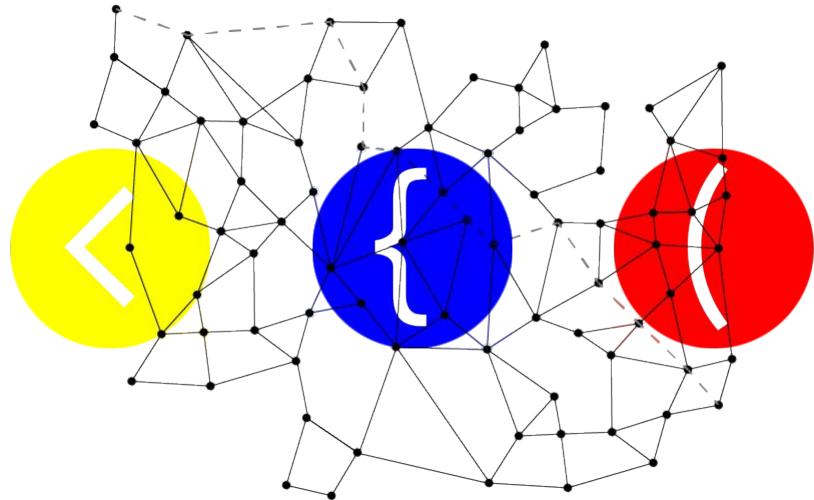
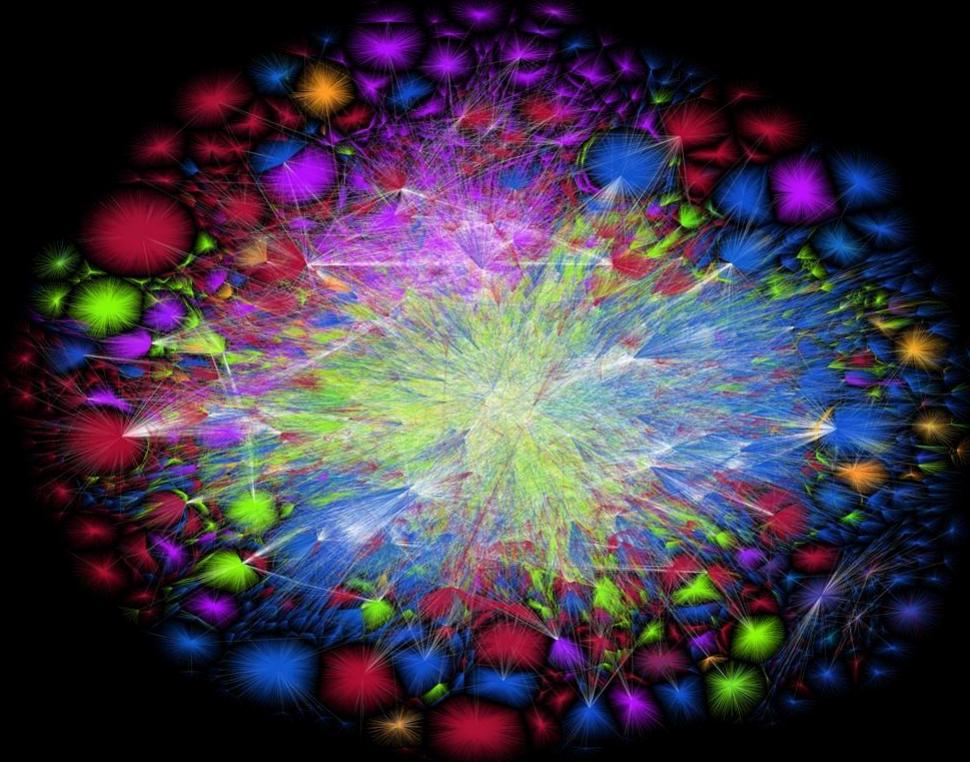


Introduction to Web Design

Introduction and Overview





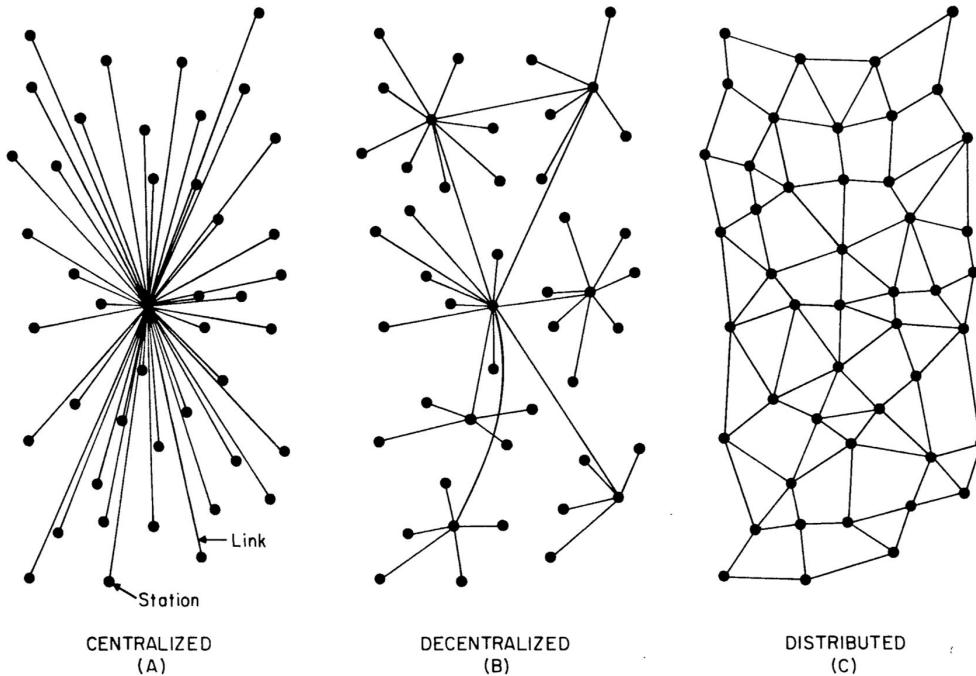
2015. Opte Project

Introduction to Web Design

What is the Internet?

Introduction and Overview

A computer network consisting of a worldwide network of computer networks that use standardized network protocols to facilitate data transmission and exchange



1964, On Distributed Communications

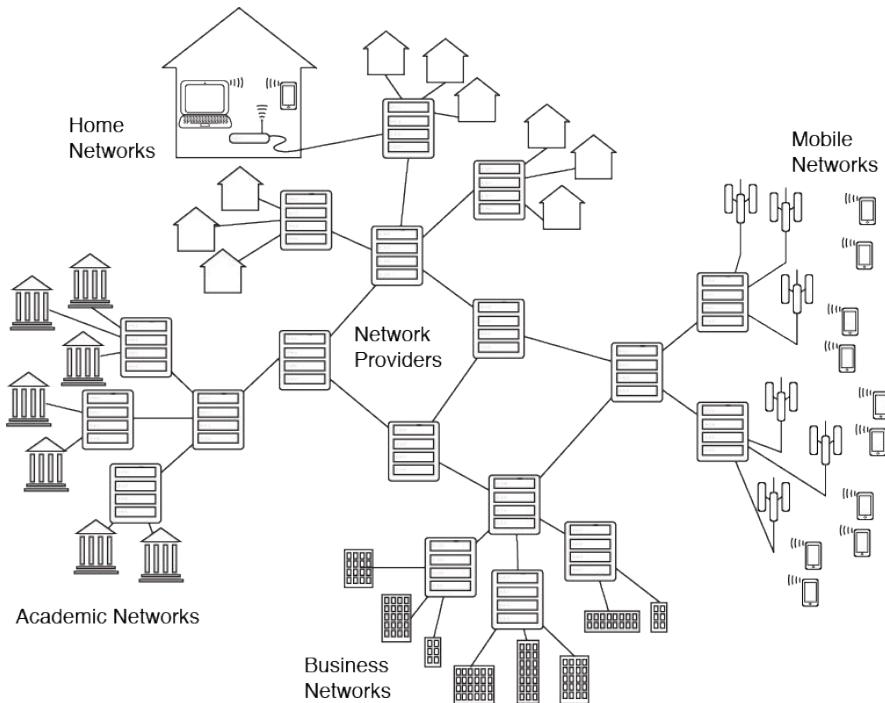
Introduction to Web Design

Centralized, Decentralized, and Distributed Networks

Introduction and Overview

A decentralized network represents a less-hierarchical structure than a centralized network. Complete reliance on a single point is not required.

The foundational concept of decentralized networks would be deployed in tandem with what came to be known as “packet-switching,” which entails breaking up communications into small parts, sending them along, and reconstructing them at the end.



[Simplified model of the internet. It's made of routers.](#)

Introduction to Web Design

Routers

Introduction and Overview

A router is a networking device that relays data packets between computer networks.

Routers direct the flow of Internet traffic so that packets arrive at their appropriate destination.

The address to which data is sent is normally in the form of a numeric IP address (IP stands for Internet Protocol).

Introduction to Web Design

The Internet and the World Wide Web

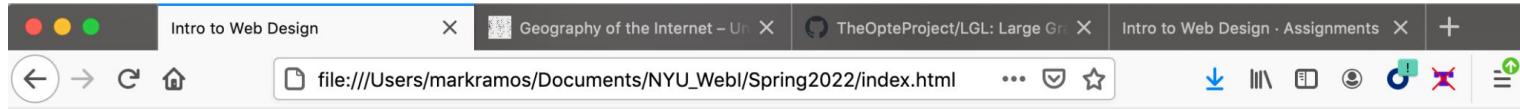
Introduction and Overview

The Internet and the Web are separate but related things.

The Internet is a massive network of networks, a networking infrastructure that connects computers globally.

The Web is a way of accessing information over the medium of the Internet, an information sharing model that is built on top of the Internet.

The Web is just one of the ways that information can be disseminated over the Internet but it is the one we are focused on in this class.



Introduction to Web Design and Computer Principles

Syllabus

CSCI-UA 4-4

Class Notes

Monday/Wednesday, 11:00

Assignments

a.m.–12:15 p.m.

Resources

Room 808, Kimmel Center, 60

Washington Sq South

Professor: Mark Ramos

Overview

There are two primary aspects to this course. The first is learning how to build websites and prepare the various elements that comprise them.

The second is understanding concepts behind computers in general and the web in particular.

Grading and Exams

The following rubric serves as a guideline for how grades will be calculated. The final percentage allotted to each category is at the instructor's discretion.

- Assignments: 40%
- Final Project: 10%
- Midterm Exam: 20%
- Final Exam: 30%

Our exam schedule for the semester is as follows.

TBA



on the internet, no one knows you're a cat.

© 2000 jumble

Introduction to Web Design

The New Dark Web

Introduction and Overview

In many ways we are experiencing the afterglow of the technological promise of freedom and openness.

Networked tools and digital media still offer lots of possibilities but also significant problems.

What are some of the dystopian aspects of the Internet and the web today?

Introduction to Web Design

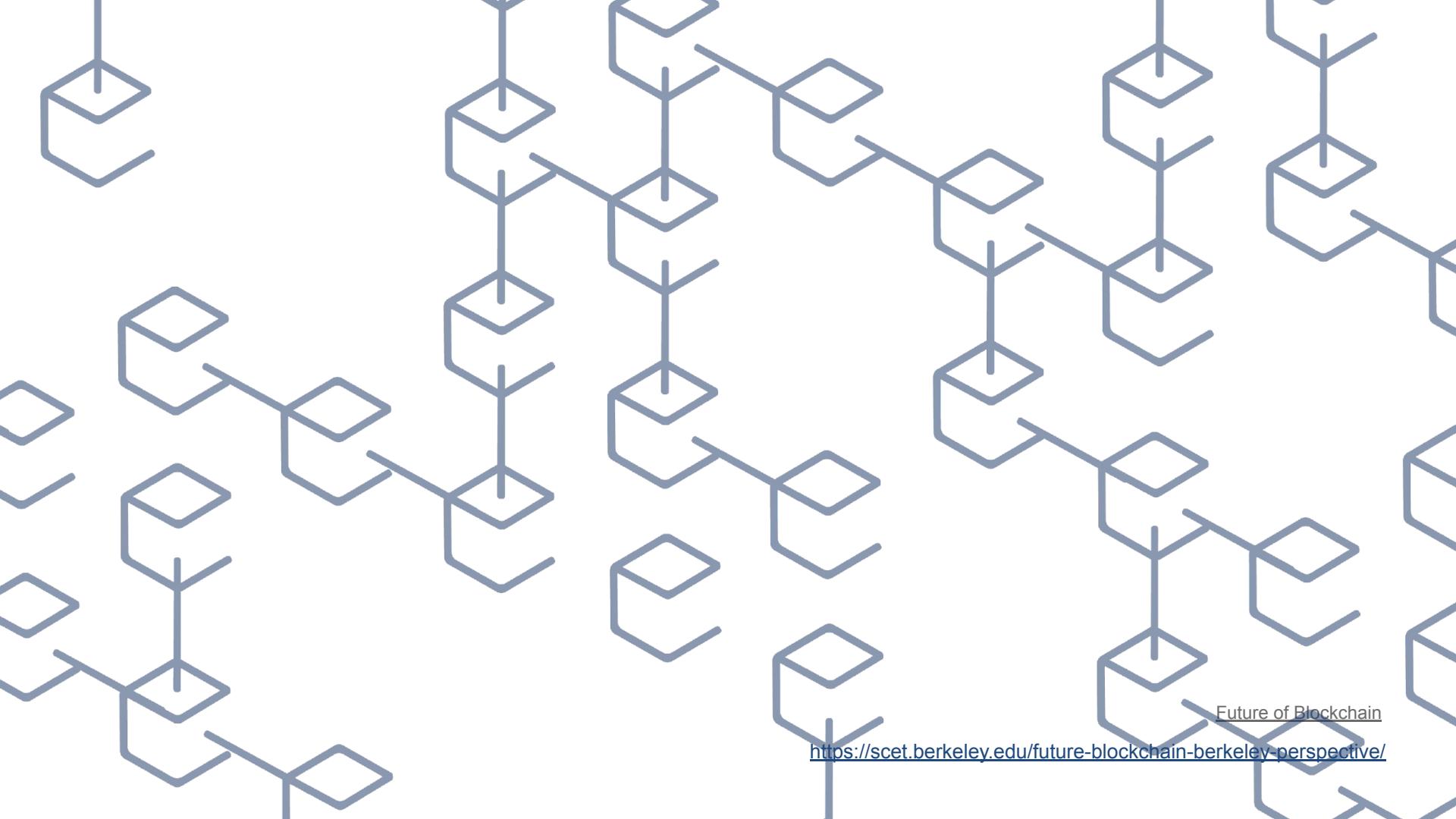
Re-Decentralization of the Web

Introduction and Overview

“A new Decentralized Web has the potential to be open, empowering users around the globe to control and protect their own personal data better than before.”

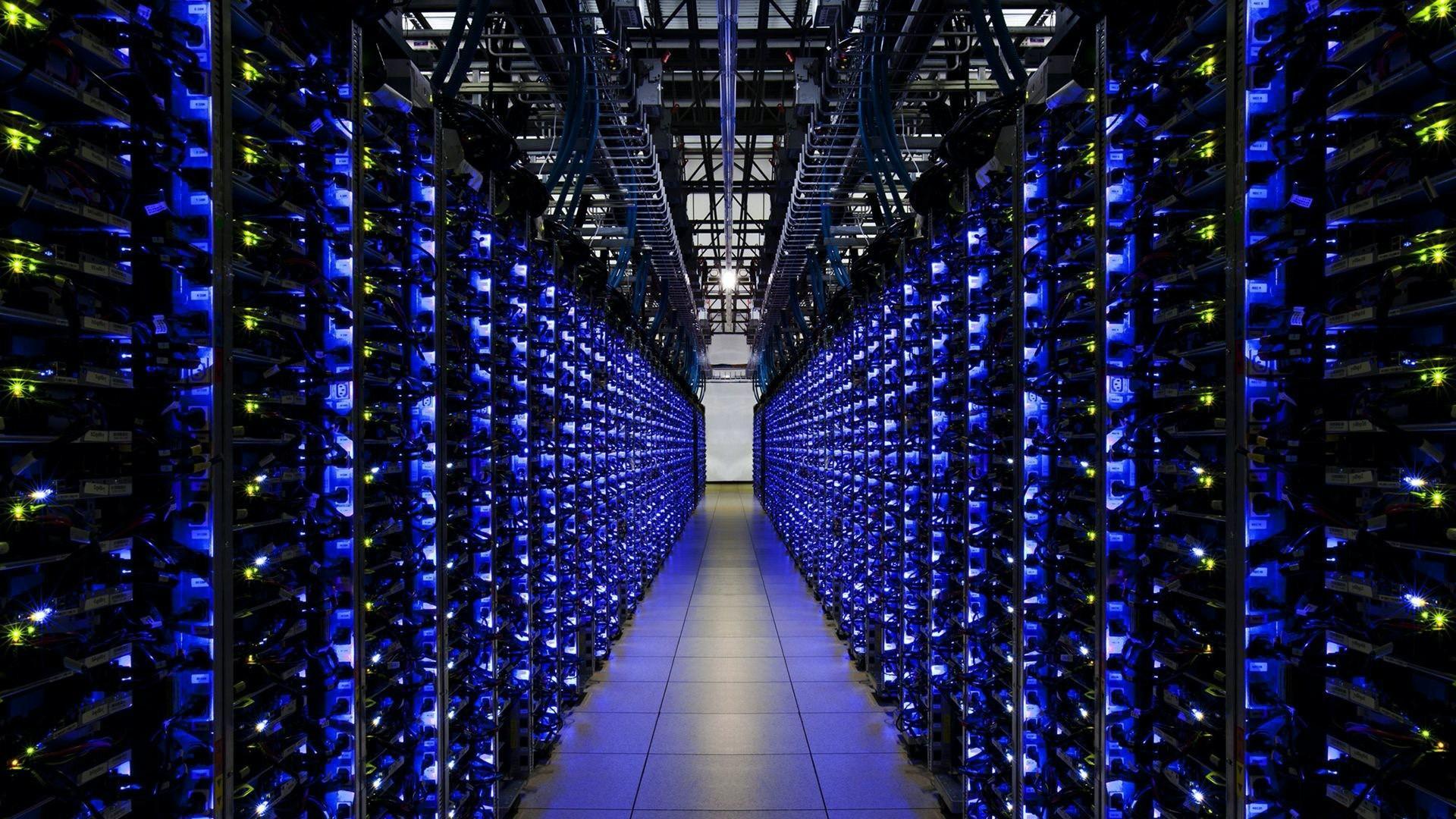
—Decentralized Web Summit

AKA “web 3.0”



Future of Blockchain

<https://scet.berkeley.edu/future-blockchain-berkeley-perspective/>



1

0

Introduction to Web Design

Digital Media Storage

Introduction and Overview

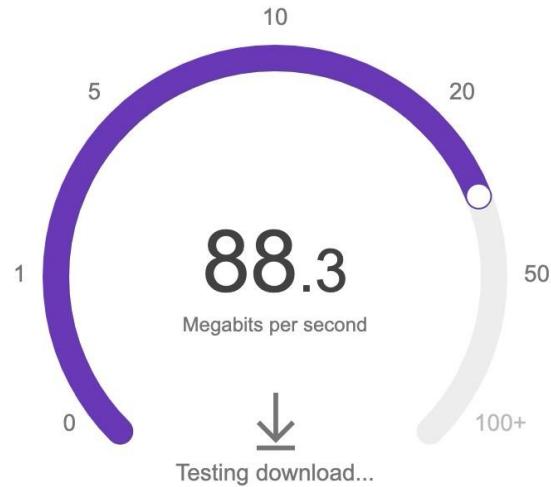
On/Off

Electrical impulses (+5v / -5v)

- Single 0 or 1 = 1 “bit”
- A group of 8 bits = 1 “byte”
- 1 million bytes ≈ 1 “megabyte”
- 1,024 megabytes = 1 “gigabyte”
- 1,000 gigabytes = 1 “terabyte”

00101011

+



Mbps download

Mbps upload

CANCEL

Introduction to Web Design

Digital Media Transfer

Introduction and Overview

Internet connection speed is normally measured in megabits.

Megabits (Mb) are not the same as megabytes (MB).

8 bits = 1 byte; therefore, a megabyte is 8 times the size of a megabit.

