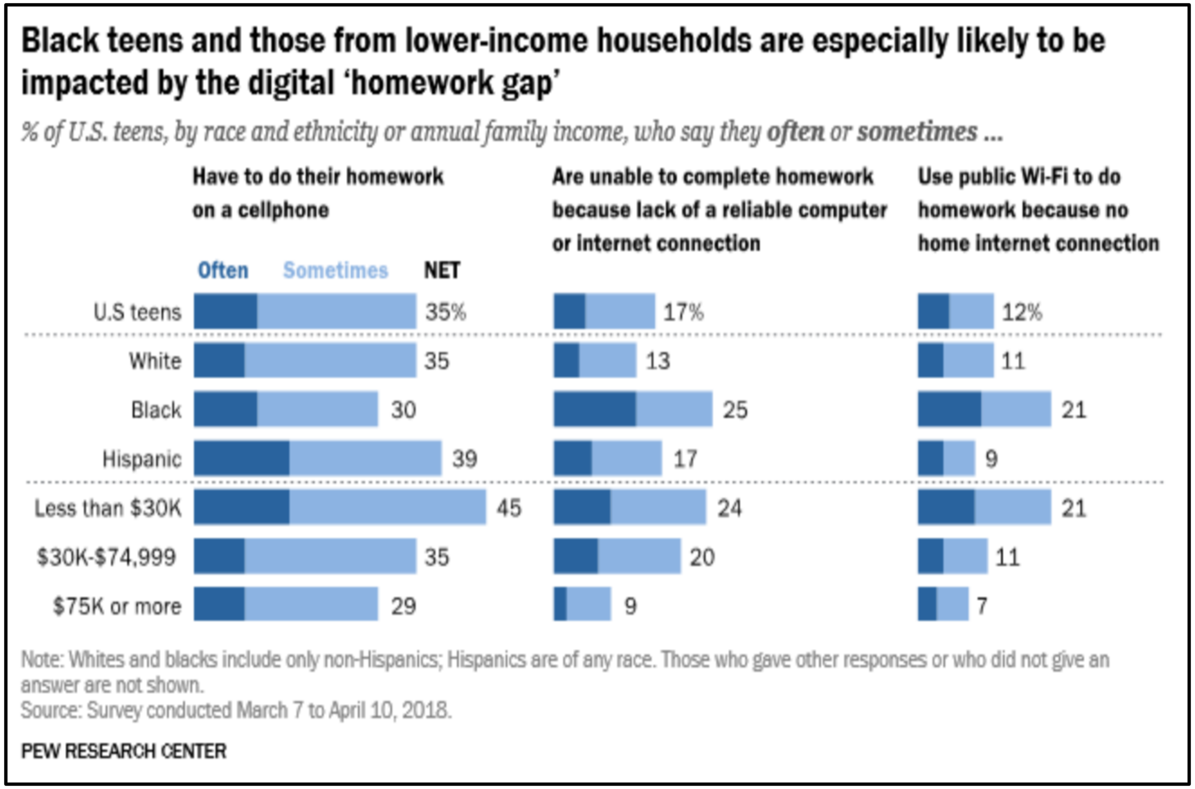
Source: Survey conducted Sept. 29-Nov. 6, 2016.

PEW RESEARCH CENTER

Homework Gap

the Digital Divide was highlighted during the 2020 COVID-19 pandemic

- many schools closed and students were transitioned to online classes
- those with home computers and reliable Internet were able to interact via videoconferencing software & complete online assignments
- those without were severely disadvantaged



Black teens and those from lower-income households are especially likely to be impacted by the digital 'homework gap'

*% of U.S. teens, by race and ethnicity or annual family income, who say they **often** or **sometimes** ...*

Note: Whites and blacks include only non-Hispanics; Hispanics are of any race. Those who gave other responses or who did not give an answer are not shown.
Source: Survey conducted March 7 to April 10, 2018.