For fixed broadband connections, the average download speed in the United States is around 96 Mb/second; average upload speed is around 33 Mb/s

For mobile connections, the average download speed in the United States is around 34 Mb/second; average upload speed is around 10 Mb/s

[Speedtest Reports](#)



Introduction to Web Design

Introduction and Overview



Introduction to Web Design

Introduction and Overview



Introduction to Web Design

Introduction and Overview



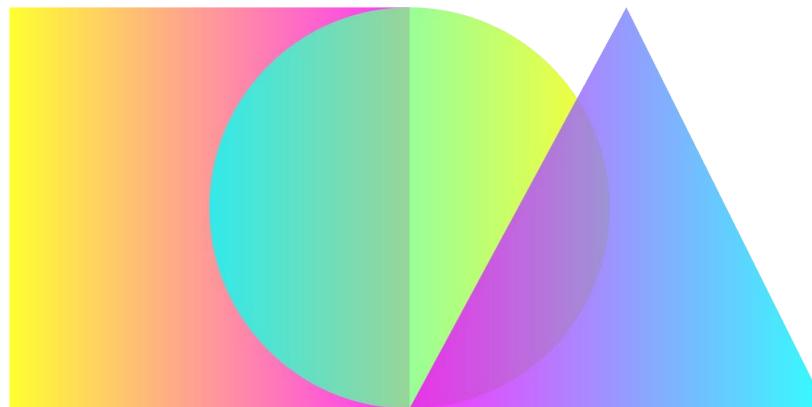


Introduction to Web Design

Introduction and Overview

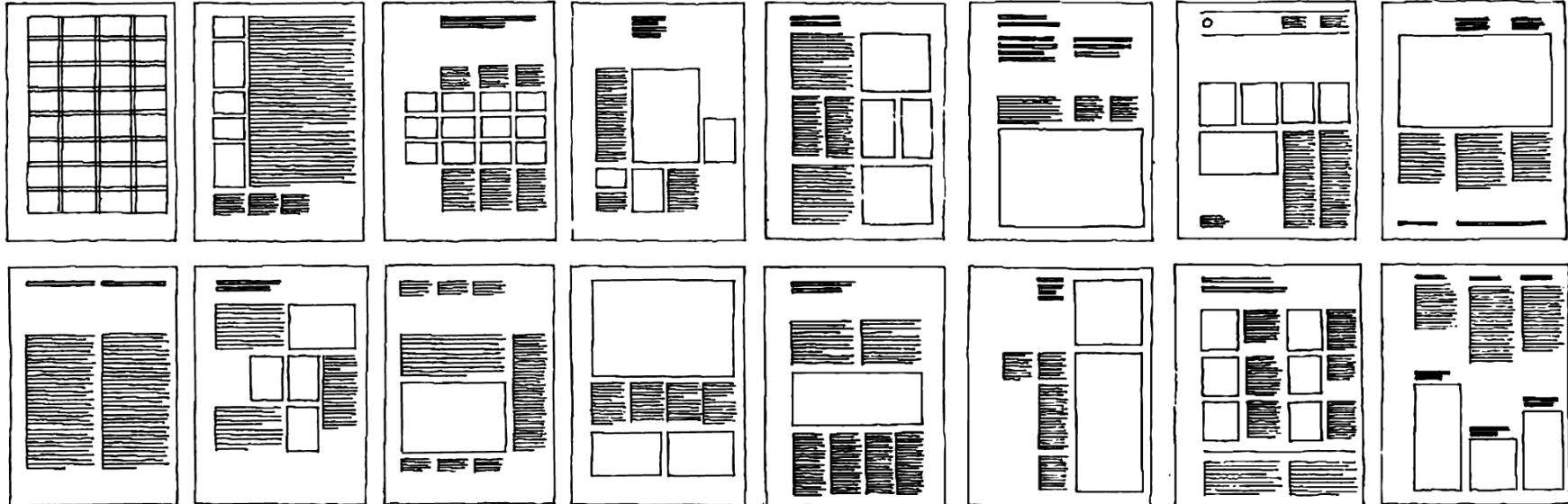
Introduction to Web Design

Introduction and Overview



Introduction to Web Design

Introduction and Overview



Introduction to Web Design

Introduction and Overview



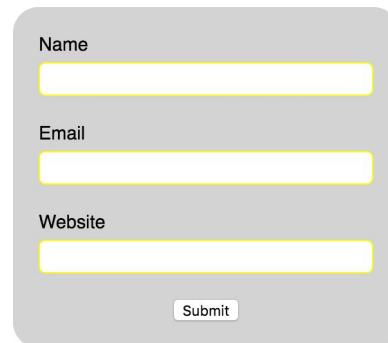
Introduction to Web Design

Introduction and Overview



Introduction to Web Design

Introduction and Overview



Name

Email

Website

Introduction to Web Design

Introduction and Overview



Introduction to Web Design

Introduction and Overview





Introduction to Web Design



Introduction and Overview

Introduction to Web Design

Course Outline

Introduction and Overview

Unix command line

HTML

CSS

Web graphics

Design and accessibility

Website layout and responsive design

Interactivity with JavaScript

Web forms

Web audio and video

Version control

Web hosting and domain names

Introduction to Web Design

Guiding Principles: Open Source

Introduction and Overview

- Anyone is free to use it
- Usually free of charge
- Source code is made available
- Can be modified and redistributed

Introduction to Web Design

Guiding Principles: Accessibility and Net Neutrality

Introduction and Overview

"When we talk about accessible code, what we are really talking about at its core is inclusiveness. . . . Inclusive development means making something valuable, not just accessible, to as many people as we can." —Carie Fisher

Net neutrality is the principle that Internet service providers should enable access to all content and applications regardless of the source, and without favoring or blocking particular products or websites.

Introduction to Web Design

Guiding Principles: Web Standards

Introduction and Overview

The formal, non-proprietary standards and technical specifications that define and describe aspects of the World Wide Web and its interoperability.

These include:

- HTML
- CSS
- JavaScript
- SVG
- WOFF

Introduction to Web Design

Introductions

Introduction and Overview

Mark Ramos(he/him/his)
Adjunct Professor

mr6465@nyu.edu

Office hours:
• Thursday, 3:00–6:00 p.m.

Pls. email me to schedule a meeting over Zoom

Introduction to Web Design

Class Format

Introduction and Overview

- Synchronous, interactive lectures
- Asynchronous, assignments and exercises
- In-person and remote tutoring with the Computer Science Department tutors

Introduction to Web Design

Attendance

Introduction and Overview

You are expected to keep up with all classes.

You are encouraged to participate to the fullest extent you're able to.

If you ever feel overwhelmed or need extra help, we will be available to you.

Introduction to Web Design

Tutoring

Introduction and Overview

Tutors will be available in person (for those on campus) and remotely (for those on and off campus).

You are required to sign up for in-person tutoring sessions through NYU Connect.

Remote tutoring is on a drop-in basis and does not require signing up.

Introduction to Web Design

Required Textbook

Introduction and Overview

Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics

5th Edition

Jennifer Robbins

ISBN: 978-1-491-96020-2

Introduction to Web Design

Assignment S

Introduction and Overview

There will be nine assignments over the course of the semester.

Details of each will be posted on the class website.

All assignments are due before class and should be submitted via NYU Classes.

Do your best to turn work in on time; 10% will be deducted for each class day after the deadline.

No assignments will be accepted after three classes or after the final exam.

Introduction to Web Design

Grading Rubric

Introduction and Overview

Assignments: 40%

Final Project: 10%

Midterm Exam: 20%

Final Exam:
30%

Introduction to Web Design

Next

Introduction and Overview

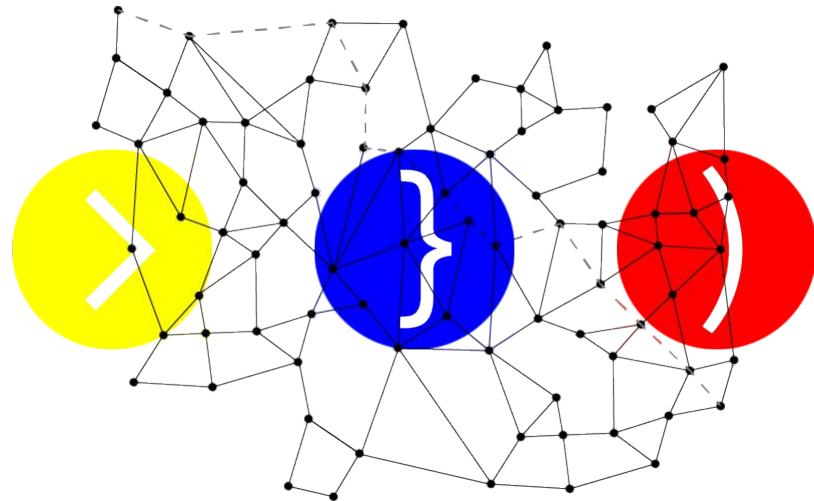
Review class website:

<https://cs.nyu.edu/courses/spring22/CSCI-UA.0004-004/syllabus/>

Read chapter 2 of *Learning Web Design*:
“How the Web Works”

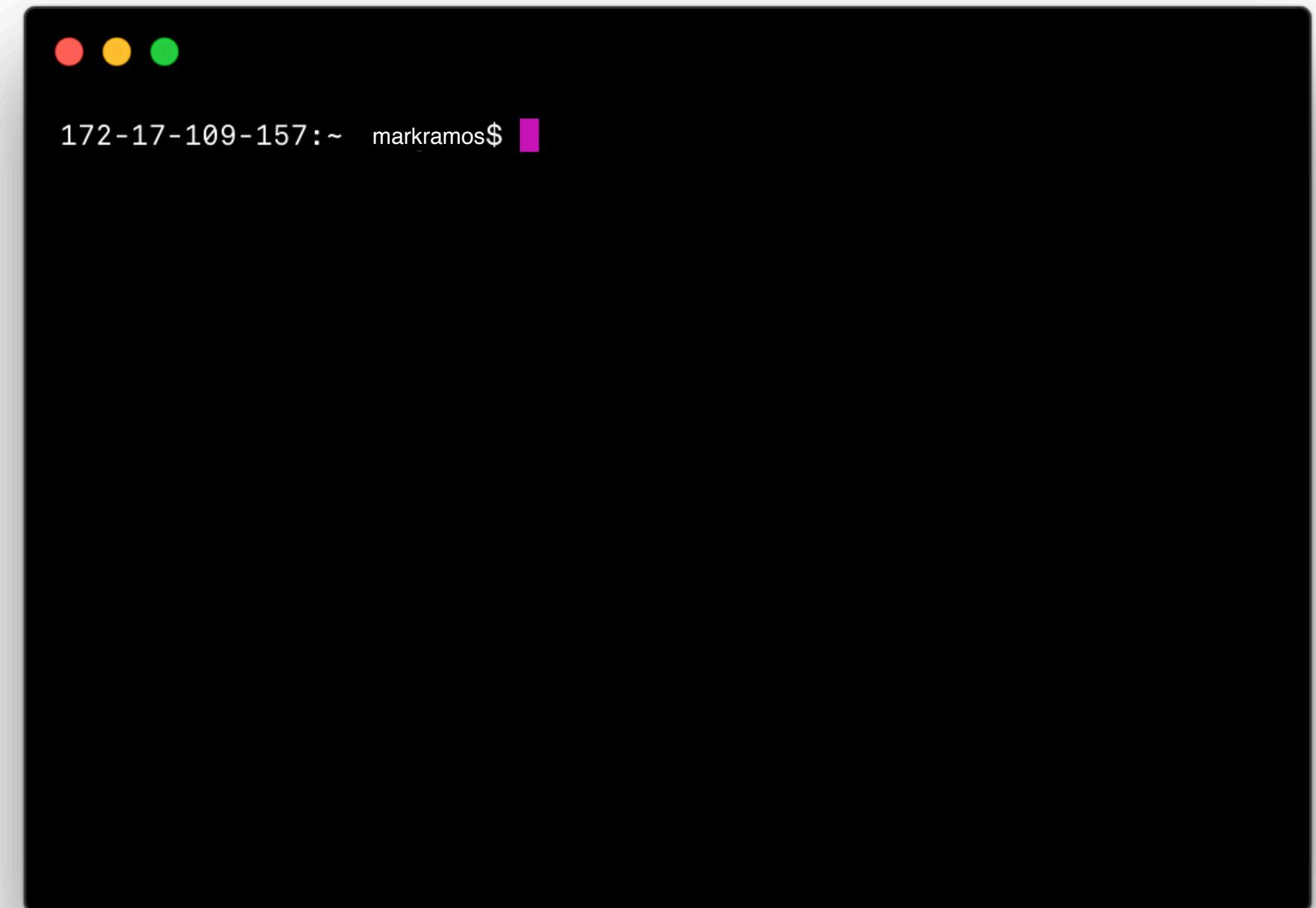
Introduction to Web Design

Introduction and Overview



Introduction to Web Design

Operating Systems



Introduction to Web Design

What is an operating system?

Operating Systems

Software that manages a computer's resources

Allocates resources among other programs

Resources include the central processing unit (CPU), computer memory, file storage, input/output (I/O) devices, and network connections

Runs indefinitely and terminates only when the computer is turned off

Introduction to Web Design

Contemporary Examples

Operating Systems

Microsoft OS

Mac OS

Linux

iOS

Android

Symbian OS

Introduction to Web Design

History

Operating Systems

First digital computers had no operating systems

Ran one program at a time, which had command of all system resources

A human operator would provide any special resources needed

First operating systems were developed in the mid-1950s

Introduction to Web Design

OS Interface

Operating Systems

The graphical user interface (GUI) is most familiar to us.

It provides visual metaphors for the operations we perform on a computer.

A command line interface (CLI), on the other hand, receives typed commands for each operation.

Introduction to Web Design

Unix

Operating Systems

An open source OS produced by AT&T Bell Labs

Originally developed in 1969

Command line interface

Portable, multi-tasking, multi-user

Free distribution, open system

Servers (including i6), workstations, mobile devices

Basis of Linux and MacOS

Introduction to Web Design

Unix Commands

Operating Systems

\$ ls

\$ pwd

\$ cd

\$ cp

\$ rm

See the course website for more basic Unix commands.

Introduction to Web Design

chmod Command

Operating Systems

Every file and directory has nine permissions associated with it

The Unix chmod command sets permissions of files and directories

Files and directories have three types of permissions (or none):

- r** (read)
- w** (write)
- x** (execute)
- (no permission)

The above permissions occur for each of the following classes or users:

- u** (user/owner)
- g** (group)
- o** (other/world)

Introduction to Web Design

Operating Systems

Permissions

U G W

rwx rwx rwx

rwx rwx r-x

rwx r-x r-x

rw- rw- r--

rw- r-- r--

Unix Commands

\$ chmod 777 filename

\$ chmod 775 filename

\$ chmod 755 filename

\$ chmod 664 filename

\$ chmod 644 filename

Introduction to Web Design

Standard Website Permissions

Operating Systems

Standard file permission:
644

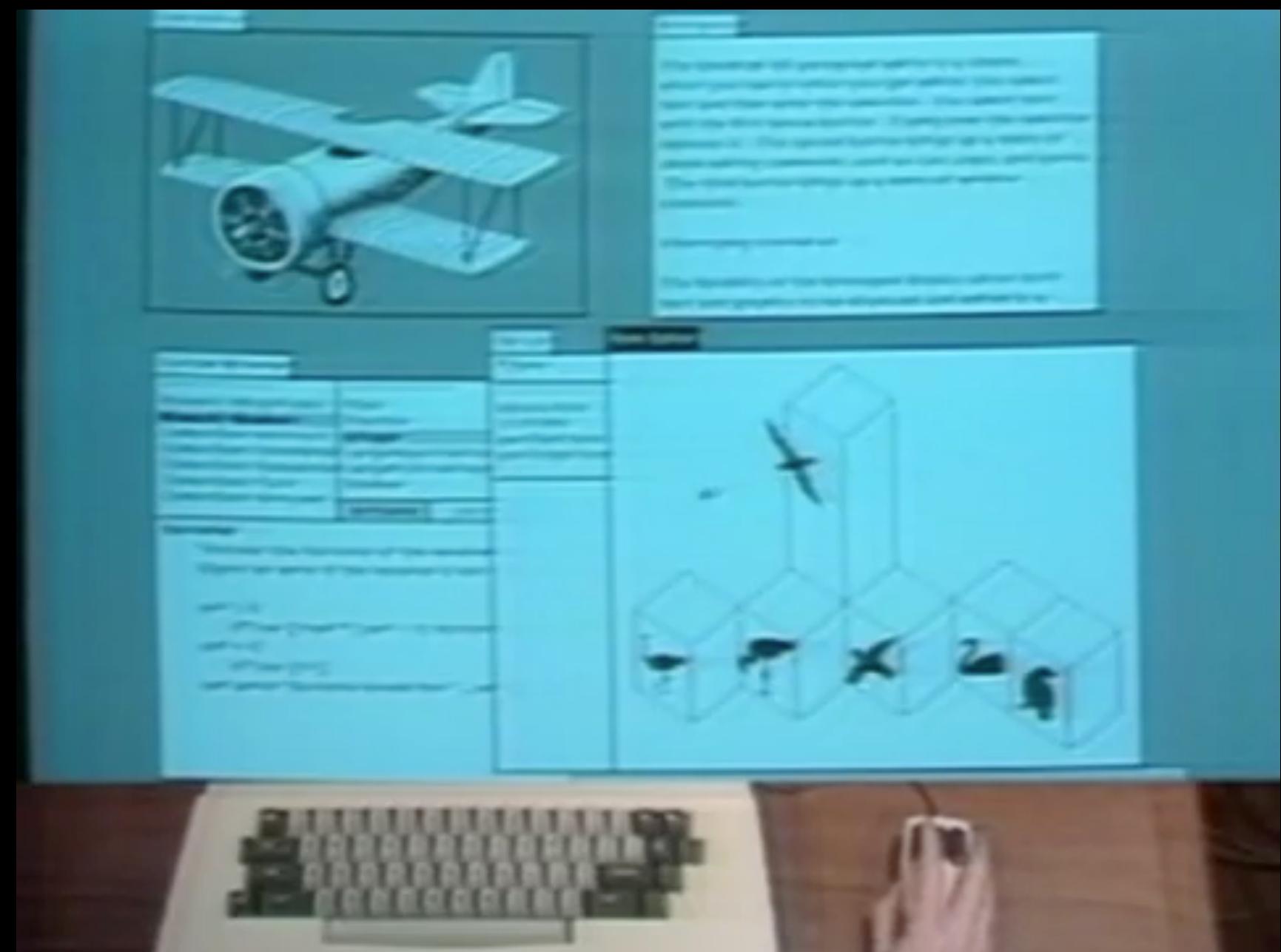
Owner can read and write file;
group can read file;
others can read file

Standard directory permission:
755

Owner can read, write and execute file;
group can read and execute file;
others can read and execute file

Introduction to Web Design

Operating Systems



Triumph of the Nerds, 1996, PBS

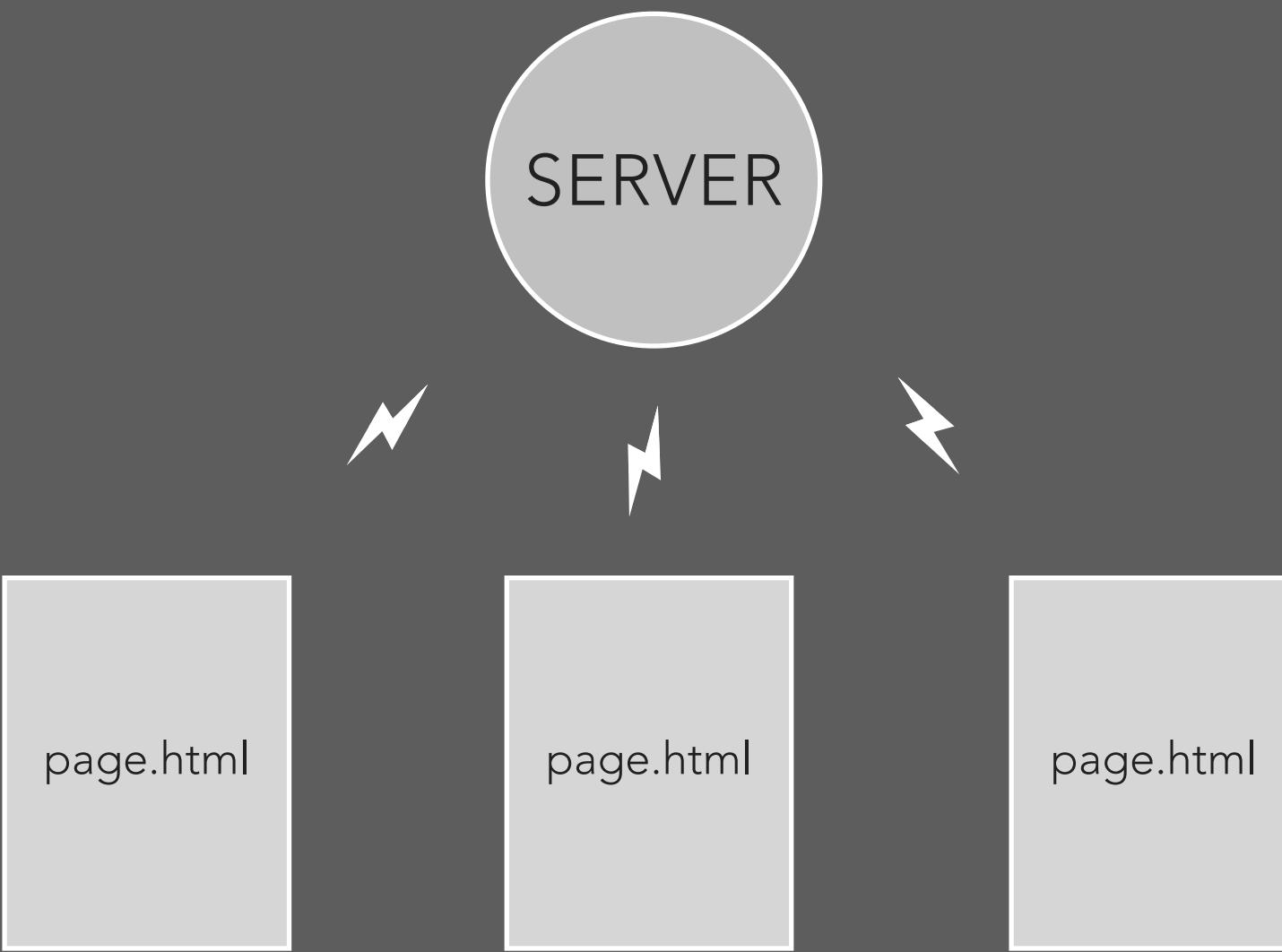


Palo Alto Research Center, Incorporated, Courtesy of the PARC Library, 1970 ca.

A close-up portrait of a woman with voluminous, curly hair. She has a warm, reddish-orange hue to her skin and hair, possibly from lighting or a filter. Her gaze is directed towards the viewer, and she has a slight, pleasant smile. The background is dark and out of focus.

Let's get started!

Static Pages / Dynamic Pages



SERVER

page.html

page.html

page.html

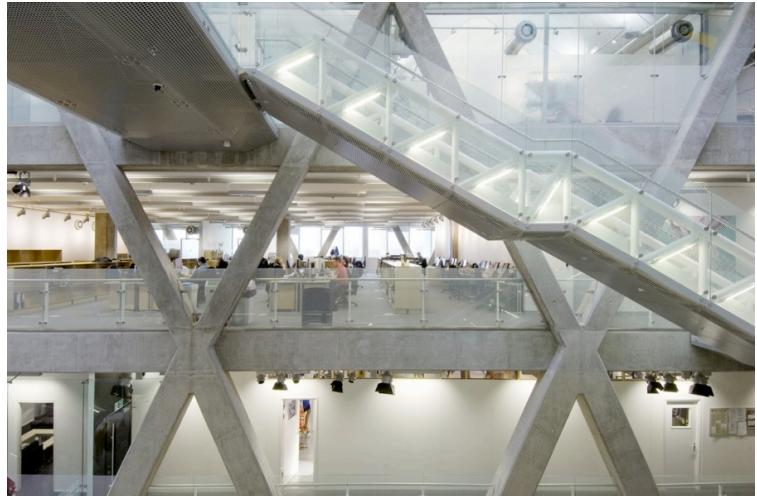
A static website is a group of self-contained, individual pages (or page), sent to the browser from the server one-page-at-a-time.

HTML, CSS, Javascript

Three layers of web design:
Structure, Style, Behavior

BEHAVIOR

Javascript



PRESENTATION

CSS

Imagery



STRUCTURE

HTML markup

Site planning



```
<div style='clear: both;'></div>
</div>

<div id='nav-bar-outer'>

    <div id='nav-logo-borderfade'><div class='nav-fade-mask'></div><div class='nav-fade nav-sprite'></div></div>

    <div id='nav-bar-inner' class='nav-sprite'>

        <a id='nav-shop-all-button' href='/gp/site-directory/ref=topnav_sad' class='nav_a nav-button-outer nav-menu-inactive' alt='Shop All' data-value="search-alias=aps">
            <span class='nav-button-mid nav-sprite'>
                <span class='nav-button-inner nav-sprite'>
                    <span class='nav-button-title nav-button-line1'>Shop by</span>
                    <span class='nav-button-title nav-button-line2'>Department</span>
                </span>
            </span>
            <span class='nav-down-arrow nav-sprite'></span>
        </a>

        <label id='nav-search-label' for='twotabsearchtextbox'>
            Search
        </label>

        <div>
            <form
                action='/s/ref=nb_sb_noss'
                method='get' name='site-search'
                class='nav-searchbar-inner'
            >

                <span id='nav-search-in' class='nav-sprite'>
                    <span id='nav-search-in-content' data-value="search-alias=aps">
                        All
                    </span>
                    <span class='nav-down-arrow nav-sprite'></span>
                    <select name="url" id="searchDropdownBox" class="searchSelect" title="Search in" data-value="search-alias=aps">
                        <option value="search-alias=instant-video">Amazon Instant Video</option>
                        <option value="search-alias=appliances">Appliances</option>
                        <option value="search-alias=arts-crafts">Arts, Crafts & Sewing</option>
                        <option value="search-alias=automotive">Automotive</option>
                        <option value="search-alias=baby">Baby</option>
                        <option value="search-alias=beauty">Beauty</option>
                        <option value="search-alias=stripbooks">Books</option>
                        <option value="search-alias=accessories">Accessories</option>
                        <option value="search-alias=apparel">Clothing & Accessories</option>
                        <option value="search-alias=collectibles">Collectibles</option>
                        <option value="search-alias=computers">Computers</option>
                        <option value="search-alias=electronics">Electronics</option>
                        <option value="search-alias=gift-cards">Gift Cards</option>
                        <option value="search-alias=grocery">Grocery & Gourmet Food</option>
                        <option value="search-alias=hpc">Health & Personal Care</option>
                        <option value="search-alias=kitchen">Kitchen</option>
                        <option value="search-alias=industrial">Industrial & Scientific</option>
                        <option value="search-alias=jewelry">Jewelry</option>
                        <option value="search-alias=store">Store</option>
                        <option value="search-alias=magazines">Magazine Subscriptions</option>
                        <option value="search-alias=movies-tv">Movies & TV</option>
                        <option value="search-alias=downloads">Downloads</option>
                        <option value="search-alias=popular">Music</option>
                        <option value="search-alias=mi">Musical Instruments</option>
                        <option value="search-alias=products">Products</option>
                        <option value="search-alias=lawngarden">Patio, Lawn & Garden</option>
                        <option value="search-alias=pets">Pet Supplies</option>
                        <option value="search-alias=shoes">Shoes</option>
                        <option value="search-alias=software">Software</option>
                        <option value="search-alias=sporting">Sports & Outdoors</option>
                        <option value="search-alias=home-improvement">Home Improvement</option>
                        <option value="search-alias=toys-and-games">Toys & Games</option>
                        <option value="search-alias=videogames">Video Games</option>
                        <option value="search-alias=watches">Watches</option>
                    </select>
                </span>
            </div>
            <div class='nav-searchfield-outer nav-sprite'>
                <div class='nav-searchfield-inner nav-sprite'>

```

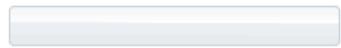
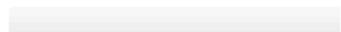
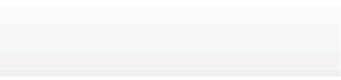
amazon amazon



Cart
Prime

amazon

PrimeFresh



[Amazon Try Prime](#)

- [Your Amazon.com](#)
- [Today's Deals](#)
- [Gift Cards](#)
- [Sell](#)
- [Help](#)

[Shop by Department](#) Search

All

All Departments



Go

Hello. Sign in Your Account Try Prime Cart 0 Wish List

•

•

Shop by
Department

Search

All ▾

Go

Hello, Shawn
Your Account ▾

Cart ▾

Wish
List ▾

Unlimited Instant Videos

MP3s & Cloud Player

20 million songs, play anywhere

Amazon Cloud Drive

5 GB of free storage

Kindle

Appstore for Android

Get Quell Reflect for free today

Digital Games & Software

Audible Audiobooks

Books

Movies, Music & Games

Electronics & Computers

Home, Garden & Tools

Grocery, Health & Beauty

Toys, Kids & Baby

Clothing, Shoes & Jewelry

Sports & Outdoors

Automotive & Industrial

> Full Store Directory

Dear Customers,



Today we're excited to introduce the all-new Kindle Fire HD.

Kindle Fire HD 8.9" 4G
\$499Kindle Fire HD 8.9"
\$299Kindle Fire HD
\$199

[Kindle Fire HD 8.9" 4G](#) isn't just the best tablet for the price, it's the best tablet. \$499 now gets you a large-screen HD tablet with a stunning 8.9" display, exclusive Dolby audio, dual stereo speakers, the fastest Wi-Fi, ultra-fast 4G LTE wireless, plus our new unprecedented \$49.99 one-year 4G data package. Customers save hundreds of dollars in the first year compared to other 4G tablets. Kindle Fire HD 8.9" is also available in a [Wi-Fi only model for \\$299](#).

[Kindle Fire HD](#) is the world's most-advanced 7" tablet, with a stunning HD display, plus the same exclusive Dolby audio, dual stereo speakers, the fastest Wi-Fi, and 16 GB of storage. Kindle Fire HD is just \$199.

We are also introducing the world's most advanced e-readers, [Kindle Paperwhite](#) and [Kindle Paperwhite 3G](#).

K-12 School Essentials

[Shop now](#)

Advertisement

[Star Wars: The Clone Wars: The...](#)

The Clone Wars goes back to the original Star Wars film when Obi-Wan Kenobi tells Luke... [Read more](#)
\$44.98 **\$31.93**



Back to School
with Top Brands
In School Supplies
[Shop now](#)

Best Sellers

[Movies & TV : Criterion Collection](#)

Updated hourly



1. [The Game \(The Criterion Collection\) \[Blu-ray\]](#)
\$39.95 **\$27.96**



2. [Eclipse Series 5: The First Films of Samuel Fuller \(The...](#)
\$13.98

Unlimited Instant Videos
MP3s & Cloud Player
20 million songs, play anywhere
Amazon Cloud Drive
5 GB of free storage

Kindle
Appstore for Android
Get Quell Reflect for free today

Digital Games & Software
Audible Audiobooks

Books
Movies, Music & Games
Electronics & Computers
Home, Garden & Tools
Grocery, Health & Beauty
Toys, Kids & Baby
Clothing, Shoes & Jewelry
Sports & Outdoors
Automotive & Industrial

› Full Store Directory

Amazon Cloud Drive

Your Cloud Drive
5 GB of free storage

Get the Desktop App
For Windows and Mac

Learn More About Cloud Drive



Your photos and more
always at hand
and safely stored.

Get the desktop app ➤

amazon

Kindle Fire HD.



Kindle Fire HD
\$199

[Kindle Fire HD 8.9" 4G](#) isn't just the best tablet for the price, it's the best tablet. \$499 now gets you a large-screen HD tablet with a stunning 8.9" display, exclusive Dolby audio, dual stereo speakers, the fastest Wi-Fi, ultra-fast 4G LTE wireless, plus our new unprecedented \$49.99 one-year 4G data package. Customers save hundreds of dollars in the first year compared to other 4G tablets. Kindle Fire HD 8.9" is also available in a [Wi-Fi only model for \\$299](#).

[Kindle Fire HD](#) is the world's most-advanced 7" tablet, with a stunning HD display, plus the same exclusive Dolby audio, dual stereo speakers, the fastest Wi-Fi, and 16 GB of storage. Kindle Fire HD is just \$199.

We are also introducing the world's most advanced e-readers, [Kindle Paperwhite](#) and [Kindle Paperwhite 3G](#).



TEXTBOOKS
RENT, BUY, SELL

Shop now

Advertisement

Star Wars: Clone Wars - Volume One



The saga continues with the Emmy-winning "Star Wars: Clone Wars," available for the first... [Read more](#)



Portable Size. Massive Sound.
Jabra SOLEMATE

[Learn more](#)

Best Sellers

[Movies & TV : Criterion Collection](#)

Updated hourly



1. [The Game \(The Criterion Collection\) \[Blu-ray\]](#)
\$39.95 **\$27.96**

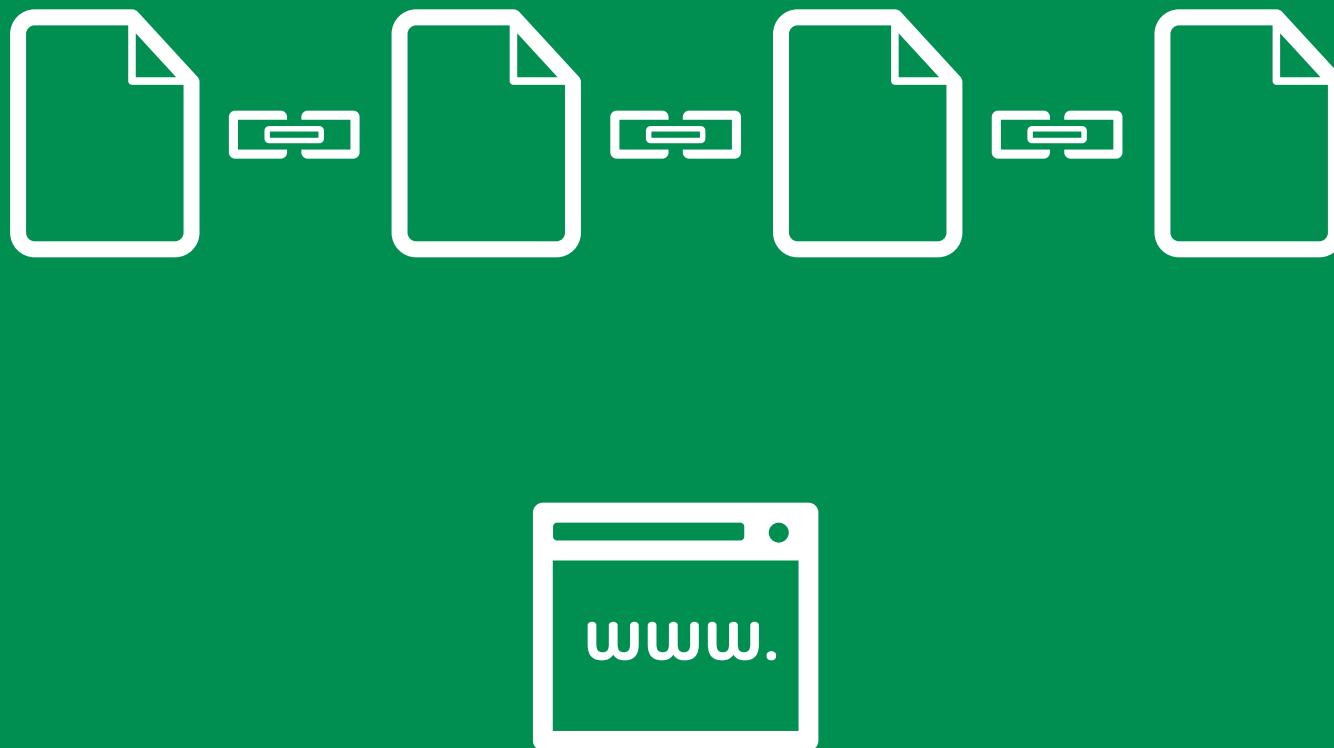


2. [Eclipse Series 5: The First Films of Samuel Fuller \(The...](#)
\$13.98

HTML

Hyper Text
+
Markup Language

Hyper Text



Markup Language

A markup language is a set of markup tags.

The purpose of the tags is to group and describe page content.

Markup Language

Without any markup to give your content structure, the browser renders unformatted and unstyled text, also known as “plain text”.

Pitchfork Music Festival Set Times Revealed
Check out the schedule for all three days

June 8, 2012 at 12:35 p.m.

Pitchfork Music Festival 2012, which returns to Chicago's Union Park July 13-15, is just five weeks away! The festival will feature Vampire Weekend, Feist, Godspeed You! Black Emperor, Beach House, Dirty Projectors, Hot Chip, Sleigh Bells, Wild Flag, Flying Lotus, [Japandroids](#), Chromatics, Real Estate, A\$AP Rocky, Danny Brown, and many, many more. And now, we're happy to announce the complete three-day schedule for the festival. Below, find the set times for all of the bands playing.

Friday, July 13:
Gates at 3 p.m.

8:30 Feist (GREEN)
8:20 Purity Ring (BLUE)
7:20 Dirty Projectors (RED)
7:15 Clams Casino (BLUE)
6:25 Big K.R.I.T. (GREEN)
6:15 [Japandroids](#) (BLUE)
5:30 A\$AP Rocky (RED)

Saturday, July 14:
Gates at 12 p.m.

8:40 Grimes (BLUE)
8:30 Godspeed You! Black Emperor (GREEN)
7:40 Danny Brown (BLUE)
7:25 Hot Chip (RED)
6:45 Chromatics (BLUE)
6:15 Sleigh Bells (GREEN)
5:45 Schoolboy Q (BLUE)

Pitchfork Music Festival Set Times Revealed Check out the schedule for all three days June 8, 2012 at 12:35 p.m. Pitchfork Music Festival 2012, which returns to Chicago's Union Park July 13-15, is just five weeks away! The festival will feature Vampire Weekend, Feist, Godspeed You! Black Emperor, Beach House, Dirty Projectors, Hot Chip, Sleigh Bells, Wild Flag, Flying Lotus, Japandroids, Chromatics, Real Estate, A\$AP Rocky, Danny Brown, and many, many more. And now, we're happy to announce the complete three-day schedule for the festival. Below, find the set times for all of the bands playing. Friday, July 13: Gates at 3 p.m. 8:30 Feist (GREEN) 8:20 Purity Ring (BLUE) 7:20 Dirty Projectors (RED) 7:15 Clams Casino (BLUE) 6:25 Big K.R.I.T. (GREEN) 6:15 Japandroids (BLUE) 5:30 A\$AP Rocky (RED) Saturday, July 14: Gates at 12 p.m. 8:40 Grimes (BLUE) 8:30 Godspeed You! Black Emperor (GREEN) 7:40 Danny Brown (BLUE) 7:25 Hot Chip (RED) 6:45 Chromatics (BLUE) 6:15 Sleigh Bells (GREEN) 5:45 Schoolboy Q (BLUE) Buy tickets here.

Markup Language

HTML tags give structure and meaning to your content.
“Semantic markup” refers to the use of meaningful tags to describe content (e.g. using header tags for header content).

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8" />
5  </head>
6  <body>
7
8
9      <h1>Pitchfork Music Festival Set Times Revealed</h1>
10     <h2>Check out the schedule for all three days</h2>
11
12     <em>June 8, 2012 at 12:35 p.m.</em>
13
14     <p>Pitchfork Music Festival 2012, which returns to Chicago's Union Park July 13-15, is just five weeks away! The festival will f
15
16     <h3>Friday, July 13:</h3>
17     <b>Gates at 3 p.m.</b>
18     <ul>
19         <li>8:30 Feist (GREEN)</li>
20         <li>8:20 Purity Ring (BLUE)</li>
21         <li>7:20 Dirty Projectors (RED)</li>
22         <li>7:15 Clams Casino (BLUE)</li>
23         <li>6:25 Big K.R.I.T. (GREEN)</li>
24         <li>6:15 Japandroids (BLUE)</li>
25         <li>5:30 ASAP Rocky (RED)</li>
26     </ul>
27
28     <h3>Saturday, July 14:</h3>
29     <b>Gates at 12 p.m.</b>
30
31     <ul>
32         <li>8:40 Grimes (BLUE)</li>
33         <li>8:30 Godspeed You! Black Emperor (GREEN)</li>
```

Markup Language

Once your content is marked up, the browser applies built-in default styles to the tags. While you can override these styles with css, your marked up, non-css styled document should be readable and have a clear hierarchy.

Pitchfork Music Festival Set Times Revealed

Check out the schedule for all three days

June 8, 2012 at 12:35 p.m.

Pitchfork Music Festival 2012, which returns to Chicago's Union Park July 13-15, is just five weeks away! The festival will feature Vampire Weekend, Feist, Godspeed You! Black Emperor, Beach House, Dirty Projectors, Hot Chip, Sleigh Bells, Wild Flag, Flying Lotus, Japandroids, Chromatics, Real Estate, A\$AP Rocky, Danny Brown, and many, many more. And now, we're happy to announce the complete three-day schedule for the festival. Below, find the set times for all of the bands playing.

Friday, July 13:

Gates at 3 p.m.

- 8:30 Feist (GREEN)
- 8:20 Purity Ring (BLUE)
- 7:20 Dirty Projectors (RED)
- 7:15 Clams Casino (BLUE)
- 6:25 Big K.R.I.T. (GREEN)
- 6:15 Japandroids (BLUE)
- 5:30 A\$AP Rocky (RED)

Saturday, July 14:

Gates at 12 p.m.

doctype

html

head

body

EXCEPTION

<!DOCTYPE html>

The doctype is not actually a tag, but a declaration, telling the browser what kind of html you are using. The doctype above declares HTML 5.

```
<html></html>
```

The <html> element defines
the whole HTML document.

```
<head></head>
```

The `<head>` element contains special elements that instruct the browser where to find stylesheets, provide meta info, and more.

```
<body></body>
```

The `<body>` element contains the document content (what is shown inside the browser window).

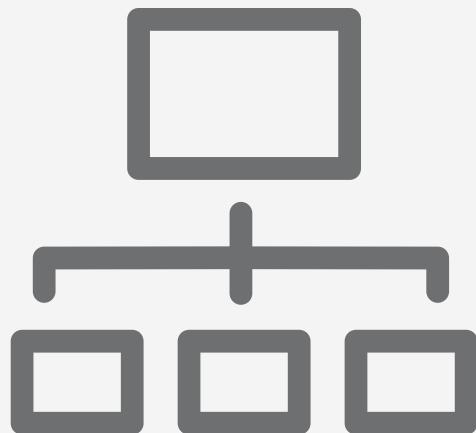
Nesting

The use of our first three tags (html, head and body), introduces and important concept: **Nesting**, which is when tags “wrap” other tags. When you create markup, you should indicate nesting by indenting the nested tags with 2 spaces (preferred) or a tab.

```
<html>  
  <head> </head>  
  <body>  
    <h1></h1>  
    <p></p>  
  </body>  
</html>
```

Document Hierarchy: Parents, children and siblings

Just as in a genealogy tree, the family hierarchy is described in terms of relationships. All elements in the document have a parent (up to 'document', which is at the top), and may have children (nested inside) or siblings (placed alongside).



<parent x>

<child and sibling y> </child and sibling y>

<child and sibling z> </child and sibling z>

</parent x>

The 'address' of an element

The document hierarchy provides us with an 'address' for each element.

```
<div class="client-text-container">
  <h1>Welcome to the Exchange!</h1>
  <h2>Applying the exchange is fast and easy</h2>|
  <div class="exchange-application-callout">
    <table>
      <tr>
        <td class="exchange-application-welcome-body">
          <p>Suspendisse vestibulum dignissim quam. Integer vel augue. Phasellus nulla purus, interdum ac, v
        </td>
        <td>
          <h4 style="text-transform: uppercase; margin-bottom: 10px;">You will need:</h4>
          <ul>
            <li>A list of your employees</li>
            <li>A list of your employees</li>
            <li>A list of your employees</li>
          </ul>
        </td>
      </tr>
    </table>
  </div>
```

in the div with class "client-text-container", make all of the h2 elements orange and 24px.

HTML Elements

Anatomy of an Element

`<tag>Content</tag>`

An HTML element includes both the HTML tag and everything between the tag (the content).

Anatomy of an Element

`<tag>Content</tag>`

Tags normally come in pairs. The first tag is the **start tag**, and the second tag is the **end tag**.

Anatomy of an Element

```
<h1>Main Headline</h1>
```

HTML has a defined set of tag names (also called keywords) that the browser understands.



The essential element tags

Primary Structure

html

head

body

Head Elements

title

meta

link

Structural Elements (block)

p

br

h1 – h6

ul

ol

a

img

(div)

Formatting Elements (inline)

em

i

strong

b

q

blockquote

(span)

Anatomy of an Element

```
<html lang="en"></html>
```

Most elements can have attributes, which provides additional information about the element.

Anatomy of an Element