PEW RESEARCH CENTER

Addressing Inequities

in late 1990s, U.S. government prioritized Internet connectivity

- by 2003, nearly all public schools were connected (vs. 35% in 1994)
- most public libraries provide free access to Internet-enabled computers
- programs (e.g., Lifeline, Emergency Broadband Benefit) subsidize cellular and Internet bills for low-income families

numerous not-for-profit organizations address the Digital Divide

- *National Digital Inclusion Alliance* coordinates hundreds of organizations
- *EveryoneOn* connects low-income families with affordable Internet plans and devices
- *Computers for Kids* takes old computers, refurbishes them, and donates them to schools and needy kids
- *Boys and Girls Club of America* offers educational programs

The Global Divide

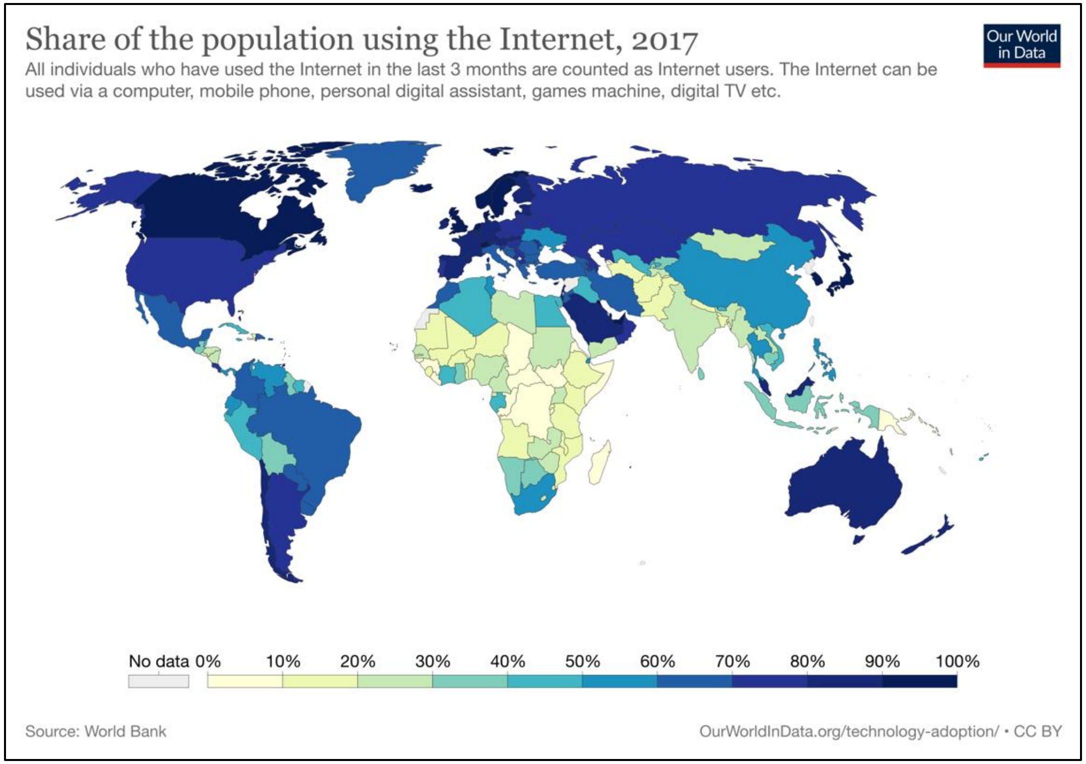
globally, wealthier countries are more likely to have computers/Internet

the U.N. and World Economic Forum lead numerous initiatives

- 2021 Generation Equality Forum raised $40 billion in new investments

many private sector initiatives as well

- One Laptop Per Child has donated more than 3 million low-power laptops



Share of the population using the Internet, 2017

All individuals who have used the Internet in the last 3 months are counted as Internet users. The Internet can be used via a computer, mobile phone, personal digital assistant, games machine, digital TV etc.