```
<div class="left-nav"></div>
```

Attributes always follow the same format: name="value". You can use either single or double quotes.



The essential attributes

- link** <link rel="stylesheet" type="text/css" href="stylesheet/styles.css">

- img**

- a** My school

Introduction to Web Design

Hypertext Markup Language



Introduction to Web Design

HTML

Hypertext Markup Language

A language for describing Web pages

HTML is not a programming language, it is a *markup* language

A markup language is a set of markup tags

HTML uses markup tags to describe web pages

“Hypertext” is the ability to link one page to another

Introduction to Web Design

Early History

Hypertext Markup Language

1990: Original HTML specification written by physicist, Tim Berners-Lee for cross-referencing documents

1993: First text-based browser, Lynx, released

1993: Mosaic browser released, adding images, nested lists, forms

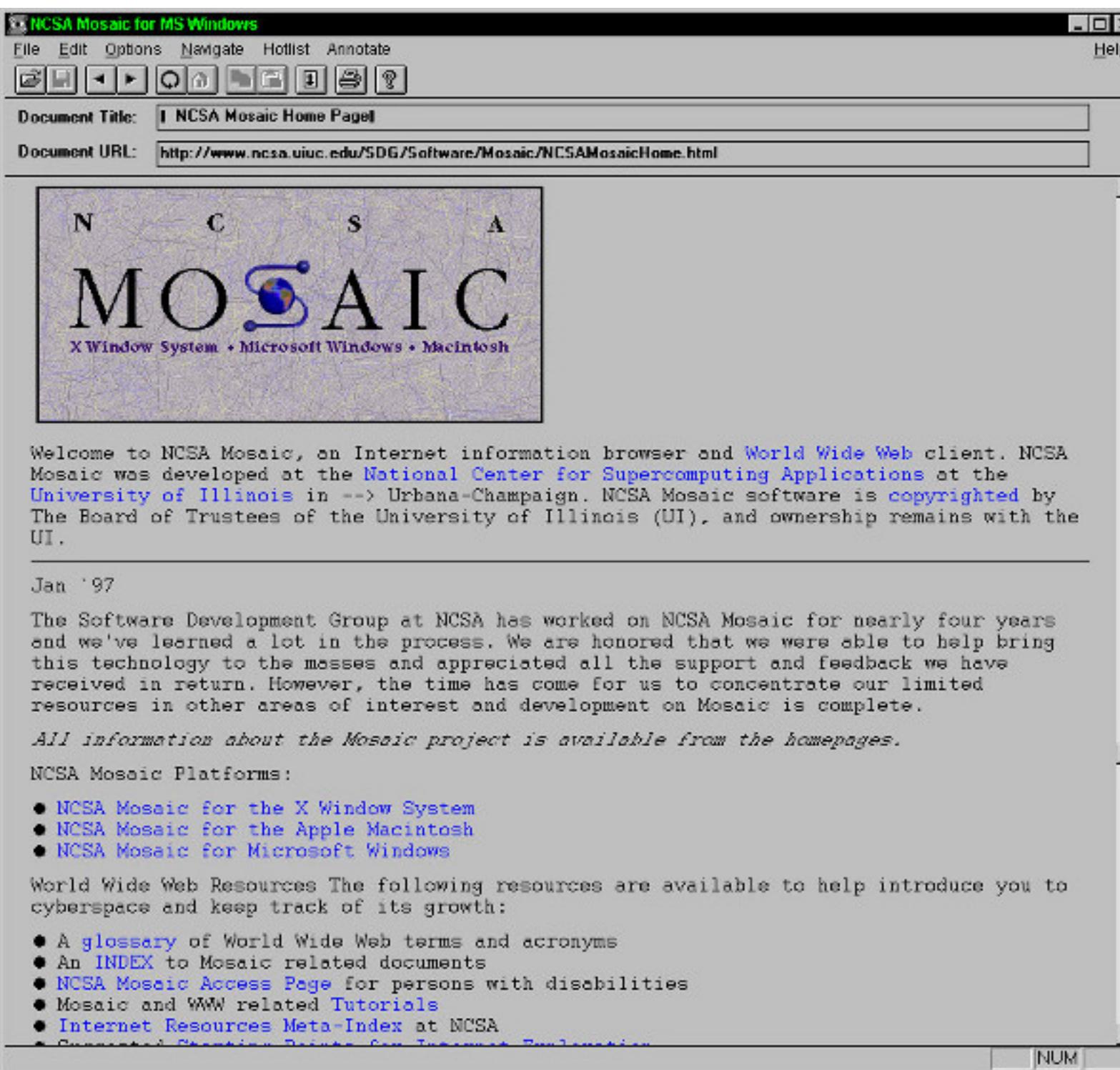
1994: First World Wide Web conference held in Geneva

1994: Netscape is formed

1994: The World Wide Web Consortium is formed, w3.org

Introduction to Web Design

Hypertext Markup Language



1993, NCSA Mosaic web browser

Introduction to Web Design

HTML Tag

Hypertext Markup Language

Keywords surrounded by angle brackets, for example:
`<html>`

HTML tags normally come in pairs, like `<h1>` and `</h1>`

The first tag in a pair is the “start tag,” the second tag is the “end tag”

Start and end tags are also called “opening” and “closing” tags

Some tags, such as ``, are self-closing

Introduction to Web Design

HTML Element

Hypertext Markup Language

An HTML element is everything from the start tag to the end tag.

For example:

<p>This is a paragraph.</p>

Start tag:

<p>

Element content:

This is a paragraph.

End tag:

</p>

Introduction to Web Design

HTML Documents

Hypertext Markup Language

HTML documents describe web pages

All they consist of is HTML tags in plain text

Networked HTML documents are web pages

Recommended plain text editors:

Sublime Text, Brackets, and Visual Studio Code

Introduction to Web Design

Web Browsers

Hypertext Markup Language

Web browsers read HTML documents and display them as web pages

Web browsers do not display HTML tags, but use them to interpret the content of the page

Current browsers include:

- Firefox
- Safari
- Chrome
- Edge
- Opera
- Tor
- Brave

Introduction to Web Design

HTML Document Essentials

Hypertext Markup Language

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Page Title</title>
  </head>
  <body>
  </body>
</html>
```

Introduction to Web Design

HTML Document Essentials

Hypertext Markup Language

`<!DOCTYPE html>` tells browsers that they are interpreting an HTML document

Text between `<html>` and `</html>` describes the web page

Text between `<title>` and `</title>` is displayed as the page title (usually at the top of the browser window)

Text between `<body>` and `</body>` is the visible page content

Introduction to Web Design

SFTP: SSH (Secure) File Transfer Protocol

Hypertext Markup Language

Web pages are usually created “locally” on a personal computer, then uploaded to a web server

A web page is not publicly accessible until it’s published to a web server

An FTP client is used to transfer files from a personal computer to a server

Cyberduck, Fetch, WinSCP, Transmit, and FileZilla are a few FTP clients to choose from

“Local” files are those on a personal computer, “remote” files are those on a web server—“live”

Introduction to Web Design

HTML5

Hypertext Markup Language

First version published in 2008, HTML5 is the latest HTML standard.

It became an official W3C recommendation as of October 2014.

- New elements
- New attributes
- Full CSS3 support
- Video and audio
- 2D and 3D graphics
- Web applications
- Smartphone apps

File Paths as Maps

- A file path is like directions to a file on a server
- Just like a map → get to a building
- Wrong path = 404 Not Found



— the ballad of a laborer, jason isolini, 2019

Absolute vs Relative Paths

Absolute Path = Full street address

Example: <https://www.nyu.edu/images/logo.png>

Relative Path = Directions from where you are

From index.html → images/photo.jpg

From pages/about.html → ../images/photo.jpg

Navigating the Hierarchy

./ = current folder

../ = go up one folder

.../.. = go up two folders

Example Tree:

/ (root)

 |—— index.html

 |—— images/

 └—— favorites/photo1.jpg

 |—— pages/

 └—— projects/project1.html

Same-Level Hierarchy

Directory Example:

```
/ (root)
|   — index.html
|   — about.html
|   — contact.html
```

From index.html → about.html

From about.html → contact.html

Rule of Thumb:

- Same folder → just filename
- Child folder → include folder name
- Parent folder → use ../

From File Path to URL

Local file: images/photo.jpg

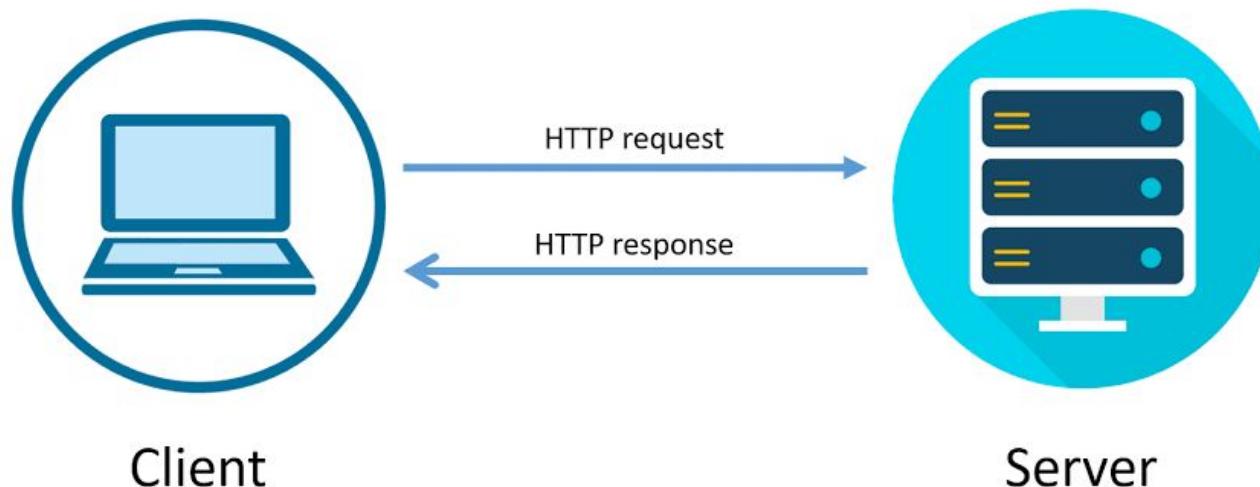
On server:

`https://i6.cims.nyu.edu/~student/images/photo.jpg`

Uploading to i6 = putting files in public directory tree

What Happens in a Request

1. Type URL → Browser parses domain + path
2. DNS Lookup → Finds server IP
3. HTTP Request → 'GET /images/photo.jpg'
4. Server Response → '200 OK' + file
5. Browser Render → Displays content



Default Document Hierarchy

When you visit a directory, server looks for defaults:

1. index.html
2. index.htm
3. default.html

Example:

<https://i6.cims.nyu.edu/~student/> → index.html
<https://i6.cims.nyu.edu/~student/pages/> → pages/index.html

HTML attributes

Elements in HTML have **attributes**; these are additional values that configure the elements or adjust their behavior in various ways to meet the criteria the users want.

REFERENCE: <https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes>

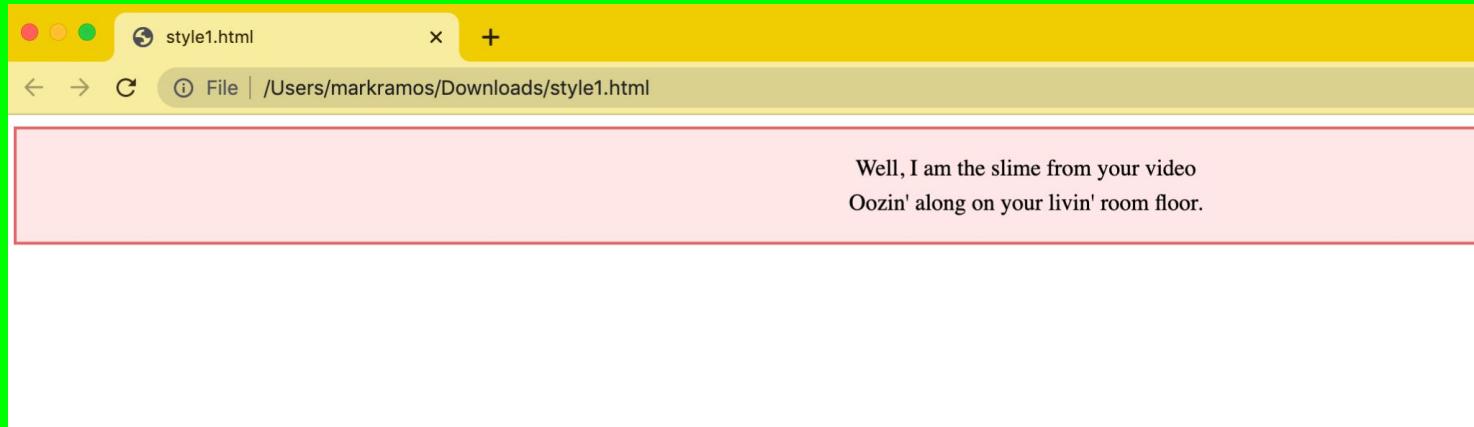
Attribute list

Attribute Name	Elements	Description
accept	<form> , <input>	List of types the server accepts, typically a file type.
accept-charset	<form>	List of supported charsets.
accesskey	Global attribute	Keyboard shortcut to activate or add focus to the element.
action	<form>	The URI of a program that processes the information submitted via the form.
align	<applet> , <caption> , <col> , <colgroup> , <hr> , <iframe> , , <table> , <tbody> , <td> , <tfoot> , <th> , <thead> , <tr>	Specifies the horizontal alignment of the element.
allow	<iframe>	Specifies a feature-policy for the iframe.

STYLE

The **style** global attribute contains CSS styling declarations to be applied to the element. *Note that it is recommended for styles to be defined in a separate file or files. This attribute and the `<style>` element have mainly the purpose of allowing for quick styling, for example for testing purposes.

```
style1.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title></title>
5 </head>
6 <body>
7     <div style="background: #ffe7e8; border: 2px solid #e66465;">
8         <p style="margin: 15px; line-height: 1.5; text-align: center;">
9             Well, I am the slime from your video<br>
10            Oozin' along on your livin' room floor.</p>
11     </div>
12
13 </body>
14 </html>
```



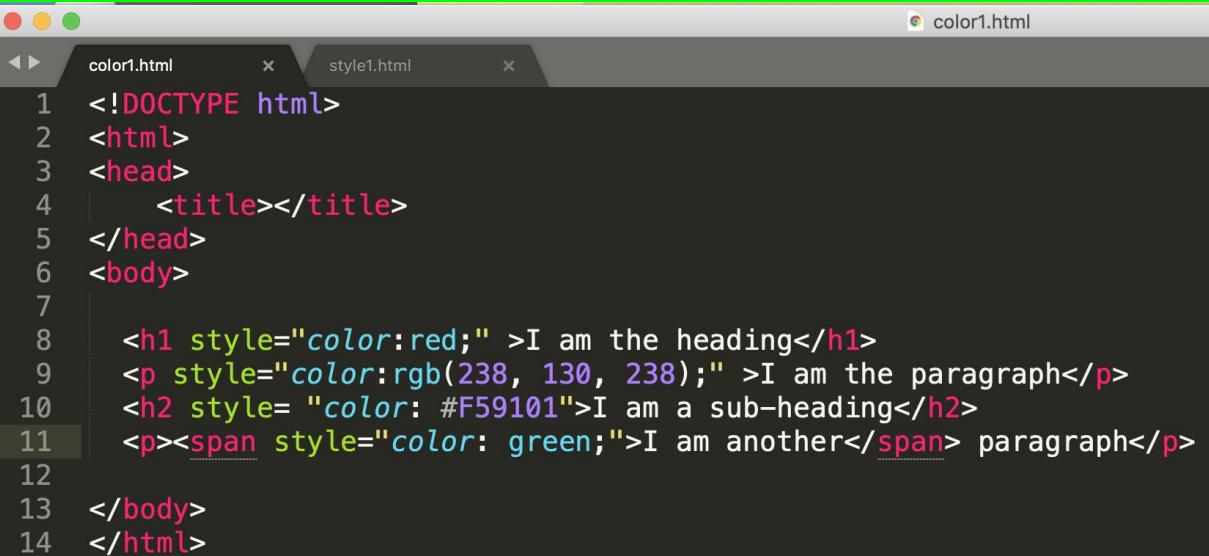
DEFINITION AND USAGE

- The style attribute specifies an **inline** style for an element.
- The style attribute will override any style set globally, e.g. styles specified in the <style> tag or in an external style sheet.
- The style attribute can be used on **any** HTML element (it will validate on any HTML element. However, it is not necessarily useful).

SYNTAX:

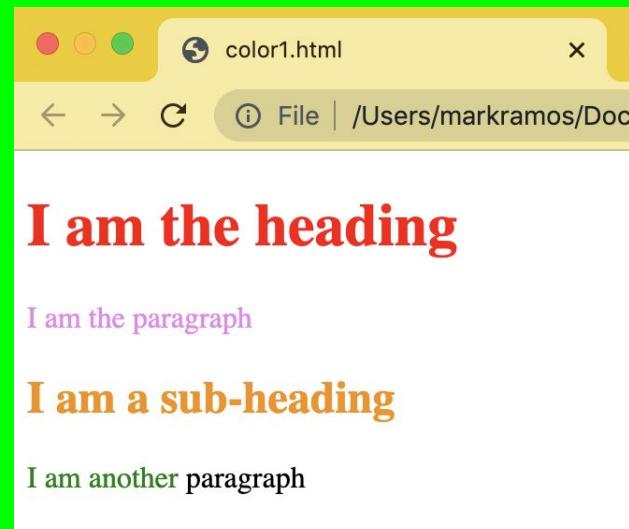
```
<element style="style_definitions">
```

COLOR - CHANGE THE TEXT COLOR

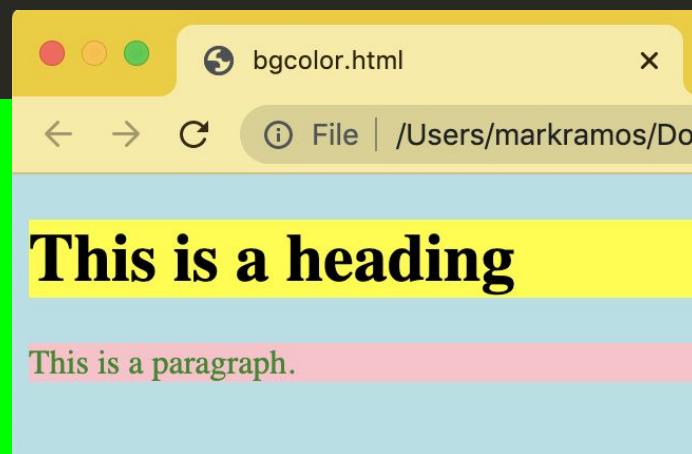


A screenshot of a code editor window titled "color1.html". The code editor shows two tabs: "color1.html" and "style1.html". The "color1.html" tab contains the following HTML code:

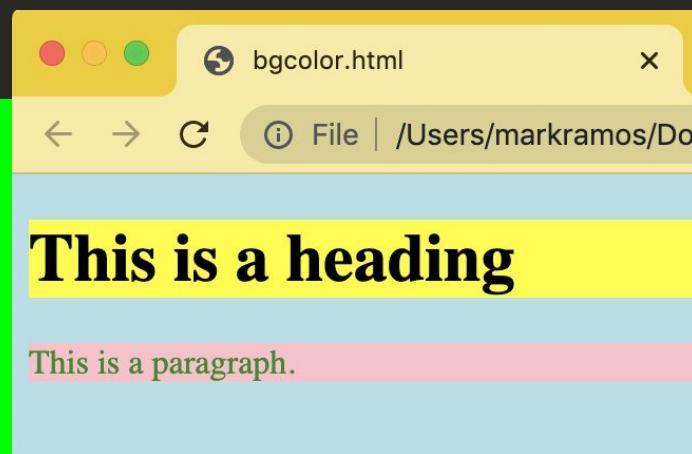
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7
8   <h1 style="color:red;">I am the heading</h1>
9   <p style="color:rgb(238, 130, 238);">I am the paragraph</p>
10  <h2 style="color: #F59101">I am a sub-heading</h2>
11  <p><span style="color: green;">I am another</span> paragraph</p>
12
13 </body>
14 </html>
```



BACKGROUND COLOR - CHANGE THE BG COLOR



```
1 <!DOCTYPE html>
2 <html>
3 <body style="background:pink;">
4   <h1 style="background:yellow;">This is a heading</h1>
5   <p style="background:pink; color:forestgreen">This is a paragraph.</p>
6 
7 </body>
8 </html>
```



alignment



A kitten is sitting on a laptop keyboard, looking at a computer screen. The screen displays the Google homepage. The kitten is positioned in front of the laptop, with its back to the viewer, looking towards the screen.

I am the heading

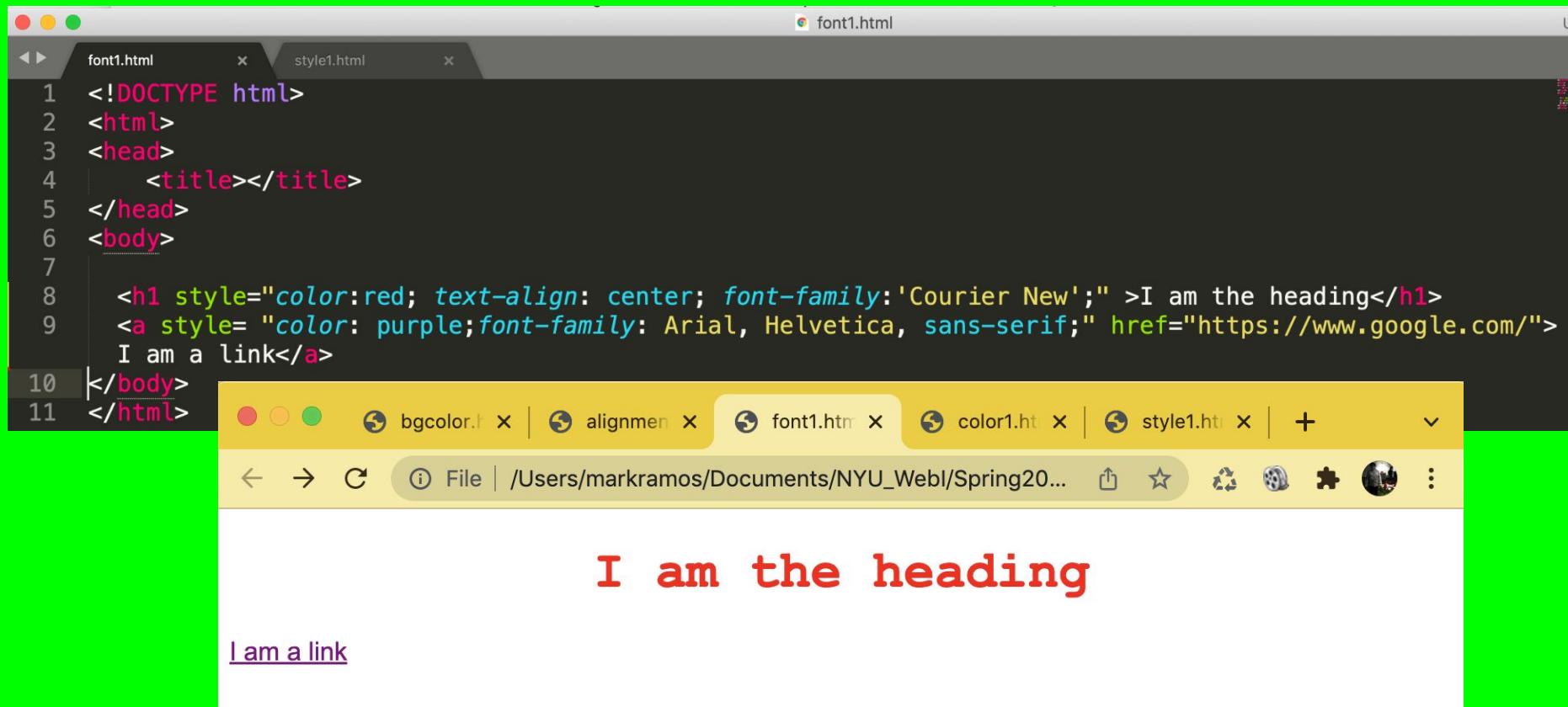
I am the paragraph

I am a sub-heading

```
alignment1.html
```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7
8   <h1 style="color:red; text-align: center;">I am the heading</h1>
9   <p style="color:rgb(238, 130, 238); text-align: left;">I am the paragraph</p>
10  <h2 style="color: #F59101; text-align: right;">I am a sub-heading</h2>
11  <p style="text-align:center;"></p>
12 </body>
13 </html>
```

FONT-FAMILY



A screenshot of a web browser window titled "font1.html". The browser has multiple tabs open, including "font1.html", "style1.html", "bgcolor.html", "alignmen.html", "font1.htm", "color1.htm", and "style1.htm". The address bar shows the file path: "/Users/markramos/Documents/NYU_WebI/Spring20...". The main content area displays the HTML code and its rendered output.

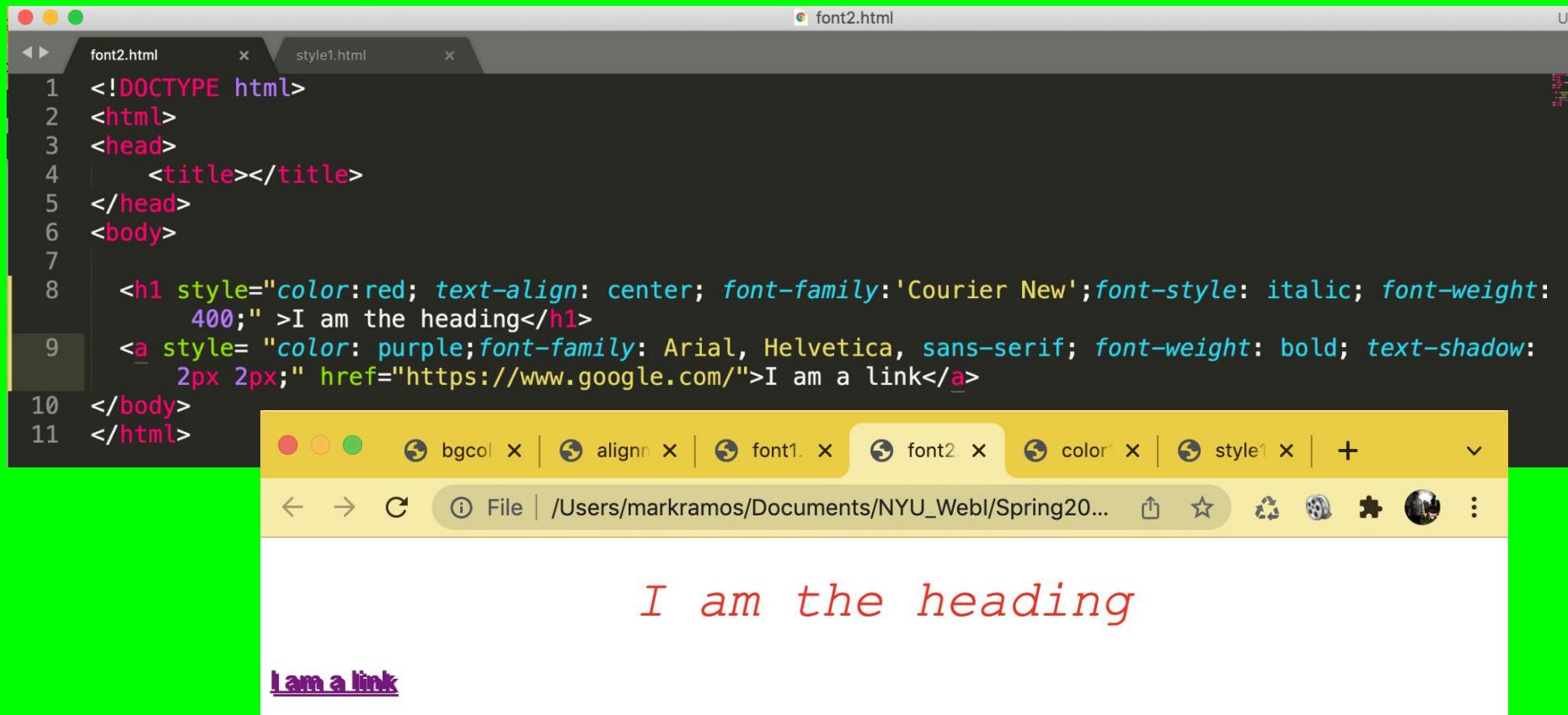
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title></title>
5 </head>
6 <body>
7
8     <h1 style="color:red; text-align: center; font-family:'Courier New';" >I am the heading</h1>
9     <a style= "color: purple;font-family: Arial, Helvetica, sans-serif;" href="https://www.google.com/">
10    I am a link</a>
11 </body>
12 </html>
```

The rendered output shows:

I am the heading

I am a link

FONT-STYLE, FONT-WEIGHT



A screenshot of a web browser window titled "font2.html". The browser interface includes tabs for "font2.html" and "style1.html", a toolbar with various icons, and a status bar at the bottom.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7
8   <h1 style="color:red; text-align: center; font-family:'Courier New';font-style: italic; font-weight: 400;">I am the heading</h1>
9   <a style= "color: purple;font-family: Arial, Helvetica, sans-serif; font-weight: bold; text-shadow: 2px 2px;" href="https://www.google.com/">I am a link</a>
10 </body>
11 </html>
```

The browser displays the following content:

I am the heading

I am a link

FONT-SIZE

The screenshot shows a web browser window with two tabs open: "font3.html" and "style1.html". The "font3.html" tab is active and displays a red heading and a purple link. The "style1.html" tab is visible in the background.

The "font3.html" content is:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7
8   <h1 style="color:red; text-align: center; font-family:'Courier New'; font-size:60px" >I am the
heading</h1>
9   <a style= "color: purple;font-family: Arial, Helvetica, sans-serif; font-weight: bold; text-shadow:
20px 20px; font-size:120px" href="https://www.google.com/">I am a link</a>
10 </body>
11 </html>
```

The "style1.html" content is:

```
bgcolor | alignme | font1.h | font2.h | font | color1.h | style1.h | +
```

The browser's address bar shows the file path: "/Users/markramos/Documents/NYU_WebI/Spring20...".

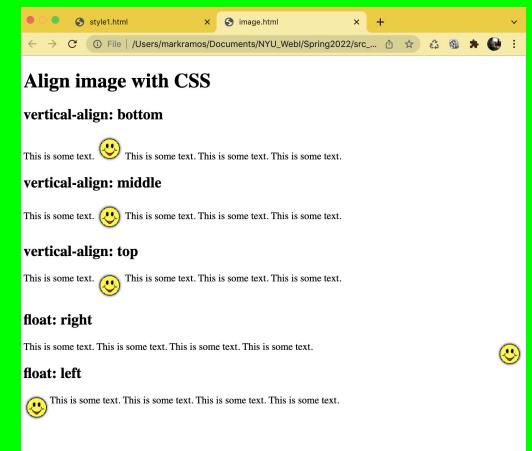
The rendered content in the browser is:

I am the heading

I am a link

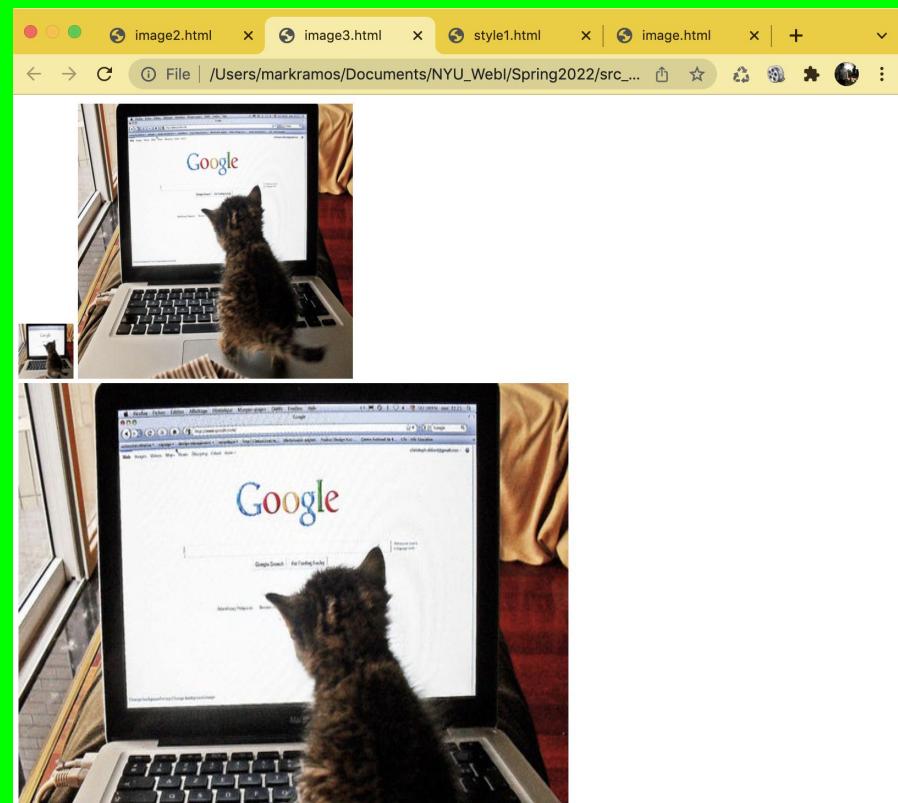
 ATTRIBUTES

```
image.html UNREGISTERED
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Align image with CSS</h1>
6
7 <h2>vertical-align: bottom</h2>
8 <p>This is some text.  This is some text. This is some text. This is some text.</p>
9
10 <h2>vertical-align: middle</h2>
11 <p>This is some text.  This is some text. This is some text. This is some text.</p>
12
13 <h2>vertical-align: top</h2>
14 <p>This is some text.  This is some text. This is some text. This is some text.</p>
15
16 <h2>float: right</h2>
17 <p>This is some text.  This is some text. This is some text. This is some text.</p>
18
19 <h2>float: left</h2>
20 <p>This is some text.  This is some text. This is some text. This is some text.</p>
21
22 </body>
23 </html>
```



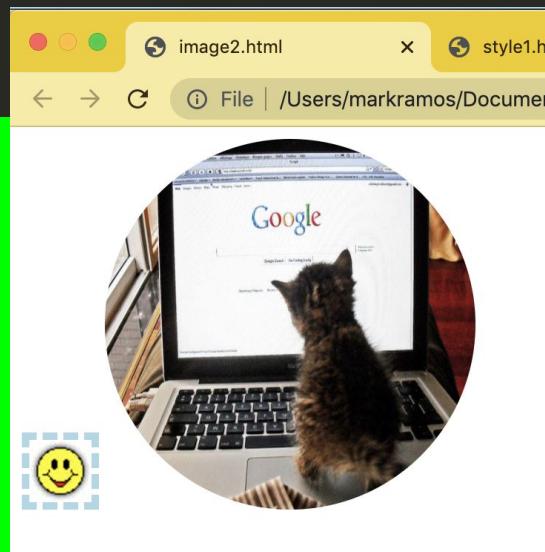
 width x height

```
image3.html
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 
6 
7 
8
9 </body>
10 </html>
```



BORDERS AND RADIUS

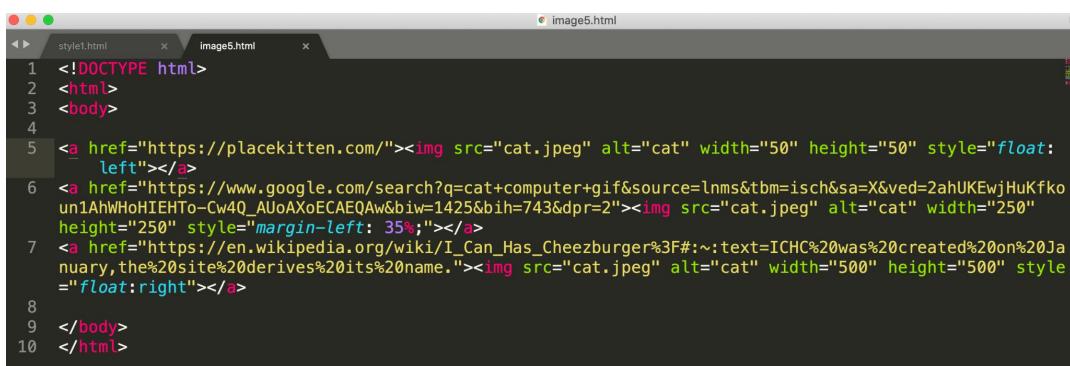
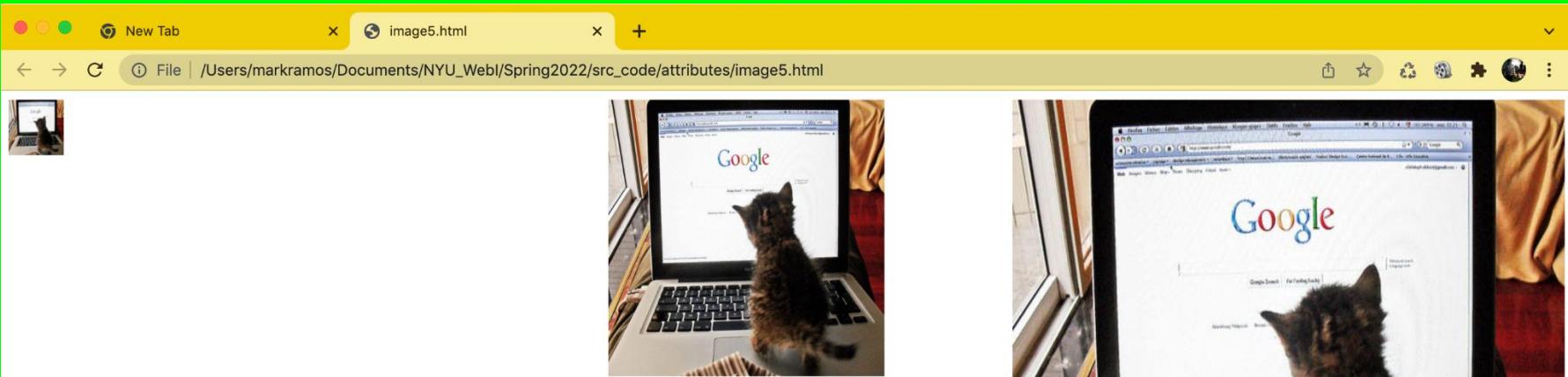
```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 
6 
7
8 </body>
9 </html>
```



 LINK

```
style1.html      x   image4.html      x
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <a href="https://placekitten.com/"></a>
6  <a href="https://www.google.com/search?q=cat+computer+gif&source=lnms&tbo=isch&sa=X&ved=2ahUKEwjHuKfko
un1AhWHoHIEHTo-Cw4Q_AUoAXoECAEQAw&biw=1425&bih=743&dpr=2"></a>
7  <a href="https://en.wikipedia.org/wiki/I_Can_Has_Cheezburger%3F#:~:text=IHC%20was%20created%20on%20Ja
nuary,the%20site%20derives%20its%20name."></a>
8
9  </body>
10 </html>
```

POSITIONING



The image shows a screenshot of a web browser window titled "image5.html". The browser's address bar indicates the file is located at "/Users/markramos/Documents/NYU_WebI/Spring2022/src_code/attributes/image5.html". The main content of the browser shows a cat sitting on a laptop keyboard. In the background, a Google search results page is visible. Below the browser, a code editor window titled "style1.html" displays the following HTML code:

```
<!DOCTYPE html>
<html>
<body>
<a href="https://placekitten.com/"></a>
<a href="https://www.google.com/search?q=cat+computer+gif&source=lnms&tbo=isch&sa=X&ved=2ahUKEwjHuKfko un1AhwHoIEHTo-Cw40_AUoAXoECAEQAw&btiw=1425&bih=743&dpr=2"></a>
<a href="https://en.wikipedia.org/wiki/I_Can_Has_Cheezburger%3F#:%text=IHC%20was%20created%20on%20January,%20the%20site%20derives%20its%20name."></a>
</body>
</html>
```

<STYLE>: THE STYLE INFORMATION ELEMENT

The `<style>` HTML element contains style information for a document, or part of a document. It contains CSS, which is applied to the contents of the document containing the `<style>` element.

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/style>

 HTML Demo: <style>

Reset

HTML **CSS**

```
1 <style>
2 p {
3   color: green;
4 }
5 </style>
6
7 <p>This text will be green. Inline styles take precedence
over CSS included externally.</p>
8
9 <p style="color: blue">The <code>style</code> attribute
can override it, though.</p>
10
```

Output

This text will be green. Inline styles take precedence over CSS included externally.

The `style` attribute can override it, though.

Introduction to Web Design

Cascading Style Sheets



Introduction to Web Design

CSS

Cascading Style Sheets

CSS defines a web page's appearance

CSS separates style and content

Consists of a plain text file with rules for the display of HTML elements

Formatting includes fonts and colors as well as layout and position

Can be created outside of your HTML and applied to multiple Web pages

Well-formed HTML is important for your CSS to work properly

Introduction to Web Design

CSS History

Cascading Style Sheets

Prior to CSS, Web pages were commonly styled with HTML tags and structured with tables

This was both tedious and inefficient

Nine different style sheet languages were proposed, two were chosen as the foundation

CSS Level 1 emerged as a W3C Recommendation in December 1996

Browsers began to support CSS over the next few years

Introduction to Web Design

CSS Application

Cascading Style Sheets

CSS can be applied in three different ways to a web page:

- In an external .css file
- In the <head> section of an HTML document
- Inline with HTML code

Introduction to Web Design

CSS Rule Set

Cascading Style Sheets

Selector: Indicates which HTML element will be formatted

Declaration block: Describes the formatting to apply

Property/value pair: Specifies format

Style rules are separated by a semicolon

```
h1 {  
    color: green;  
    background: yellow;  
}
```

Introduction to Web Design

Cascade

Cascading Style Sheets

The principle of the “cascade” is applied when style rules are in conflict

Three primary factors determine which style rule wins out:

- Inheritance
- Specificity
- Location

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

Introduction to Web Design

Display Mode

Cascading Style Sheets

Elements in HTML are primarily “inline” or “block” elements.

- An inline element allows content to flow around its left and right sides.
- A block element fills the entire line and nothing is displayed on its left or right side.

The CSS display property allows you to specify the type of box used for an HTML element.

Introduction to Web Design

CSS Box Model

Cascading Style Sheets

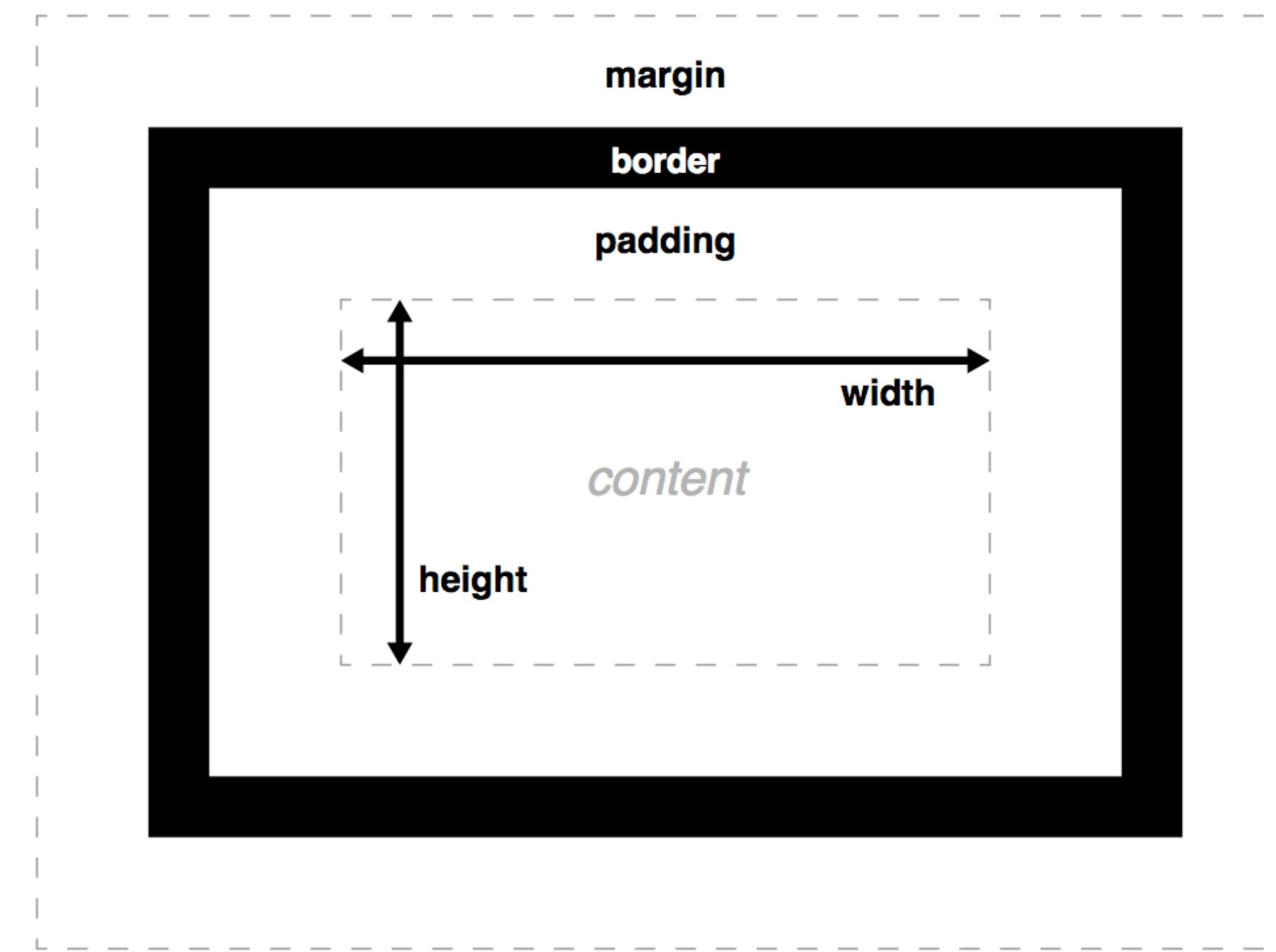
In a web page, every element is rendered as a rectangular box.

This box includes the following, changeable properties.

- Content
- Padding
- Border
- Margin

Introduction to Web Design

Cascading Style Sheets



CSS Box Model

Introduction to Web Design

Units of Length

Cascading Style Sheets

There are two types of length units in CSS, relative and absolute.

Relative units of length include:

- em (relative to font size)
- % (relative to the containing element)

Absolute units of length include:

- px (pixels)

Alternatively specifications:

- auto (browser calculates length)
- inherit (from the parent element)

Introduction to Web Design

CSS3

Cascading Style Sheets

CSS3 is the latest standard for CSS

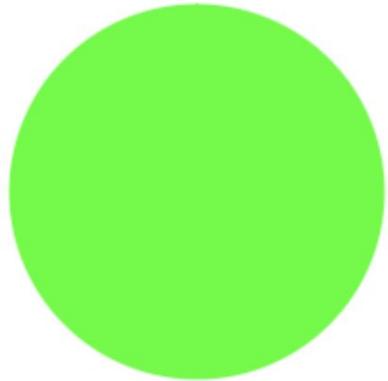
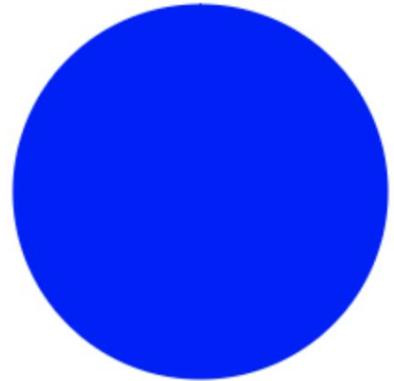
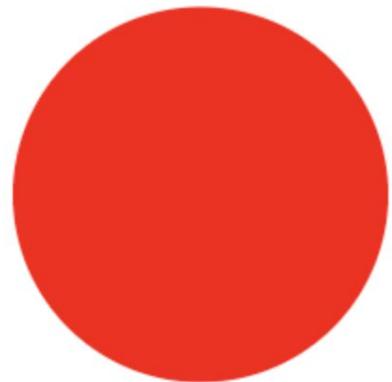
CSS2 is best supported

CSS3 is still evolving but offers new features for designers and developers

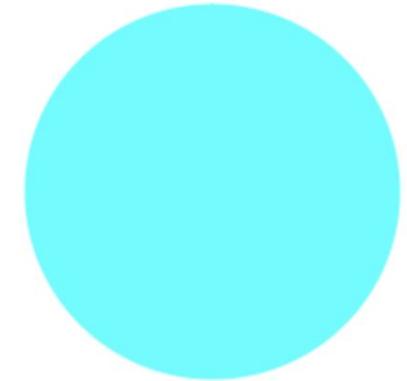
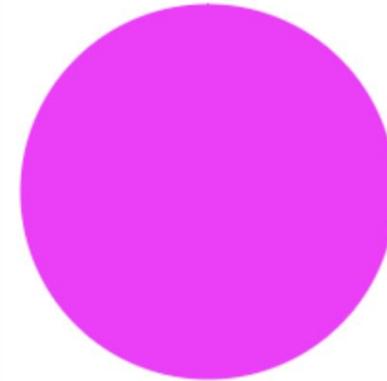
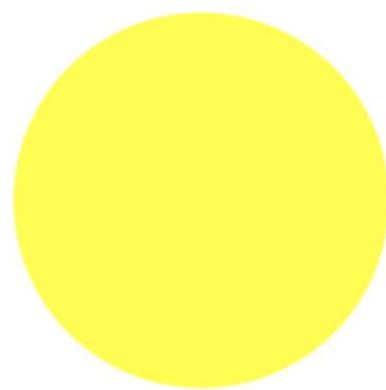
Modern browsers support many aspects of CSS3

CSS3 is backwards compatible with CSS2

CSS - color, font, background styling



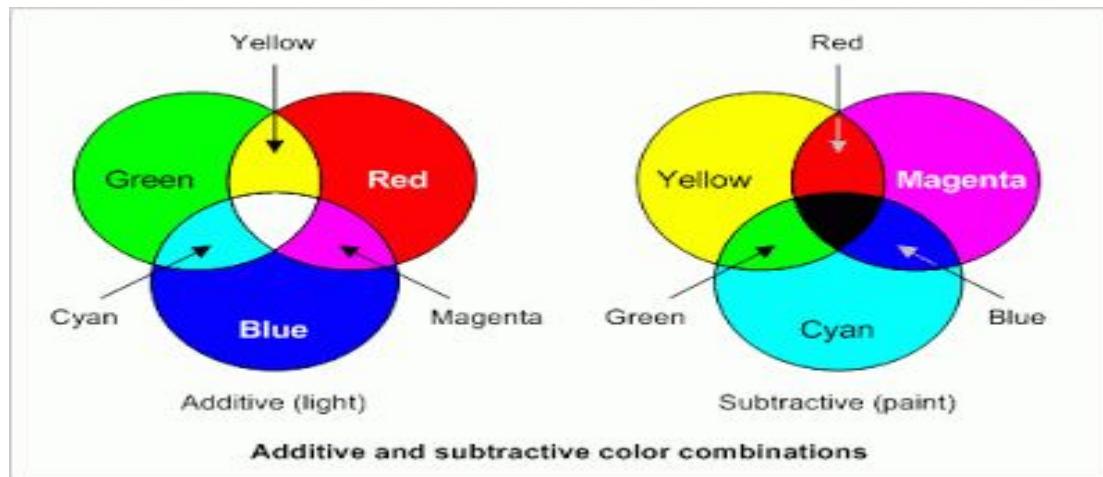
ADDITIVE



SUBTRACTIVE

Additive color

Colors displayed from a screen are called additive colors. Additive color or additive mixing is a property of a color model that predicts the appearance of colors made by coincident component lights



<named color>

Syntax