Source: World Bank

OurWorldInData.org/technology-adoption/ • CC BY

in 2014, 3 countries accounted for half of all Internet traffic

1. China (29%)
2. U.S. (13%)
3. Japan(8%)

Diversity in the Tech Sector

tech industry is one of the most dynamic & exciting sectors of the economy

- no other industry has the same potential for rapidly taking ideas and turning them into devices/applications that change the world
- e.g., cell phones & smartphones widely adopted within 10 years

many applications were the brainchild of a single person or small team

Word Wide Web	Tim Berners-Lee, 1990
Mosaic browser	Mark Andreesen & Eric Bina, 1994
Google	Larry Page & Sergey Brin, 1998
Facebook	Mark Zuckerberg, 2004
Twitter	Jack Dorsey, 2006
Instagram	Kevin Systrom & Mike Krieger, 2010

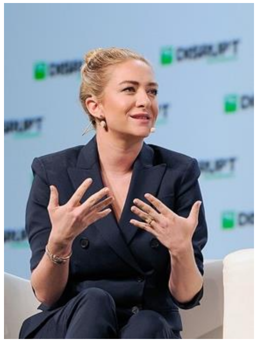
all these pioneers/innovators/entrepreneurs are white males

- women and people of color have been and continue to be underrepresented in the tech industry (especially in leadership)

Success Stories

certainly, there have been successful leaders and innovators

- Jerry Lawson (1940-2011): led development of first cartridge-based video game, founded VideoSoft in 1980.
- Marissa Mayer (1975-): first female engineer at Google, led in development of Google Maps, Google Earth & Street Maps. CEO of Yahoo! from 2012-2017.
- Luis von Ahn (1979-): founded digital security company reCAPTCHA. Also founder and CEO of DuoLingo.
- Whitney Wolfe Herd (1989-): founded Bumble, youngest woman to take her company public on NYSE & youngest female self-made billionaire.