```
color: red;  
color: orange;  
color: tan;  
color: rebeccapurple;  
color: transparent;
```



CSS Level 1 Values

CSS Level 1 only included 16 basic colors, called the *VGA colors* as they were taken from the set of displayable colors on VGA graphics cards.

<https://developer.mozilla.org/en-US/docs/Web/CSS/named-color>

CSS Level 2 Values

Keyword	RGB hex value	Sample
orange	#ffa500	

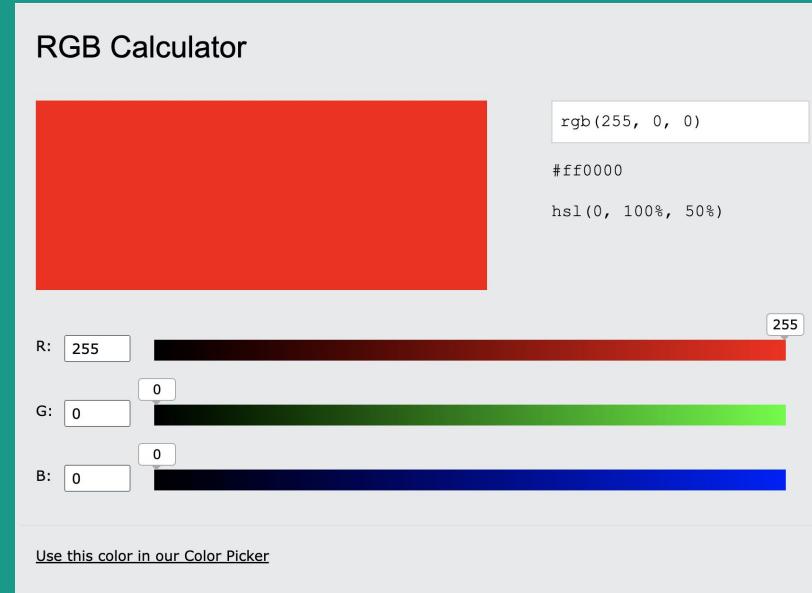


CSS Level 3 Values

Although various colors not in the specification (mostly adapted from the X11 colors list) were supported by early browsers, it wasn't until SVG 1.0 and [CSS Colors Level 3](#) that they were formally defined. They are called the *extended color keywords*, the *X11 colors*, or the *SVG colors*.

<https://developer.mozilla.org/en-US/docs/Web/CSS/named-color>

rgb color codes





RGB color (0-255)

https://www.w3schools.com/colors/colors_rgb.asp

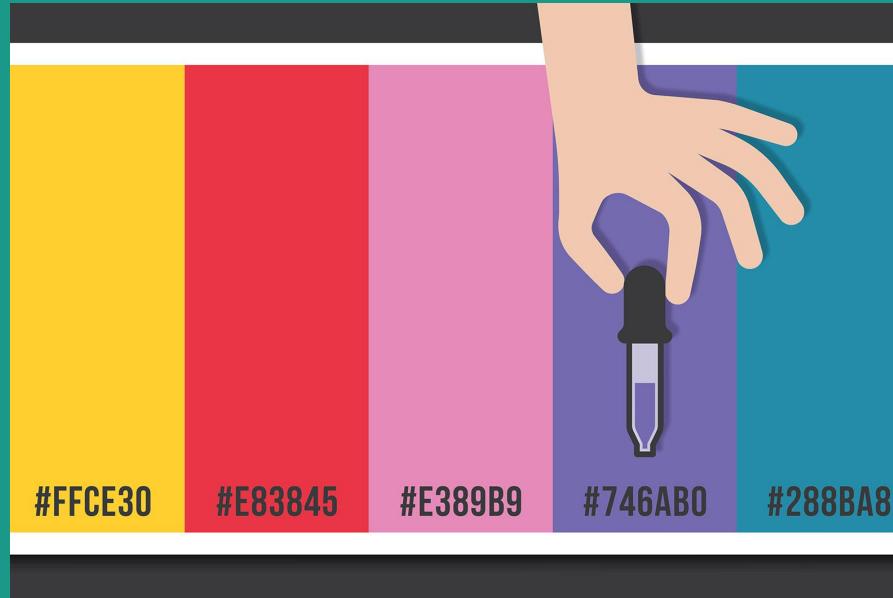
RGB color values are supported in all browsers.

An RGB color value is specified with: `rgb(red, green, blue)`.

Each parameter (red, green, and blue) defines the intensity of the color as an integer between 0 and 255.

For example, `rgb(0, 0, 255)` is rendered as blue, because the blue parameter is set to its highest value (255) and the others are set to 0.

Hexadecimal color values





Hex triplet

A hex triplet is a six-digit, three-byte hexadecimal number used in HTML, CSS, SVG, and other computing applications to represent colors. The bytes represent the red, green, and blue components of the color. One byte represents a number in the range 00 to FF (in hexadecimal notation), or 0 to 255 in decimal notation. This represents the least (0) to the most (255) intensity of each of the color components. Thus web colors specify colors in the 24-bit RGB color scheme. The hex triplet is formed by concatenating three bytes in hexadecimal notation, in the following order:

Byte 1: red value (color type red)

Byte 2: green value (color type green)

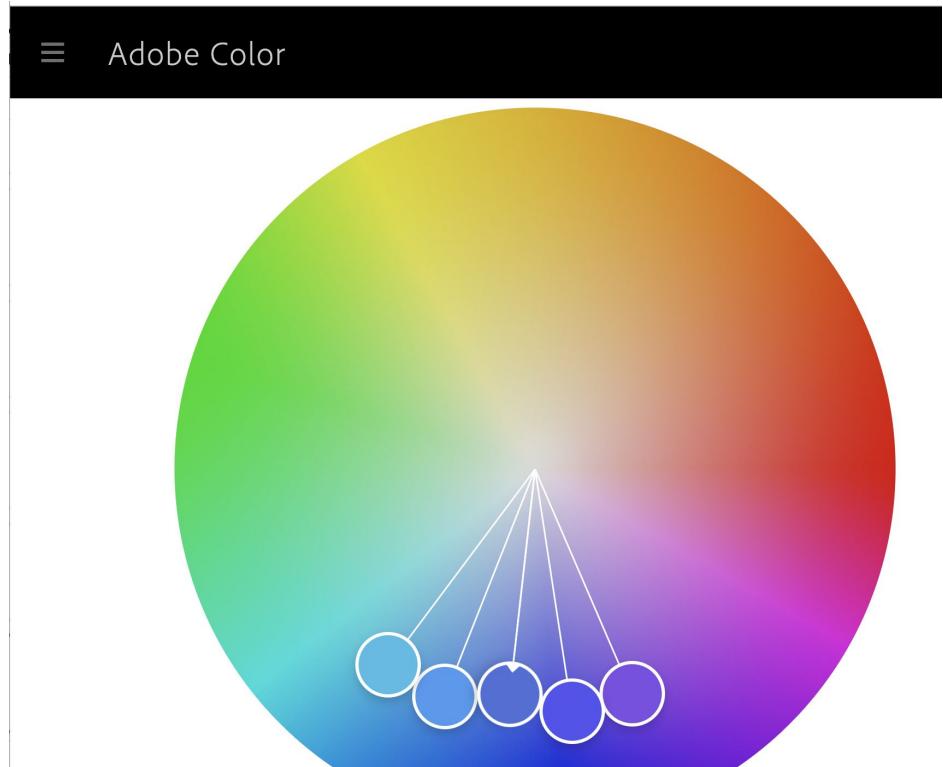
Byte 3: blue value (color type blue)

Web color wheels

There are lots of tools that help developers identify and use colors in their code.

I like this one:

<https://color.adobe.com/create/color-wheel>



fonts

Georgia Italic

Baskerville

Helvetica Bold

Bickham Script



Giddygoo

Didot Italic

@font-face / @font-family

The @font-face CSS at-rule specifies a custom font with which to display text; the font can be loaded from either a remote server or a locally-installed font on the user's own computer.

The font-family CSS property specifies a prioritized list of one or more font family names and/or generic family names for the selected element.

```
@font-face {  
    font-family: "Open Sans";  
    src: url("/fonts/OpenSans-Regular-webfont.woff2")  
        format("woff2"), url("/fonts/OpenSans-Regular-webfont.woff")  
        format("woff");  
}
```

```
p.a {  
    font-family: "Times New Roman", Times, serif;  
}
```

```
p.b {  
    font-family: Arial, Helvetica, sans-serif;  
}
```



Web-safe browser fonts

- Arial (sans-serif)
- Verdana (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Custom fonts

<http://www.fontfoundry.com/>

<https://fonts.google.com/>

background styling

CSS Demo: background

RESET

```
background: green;
```

```
background: content-box radial-
```

```
background: no-repeat url("../img/fireside.png")
```

```
background: left 5% / 15% 60% re
```

```
background: center / contain no
```

The image shows a screenshot of a CSS demo application titled "CSS Demo: background". On the left, there is a vertical list of CSS code snippets. From top to bottom, they are: "background: green;", "background: content-box radial-", "background: no-repeat url("../img/fireside.png")", "background: left 5% / 15% 60% re", and "background: center / contain no". The last snippet is currently highlighted with a blue border. To the right of the code list is a large, colorful Firefox logo. The logo is rendered with a radial gradient background, where the colors transition from red at the edges to blue in the center. It is surrounded by green leaves and small green frog-like creatures, illustrating how different background properties can change the appearance of a single element.

#eee 35% url("../img/fireside...



background

The background shorthand CSS property sets all background style properties at once, such as color, image, origin and size, or repeat method.

<https://developer.mozilla.org/en-US/docs/Web/CSS/background>

CSS

Cascading Style Sheets

CSS Setup

Cascading Style Sheets

The basic structure of every web page, HTML, is very plain on its own. The beautiful websites that you see across the internet are styled with a variety of tools, including CSS.

CSS, or Cascading Style Sheets, is a language that web developers use to style the HTML content on a web page. If you're interested in modifying colors, font types, font sizes, shadows, images, element positioning, and more, CSS is the tool for the job!

Example 1

Download the source code for example 1 on the online syllabus.

Open the .html file with Sublime, then preview it in Chrome or Firefox.

We should see all the html elements unstyled as plain html.

Add the following code to line 5 of the .html file:

```
<link href="style.css" type="text/css" rel="stylesheet">
```

Save your file, and now refresh the example.html page

We should see the entire styling of the website changed. Let's take a minute to look at the accompanying style.css file. The code we wrote linked the .css file to the .html page and applied the styling.

Inline styling

Although CSS is a different language than HTML, it's possible to write CSS code directly within HTML code using *inline styles*.

To style an HTML element, you can add the `style` attribute directly to the opening tag. After you add the attribute, you can set it equal to the CSS style(s) you'd like applied to that element.

Remember, we've done this before by using the `style` attribute to change the position of an `` with `float`.

Let's try it again with `example2.html`. Download this file and open it in your text editor and browser.

Let's change the font of the first paragraph by applying a `style` attribute into the right `<p>` tag.

```
style="font-family: Arial;"
```

The <style> Tag

HTML also allows you to write CSS code in its own dedicated section with the `<style>` element. CSS can be written between opening and closing `<style>` tags. To use the `<style>` element, it must be placed inside of the `<head>` element.

Let's delete the style attribute we added to the `<p>` tag in `example2.html` and move it into a style tag.

Like this:

```
4  <head>
5      <title>Vacation World</title>
6      <style type="text/css">
7          p{
8              font-family: "Arial";
9          }
10     </style>
11     </head>
12
```

Your turn!

Change the `font-color` of the paragraph to red.

Change the `font-style` of the paragraph to italic.

Change the `font-size` of the paragraph to 30px.

Target the ``. Change it's `float` position to right.

And finally!

Convert this html file with inline CSS to an html file that links to an external stylesheet like demonstrated in example 1.

You'll need to copy the code you wrote between the style tags and paste it into a new file. Save that file as a .css file and make sure to delete the `<style> </style>` tags from both files.

Make sure the .css file is saved in a place that is accessible to your .html file.

In your .html file, link to the .css file you created within the `<head>` tags:

```
<link href="your_file_name.css" type="text/css" rel="stylesheet">
```

CSS Selectors

Tag Names

The basic CSS selector targets html tag names.

We've seen how we are able to target html elements like the `<p>` and `` tags in the code you wrote previously.

For example, let's target the `<div>` elements in our source code and change the color to maroon.

Any element with the tag of `<div>` will be affected by the CSS styling we specify.

Class Names

CSS is not limited to selecting elements by tag name. HTML elements can have more than just a tag name; they can also have *attributes*. One common attribute is the `class` attribute. It's also possible to select an element by its `class` attribute.

For example, consider the following HTML:

```
<p class="brand">Sole Shoe Company</p>
```

The paragraph element in the example above has a `class` attribute within the `<p>` tag. The `class` attribute is set to "brand". To select this element using CSS, we could use the following CSS selector:

```
.brand {  
}  
}
```

To select an HTML element by its class using CSS, a period (.) must be prepended to the class's name. In the example above case, the class is `brand`, so the CSS selector for it is `.brand`.

In your stylesheet for example2, target the class “title” and change the color to teal.

Multiple classes

It's possible to add more than one class name to an HTML element's `class` attribute.

For instance, perhaps there's a heading element that needs to be green and bold. You could write two CSS rules like so:

```
.green {  
    color: green;  
}  
  
.bold {  
    font-weight: bold;  
}
```

And structure the html accordingly by including both classes one the html element:

```
<h1 class="green bold"> ... </h1>
```

We can add multiple classes to an HTML element's `class` attribute by separating them with a space. This enables us to mix and match CSS classes to create many unique styles without writing a custom class for every style combination needed.

Let's practice with our example2 files. In the .html file, find the h1 tag with a class of "title". Let's add a second class to this called uppercase so the html looks like this:

```
11 <h1 class="title uppercase">
```

And in our .css file, let's add the following styling:

```
10 .uppercase{  
11     text-transform: uppercase;  
12 }
```

CSS classes are meant to be reused over many elements.

The convention is to use classes to target multiple html elements at once, i.e. if multiple elements all need the same styling

ID Name

If an HTML element needs to be styled uniquely (no matter what classes are applied to the element), we can add an ID to the element. To add an ID to an element, the element needs an `id` attribute:

```
<h1 id="large-title"> . . . </h1>
```

Then, CSS can select HTML elements by their `id` attribute. To select an `id` element, CSS prepends the `id` name with a hashtag (`#`). For instance, if we wanted to select the HTML element in the example above, it would look like this:

```
#large-title {  
}  
}
```

Let's try using an #id as a selector

Let's add an id called "article-title" to our h1 element in example2.html:

```
11 ▼ <h1 class="title uppercase" id="article-title">
```

And now we'll style it in our .css file:

```
17 ▼ #article-title {  
18     font-family: cursive;  
19     text-transform: capitalize;  
20 }
```



While classes are meant to be used many times, an ID is meant to style only one element. IDs override the styles of tags and classes. Since IDs override class and tag styles, they should be used sparingly and only on elements that need to always appear the same.



Pseudo-classes

CSS pseudo classes

A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s). For example, :hover can be used to change a link's color when the user's pointer hovers over it.

```
/* Any link over which the user's pointer is hovering */  
  
a:hover {  
    color: blue;  
}
```

Class files

`class7/class_selectors/selectors.html`

`class7/class_selectors/styles/selectors.css`

Your turn!

So far, all the CSS selectors we've seen map directly to a piece of HTML markup that we wrote. However, there's more going on in a rendered web page than just our HTML content. There's "stateful" information about what the user is doing (opposed to the content we've authored).

The classic example is a link. As a web developer, you create an `<a href>` element. After the browser renders it, the user can interact with that link. They can hover over it, click it, and visit the URL.

Save a new copy of the class file, and assign unique ID's to the 4 links that don't have them.

Style each link's link, active, and hover states so they are all different.

CSS Buttons

Pseudo-classes aren't just for styling text links—they can be applied to any kind of selector (not just type selectors).

Let's look at pseudo-buttons (not actually html button elements) styled with CSS:

`class7/class_selectors/selectors2.html`

`class7/class_selectors/styles/selectors2.css`

Introduction to Web Design

Raster
Graphics

Introduction to Web Design

Binary Files

Raster Graphics

All files can be categorized into one of two file formats:

- Text
- Binary

When you write code, that is a text file.

An image file, on the other hand, is a binary file.

Binary files typically contain a sequence of bytes, or ordered groupings of eight bits.

Introduction to Web Design

Raster Graphics

Raster Graphics

Raster Graphics, also referred to as “bitmap” graphics, are binary files.

Raster graphics consist of a grid of picture elements, pixels, each of which contain color and brightness information.

Pixels can be changed individually or as a group with program algorithms.

This is contrast to vector graphics, which describe points and lines.

Introduction to Web Design

Web Formats

Raster Graphics

JPEG

“Joint Photographic Experts Group”

PNG

“Portable Network Graphics”

GIF

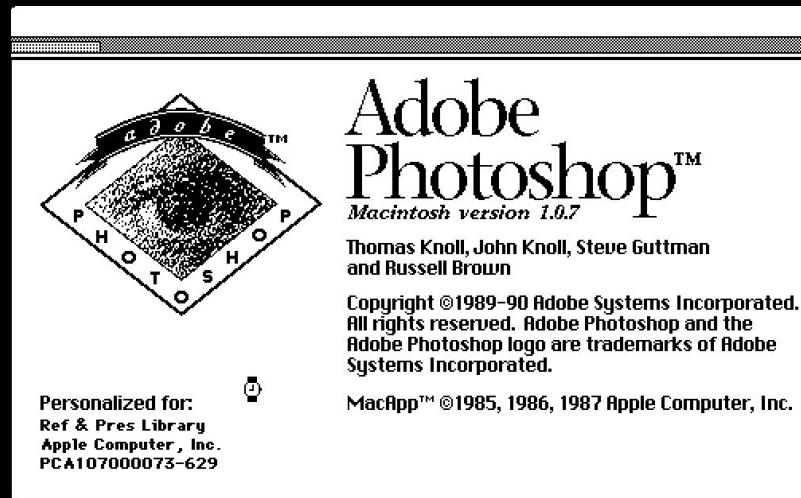
“Graphic Interchange Format”

WebP

Up-and-coming web image format

Introduction to Web Design

Raster Graphics



Introduction to Web Design

Photoshop

Raster Graphics

Photoshop was created in 1987 by Thomas Knoll, then a PhD student at the University of Michigan.

It was originally called “Display.”

Its purpose was to display and manipulate grayscale images scanned into a computer.

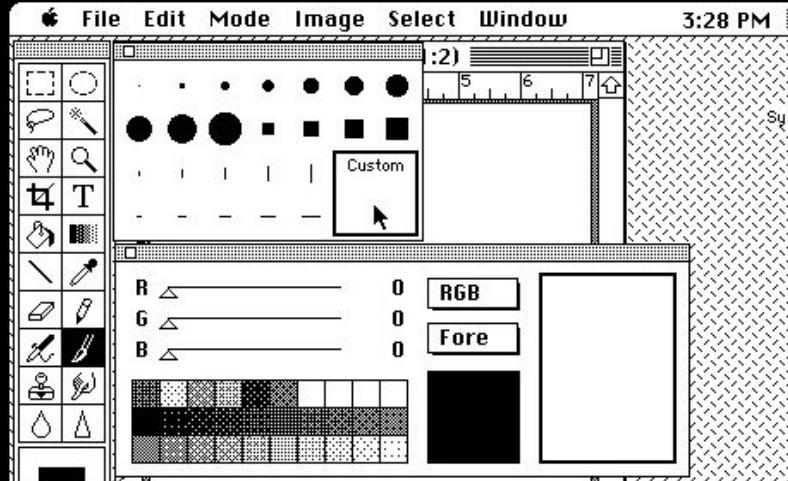
Display was acquired by Adobe in 1988.

It was released as Photoshop 1.0 for Macintosh in 1990.

Layer support was introduced in version 3 (c. 1993).

Introduction to Web Design

Raster Graphics



Jennifer in paradise: the story of the first Photoshopped image

One holiday snap has been manipulated thousands of times on thousands of computers. Here's how a woman on a beach in Bora Bora taught the whole world to tinker with pictures



© Jennifer in Paradise.tif – the picture Photoshop co-creator John Knoll took of his future wife Jennifer in Bora Bora. Photograph: John Knoll

<https://www.theguardian.com/artanddesign/photography-blog/2014/jun/13/photoshop-first-image-jennifer-in-paradise-photography-artefact-knoll-dullaart>



<https://jennifer.ps/>

Jennifer in Paradise – the correspondence

1 March 2016



Jennifer.ps, Constant Dullaart, 2014

<http://carrollfletcheronscreen.com/2016/03/01/jennifer-in-paradise-the-correspondence/>

Introduction to Web Design

Raster
Graphics



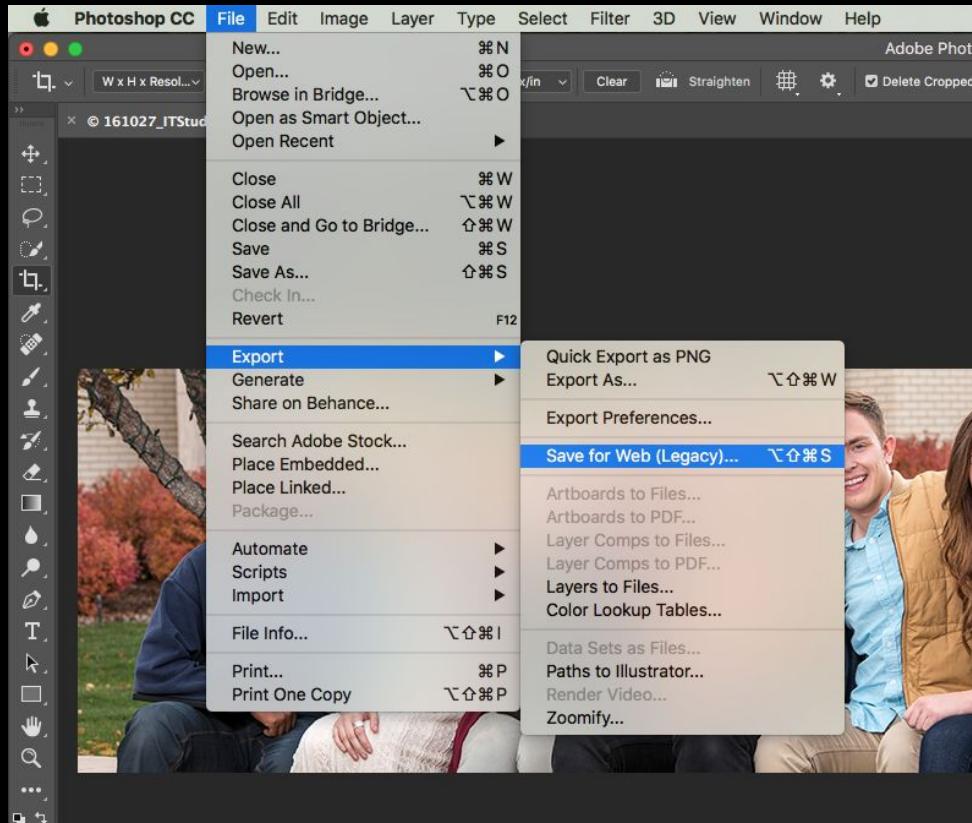
The Web and File Sizes

Large image file sizes over 200k significantly increase page load times. Even if a page with large images seems to load quickly on campus, these same pages may take much longer to load elsewhere.

Keep your image file sizes small:

- Save them at less than 100% quality
- Check progressive when saving them for web
- Double check resolution

(Progressive means that when you first get to your web page, it will show a low quality version of the full picture that will progressively gets sharper as the page loads. Otherwise, the picture will load in full quality, but only a bar of information at a time.)







Tip: Use File > Export > Export As... or right click on a layer for a faster way to export assets

Learn More

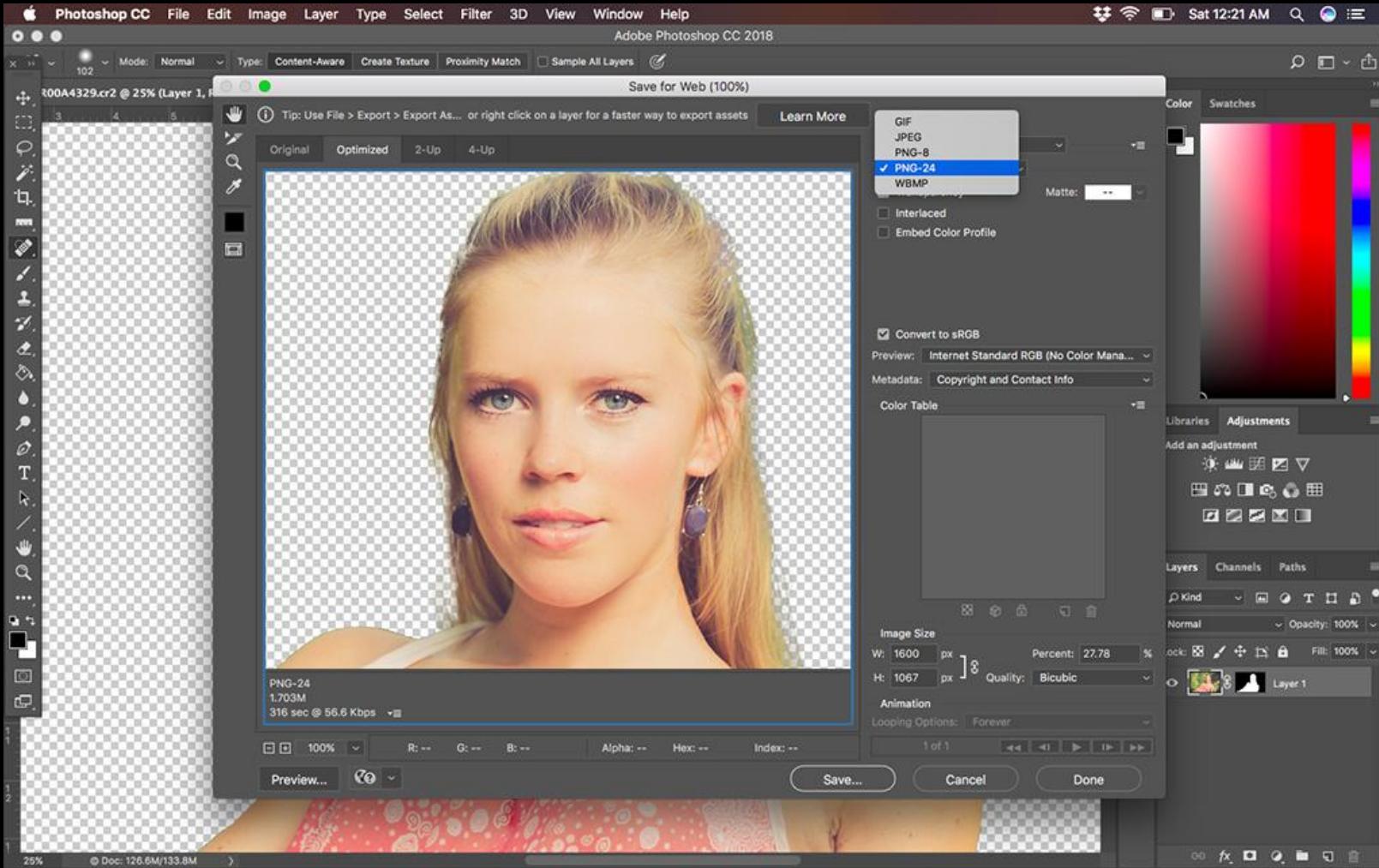
Original

Optimized

2-Up

4-Up







Introduction to Web Design

Raster Graphics



Pixlr



GIMP



Sketch



Affinity Photo

Introduction to Web Design

Raster Graphics



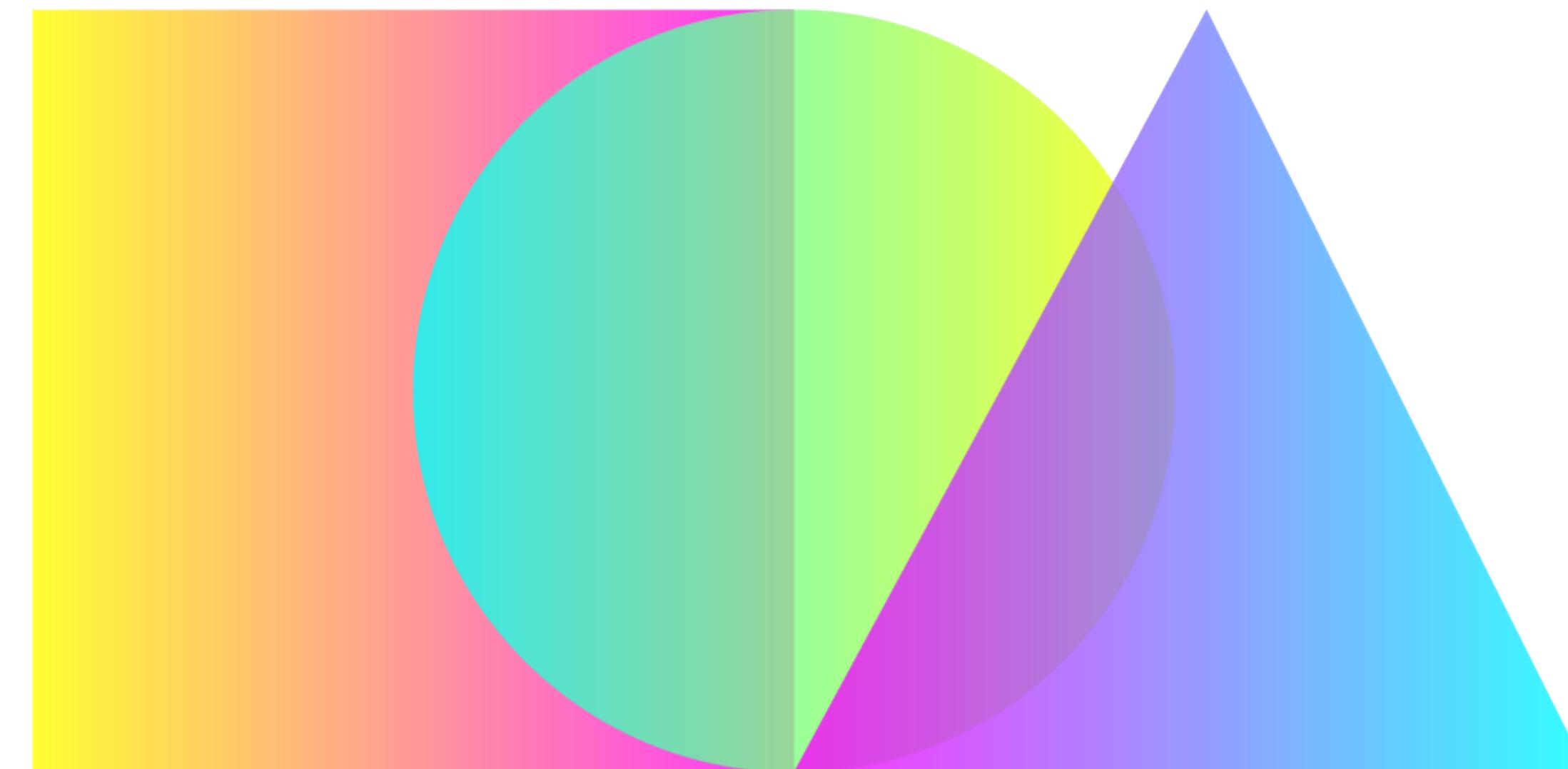
Photomontage by Wikipedia user: Mmxx

Introduction to Web Design

Raster
Graphics

Introduction to Web Design

Vector Graphics



Introduction to Web Design

Vector Graphics

Vector Graphics

Vector graphics contain geometric objects, such as lines and curves.

This has advantages compared to raster-only formats.

Since all modern displays are raster-oriented, the difference between raster-only and vector graphics comes down to where they are rasterized.

Vector graphics are “rasterized” client side; raster graphics are, by nature, already rasterized on the server.

Introduction to Web Design

Scalable Vector Graphics

Vector Graphics

Scalable Vector Graphics (SVG) is a markup language for describing two-dimensional graphics.

SVG allows for three types of graphic objects: vector graphic shapes, images, and text.

SVG drawings can be interactive and even styled with CSS.

SVG defines vector graphics in XML format.

Introduction to Web Design

Vector Graphics

XML

XML stands for “Extensible Markup Language”

It is a markup language designed to transport and store data.

Whereas HTML is about describing and displaying information, XML is about carrying information.

XML tags are not predefined; they are “extensible.”

Most XML grammars represent either textual information or raw data; they only provide rudimentary graphical capabilities.

SVG provides a rich, structured description of vector and mixed vector/raster graphics with pure XML.

Introduction to Web Design

Scalability

Vector Graphics

To be scalable means to increase or decrease uniformly.

In terms of graphics, it means not being limited to a single, fixed, pixel size.

On the web, scalability means that a particular technology can grow over time.

SVG is scalable in both senses of the word.

Introduction to Web Design

Vector Graphics

Advantages of SVG

SVG images can be created and edited with any text editor.

SVG images can be searched, indexed, scripted, and compressed.

SVG images are scalable, can be printed at any resolution, and are zoomable without degradation.

SVG is an open standard!

Introduction to Web Design

SVG and CSS

Vector Graphics

The advantages of style sheets are generally accepted, certainly for use with text and layout.

SVG extends this control to the realm of graphics.

It allows for script-based manipulation of the document tree and the style sheet.

Introduction to Web Design

Vector Graphics

SVG Path Element

The `<path>` element is foundational to drawing with SVG; it allows you to create all kinds of shapes.

The shape of a `<path>` element is defined by one attribute: `d`

The `d` attribute contains a series of commands and parameters used by those commands.

All of the commands also come in two variants: an uppercase letter specifies absolute coordinates; a lowercase letter specifies relative coordinates.

Introduction to Web Design

SVG Path Commands

Vector Graphics

M	moveto
L	lineto
H	horizontal lineto
V	vertical lineto
C	curveto
S	smooth curveto
Q	quadratic Bézier curve
T	smooth quadratic Bézier curveto
A	elliptical arc
Z	closepath

Introduction to Web Design

Vector Graphics



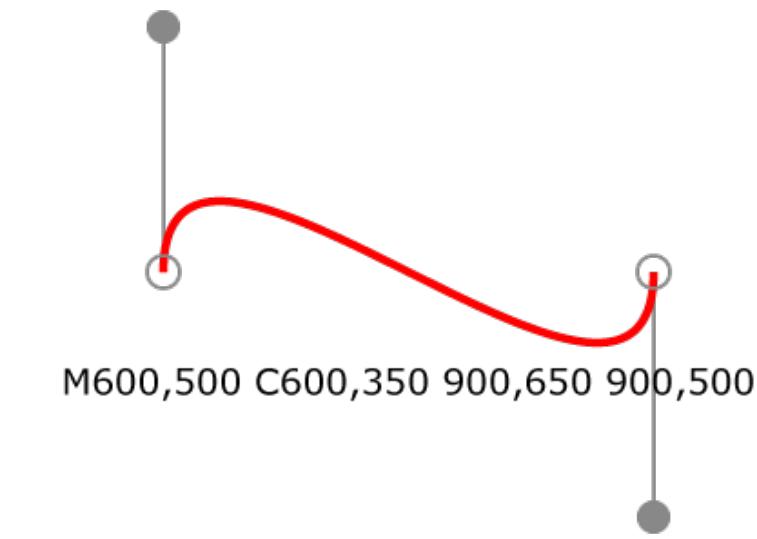
M100,200 C100,100 400,100 400,200



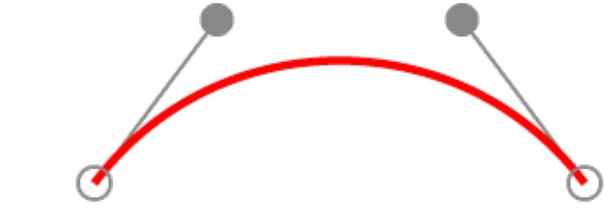
M600,200 C675,100 975,100 900,200



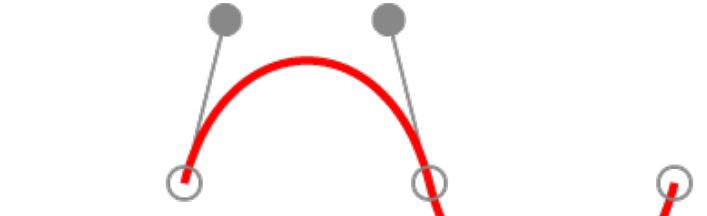
M100,500 C25,400 475,400 400,500



M600,500 C600,350 900,650 900,500



M100,800 C175,700 325,700 400,800



M600,800 C625,700 725,700 750,800
S875,900 900,800

[W3C: SVG Paths](#)

Introduction to Web Design

SVG on the Web

Vector Graphics

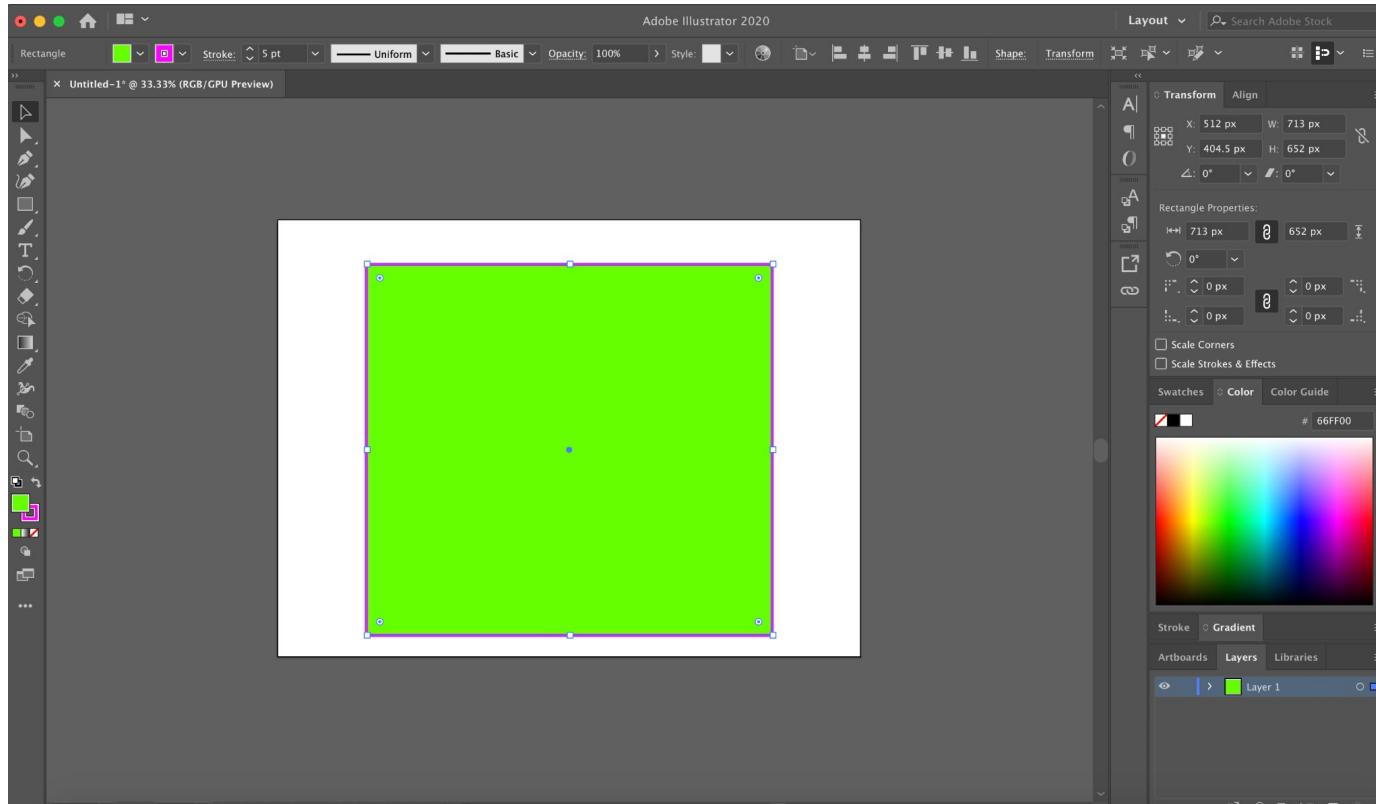
There are several ways in which SVG content can be included within a web page.

- A stand-alone SVG web page
- Embedding by reference, using the HTML element
- Embedding SVG code inline with HTML
- From an external link, using the HTML <a> element
- Referenced from a CSS property

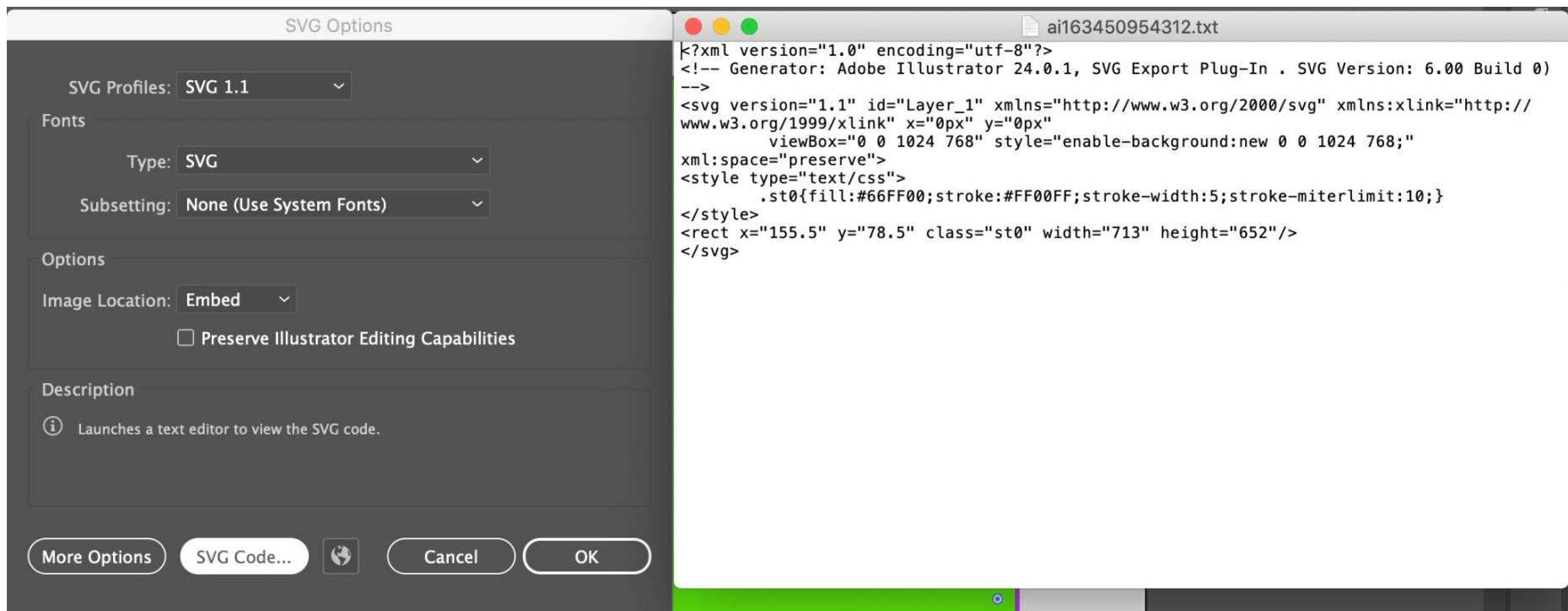
SVG GRAPHICS

2 METHODS

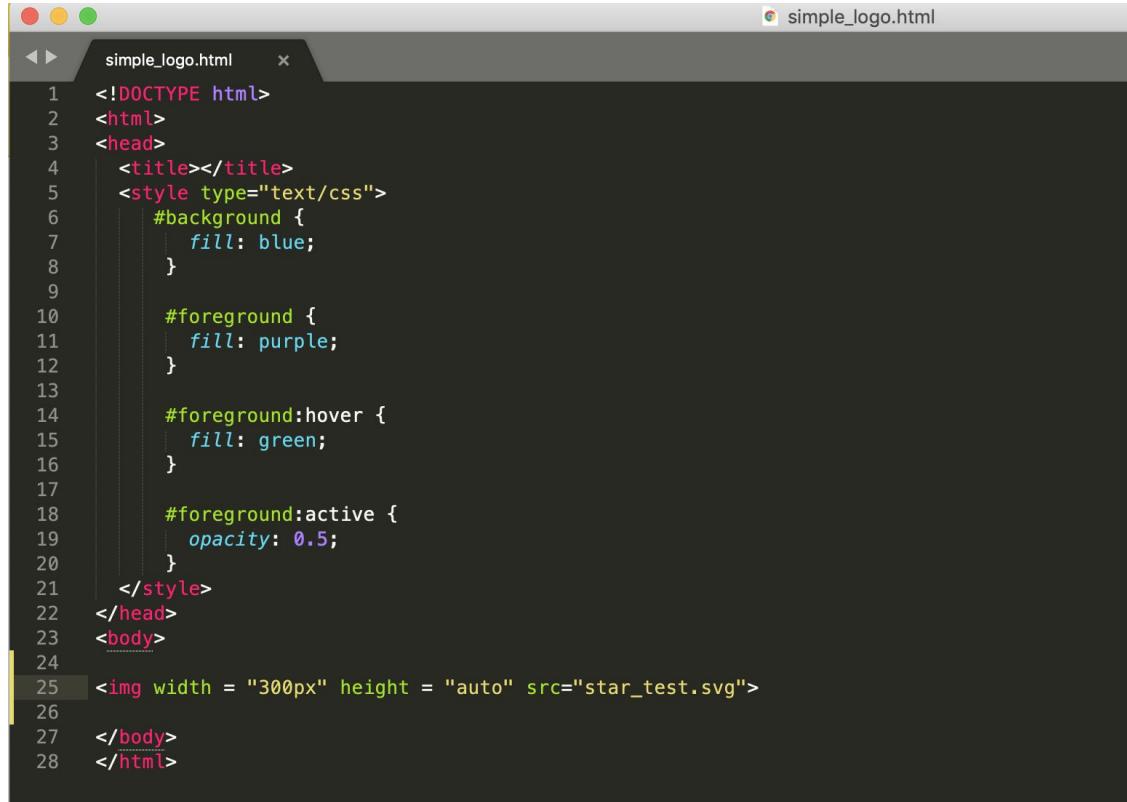
USING ADOBE ILLUSTRATOR



save AS "SVG" - uncheck preserve Illustrator editing capabilities



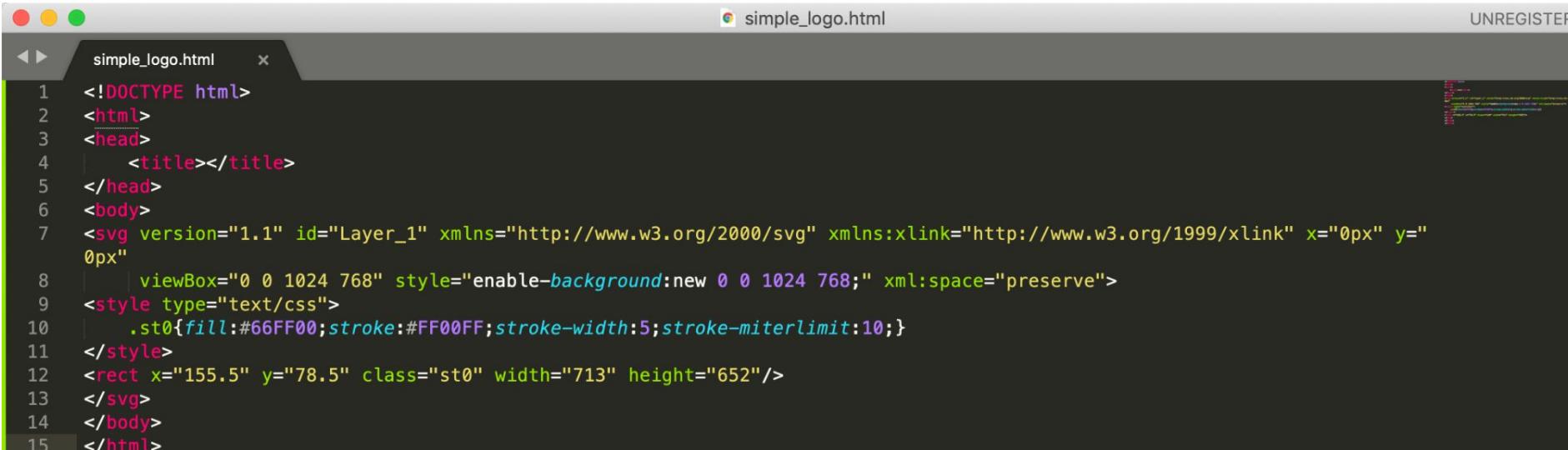
YOU CAN LINK



A screenshot of a code editor window titled "simple_logo.html". The code is written in HTML and CSS. It includes a style block with rules for "#background", "#foreground", "#foreground:hover", and "#foreground:active". The "#background" rule sets the fill color to blue. The "#foreground" rule sets the fill color to purple. The "#foreground:hover" rule sets the fill color to green. The "#foreground:active" rule sets the opacity to 0.5. The HTML part consists of a single tag with a width of 300px and an auto height, pointing to "star_test.svg".

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <style type="text/css">
        #background {
            fill: blue;
        }
        #foreground {
            fill: purple;
        }
        #foreground:hover {
            fill: green;
        }
        #foreground:active {
            opacity: 0.5;
        }
    </style>
</head>
<body>
    
</body>
</html>
```