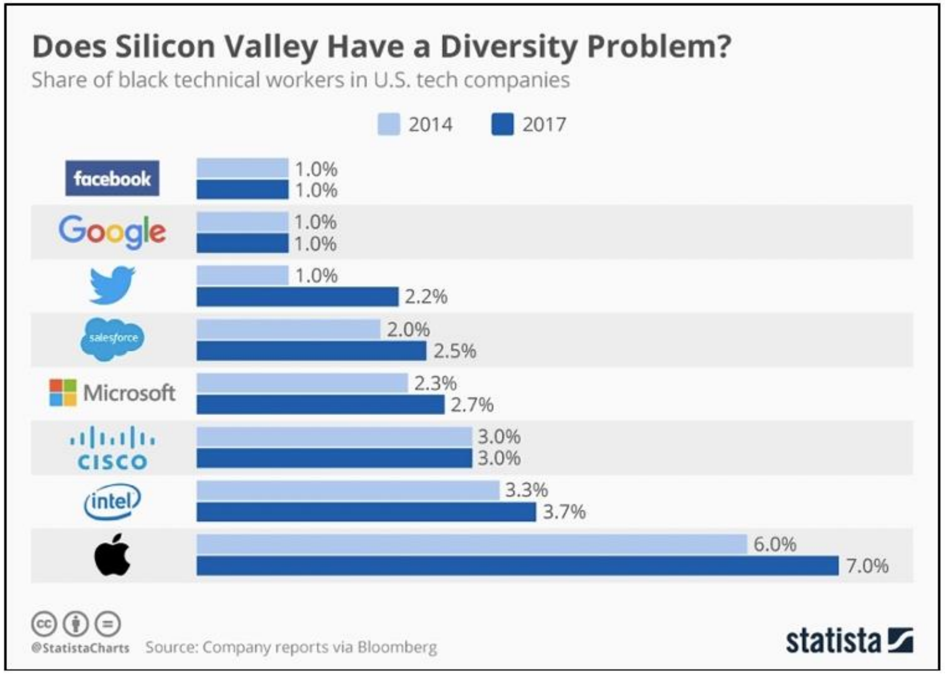
Underrepresentation

in 2020, women were > 50% of U.S. workforce, but only 26% of tech jobs

- only 7% of Fortune 500 CEOs were women

even greater underrepresentation for minority groups

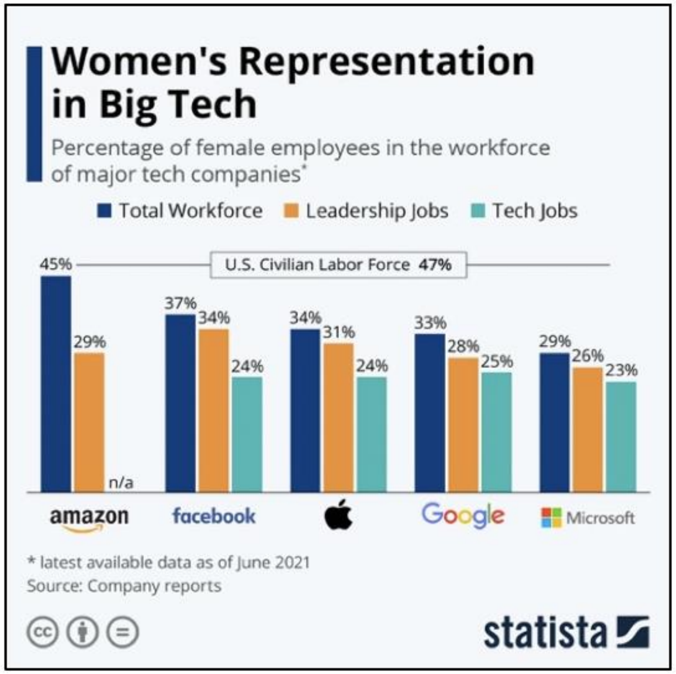
- Hispanic/Latinx Americans: 18% of workforce, 8% of tech jobs
- African Americans: 13% of workforce, 5% of tech jobs



Does Silicon Valley Have a Diversity Problem?

Share of black technical workers in U.S. tech companies

Source: Company reports via Bloomberg



Women's Representation in Big Tech

Percentage of female employees in the workforce of major tech companies*

* latest available data as of June 2021

Source: Company reports

Increasing Diversity

tech companies are addressing equity/diversity, but change has been slow

- changing company culture to be more inclusive
- changing hiring practices to bring in more diversity
- creating mentoring programs for women & minority employees

one problem is that stereotypes & misinformation influence people when they are young

- TV & movies perpetuate white/Asian male sterotypes
- access to computers in schools is not equitable
- among 2020 college graduates with major in computer science:
    - 18% were women, 10% were Hispanic/Latinx, 9% were African American

many initiatives are focused on providing access/mentoring for young people

- e.g., National Center for Women & Information Technology (NCWIT), AnitaB.org, Code.org, Black Girls Code, and Hispanics in Computing
- AP CS Principles course launched in 2017, specifically designed to increase participation by women and minorities

Benefits of Diversity

studies have shown numerous benefits of diversity in the workplace

1. companies are more productive when teams are diverse in background/perspective
2. different viewpoints & experiences produce a diversity of ideas, which can encourage innovation
3. an openness to workers who look and think differently means outstanding candidates who might have been overlooked are now hired
4. a diverse workforce is better at understanding and responding to a diverse client base
5. the products designed and built by a diverse team tend to be higher quality and more equitable

    e.g., cell phone interfaces designed by men that are difficult to use by women with smaller hands

    e.g., 5-year gap between release of Apple emojis and emojis with different skin tones

Algorithmic bias

algorithmic bias occurs when faulty assumptions and poor design practices lead to software that discriminates

1. in 2015, Amazon announced it was abandoning interview software that was proven to discriminate against women candidates
2. in 2016, ProPublica published a report documenting how sentencing software used in courts discriminated against minority suspects
3. in 2019, Science published a report documenting how hospital software for screening patients discriminated against minority and low-income patients
4. Several studies have confirmed that facial recognition software misidentifies women and people of color disproportionally, leading to more arrests due to use by law enforcement.

in all these cases, algorithmic bias was not intentional

- lack of diversity led to unquestioned assumptions & flawed development process

Bridging the Divide

when combating any form of discrimination and inequity, *awareness* is the first step

in the past decade, awareness of the Digital Divide has led to change

- governments and companies are leading equity efforts
- young women and people of color are being encouraged to consider future careers in computing
- recognition of toxic corporate cultures & unfair hiring practices are producing change (albeit slowly)
- recognition of the dangers of algorithmic bias is leading to new design and testing methodologies