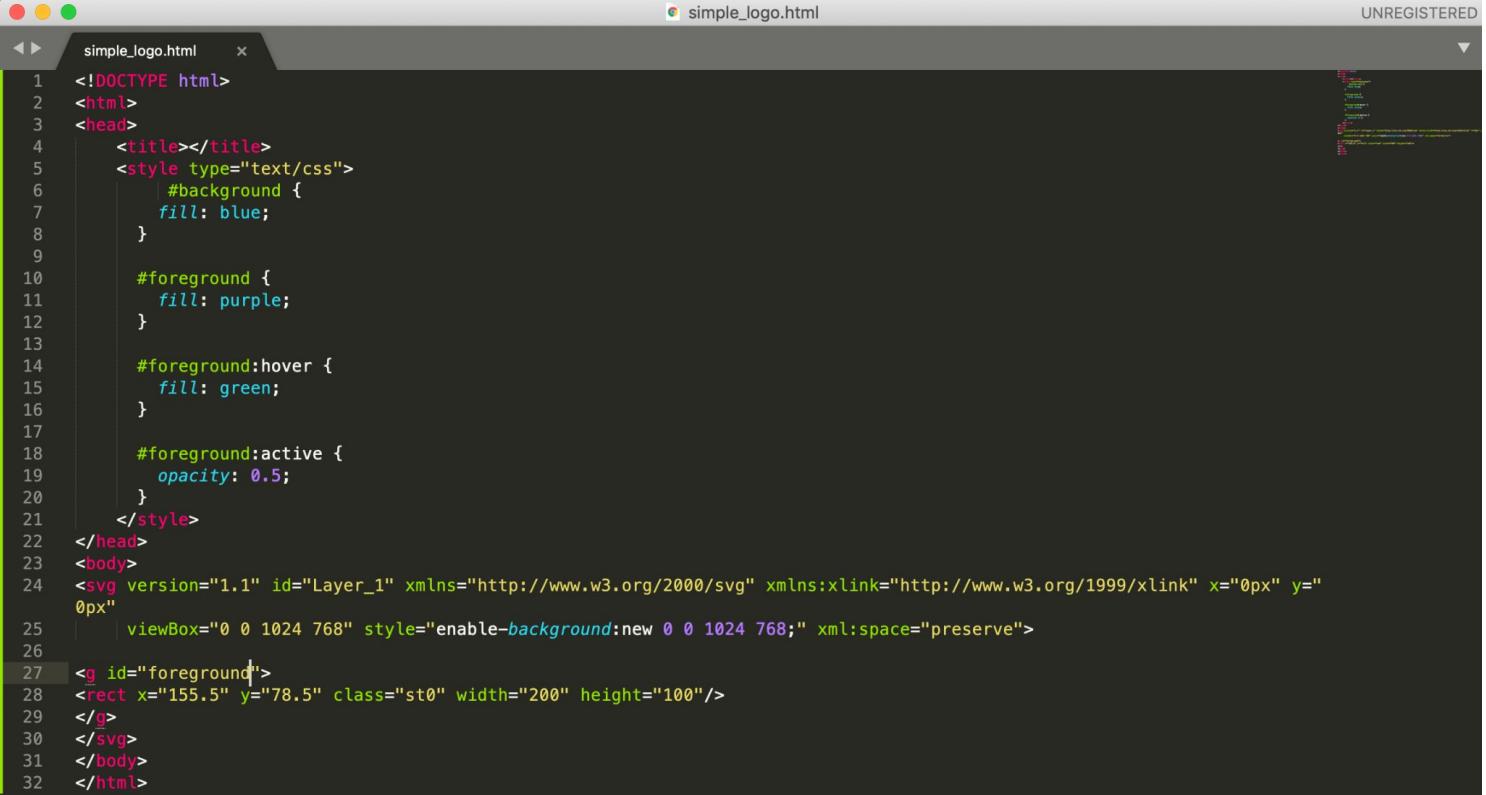
OR EMBED ONTO a WEBPAGE



A screenshot of a web browser window titled "simple_logo.html". The window shows the source code of an HTML file. The code includes an SVG element containing a single red rectangle. The browser interface includes standard window controls (red, yellow, green) and a status bar at the bottom right.

```
simple_logo.html      x      UNREGISTERED  
simple_logo.html  
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4   <title></title>  
5 </head>  
6 <body>  
7 <svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"  
8   | viewBox="0 0 1024 768" style="enable-background:new 0 0 1024 768;" xml:space="preserve">  
9 <style type="text/css">  
10  | .st0{fill:#66FF00;stroke:#FF00FF;stroke-width:5;stroke-miterlimit:10;}  
11 </style>  
12 <rect x="155.5" y="78.5" class="st0" width="713" height="652"/>  
13 </svg>  
14 </body>  
15 </html>
```

ADD STYLING ON THE CSS END



A screenshot of a code editor window titled "simple_logo.html". The code is written in HTML and CSS. The CSS styles define a blue background, a purple foreground rectangle, and a green foreground rectangle on hover. The HTML structure includes an SVG element with a viewBox of 0 0 1024 768 and a g element containing a rect with id="foreground". The code is numbered from 1 to 32.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <style type="text/css">
        #background {
            fill: blue;
        }

        #foreground {
            fill: purple;
        }

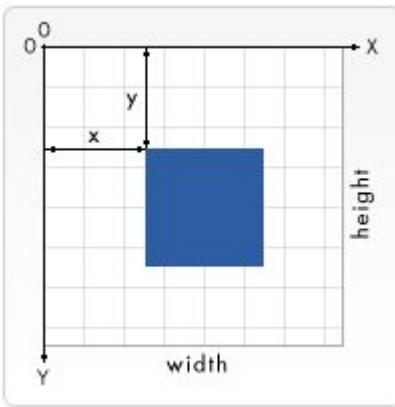
        #foreground:hover {
            fill: green;
        }

        #foreground:active {
            opacity: 0.5;
        }
    </style>
</head>
<body>
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
      viewBox="0 0 1024 768" style="enable-background:new 0 0 1024 768;" xml:space="preserve">
<g id="foreground">
    <rect x="155.5" y="78.5" class="st0" width="200" height="100"/>
</g>
</svg>
</body>
</html>
```

PURE CODE SOLUTIONS

Scalable Vector Graphics, [SVG](#), is a W3C XML dialect to mark up graphics.

POSITIONS



For all elements, SVG uses a coordinate system or **grid** system similar to the one used by [canvas](#) (and by a whole lot of other computer drawing routines). That is, the top left corner of the document is considered to be the point (0,0), or point of origin. Positions are then measured in pixels from the top left corner, with the positive x direction being to the right, and the positive y direction being to the bottom.

Note that this is slightly different than the way you're taught to graph as a kid (y axis is flipped). However, this is the same way elements in HTML are positioned (By default, LTR documents are considered not the RTL documents which position X from right-to-left).

example



A screenshot of a web browser window titled "tutorial_slides.html". The window shows the source code of an HTML file with line numbers and syntax highlighting. The code includes an SVG element with a red rectangle.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title></title>
5  </head>
6  <body>
7  |   <svg version="1.1"
8  |       width="100%" height="100%"
9  |       xmlns="http://www.w3.org/2000/svg">
10 |     <rect x="50" y="50" width="100" height="100" fill="red" stroke="green" />
11 |   </svg>
12 |
13 | </body>
14 |
15 | </html>
```

SHAPE PRIMITIVES



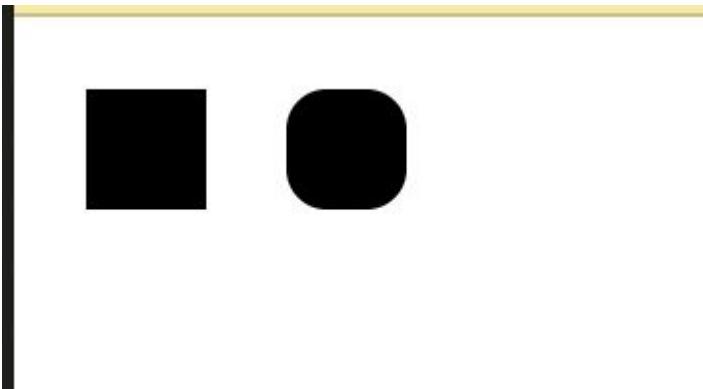
- rectangle : <`rect x y width height`>
- circle : <`circle cx cy radius`>
- ellipse : <`ellipse cx cy rx ry`>
- line : <`line x1 x2 y1 y2 stroke stroke-width`>
- polyline : <`polyline points`>
- polygon : <`polygon points`>
- path : <`path d fill stroke stroke-width`>

rectangle

The `<rect>` element draws a rectangle on the screen. There are 6 basic attributes that control the position and shape of the rectangles on screen. The one on the right has its `rx` and `ry` parameters set, giving it rounded corners. If they're not set, they default to 0.

```
<rect x="10" y="10" width="30" height="30"/>
```

```
<rect x="60" y="10" rx="10" ry="10" width="30" height="30"/>
```



`x`

The x position of the top left corner of the rectangle.

`y`

The y position of the top left corner of the rectangle.

`width`

The width of the rectangle

`height`

The height of the rectangle

`rx`

The x radius of the corners of the rectangle

`ry`

The y radius of the corners of the rectangle

CIRCLE

The `<circle>` element draws a circle on the screen. It takes 3 basic parameters to determine the shape and size of the element.

```
<circle cx="25" cy="75" r="20"/>
```

r

The radius of the circle.



cx

The x position of the center of the circle.

cy

The y position of the center of the circle.

ELLIPSE

An `<ellipse>` is a more general form of the `<circle>` element, where you can scale the x and y radius (commonly referred to as the *semimajor* and *semiminor* axes in maths) of the circle separately.

```
<ellipse cx="75" cy="75" rx="20" ry="5"/>
```

`rx`

The x radius of the ellipse.



`ry`

The y radius of the ellipse.

`cx`

The x position of the center of the ellipse.

`cy`

The y position of the center of the ellipse.

Line

The `<line>` element takes the positions of two points as parameters and draws a straight line between them.

```
<line x1="10" x2="50" y1="110" y2="150" stroke="black" stroke-width="5"/>
```

x1

The x position of point 1.

y1

The y position of point 1.

x2

The x position of point 2.

y2

The y position of point 2.



POLYLINE

A [polyline](#) is a group of connected straight lines. Since the list of points can get quite long, all the points are included in one attribute:

```
<polyline points="60, 110 65, 120 70, 115 75, 130 80, 125 85, 140 90, 135 95, 150 100, 145"/>
```

points

A list of points. Each number must be separated by a space, comma, EOL, or a line feed character. Each point must contain two numbers: an x coordinate and a y coordinate. So, the list (0,0), (1,1), and (2,2) would be written as
0, 0 1, 1 2, 2.



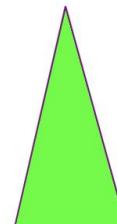
POLYGON

A `<polygon>` is similar to a `<polyline>`, in that it is composed of straight line segments connecting a list of points. For polygons though, the path automatically connects the last point with the first, creating a closed shape.

```
<polygon points="200,10 250,190 160,210" style="fill:lime;stroke:purple;stroke-width:1" />
```

points

A list of points, each number separated by a space, comma, EOL, or a line feed character. Each point must contain two numbers: an x coordinate and a y coordinate. So, the list `(0,0)`, `(1,1)`, and `(2,2)` would be written as `0, 0 1, 1 2`. The drawing then closes the path, so a final straight line would be drawn from `(2,2)` to `(0,0)`.



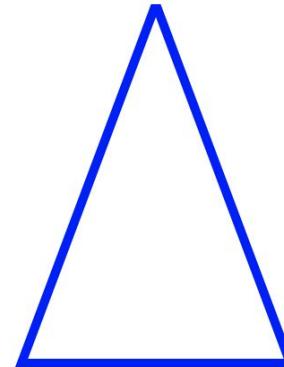
PATH

A `<path>` is the most general shape that can be used in SVG. Using a path element, you can draw rectangles (with or without rounded corners), circles, ellipses, polylines, and polygons. Basically any of the other types of shapes, bezier curves, quadratic curves, and many more.

```
<path d="M20,230 Q40,205 50,230 T90,230" fill="none" stroke="blue" stroke-width="5"/>
```

d

A list of points and other information about how to draw the path.



TEXT

The `text` element can be used to put arbitrary text in SVG documents:

```
<text x="10" y="10">Hello World!</text>
```

The `x` and `y` attributes determine, where in the viewport the text will appear. The attribute [`text-anchor`](#), which can have the values "start", "middle", "end" or "inherit", decides in which direction the text flows from this point. The attribute [`dominant-baseline`](#) decides the vertical alignment.

Like with the shape elements text can be colorized with the `fill` attribute and given a stroke with the `stroke` attribute. Both may also refer to gradients or patterns, which makes simple coloring text in SVG very powerful compared to CSS 2.1.

This is **bold and red**

ADDING CSS

STYLES AND PSEUDO-CLASSES

```
<!-->
<svg viewBox="0 0 480 480">
  <title>Star Logo</title>
  <g id="background">
    <path class="bg" d="M426.94,0H53.06A53.07,53.07,0,0,0,53.06V426.94A53.07,53.07,0,0,0,53.06,480H426.94A53.07,53.07,0,0,0,480,426.94V53.06A53.07,53.07,0,0,0,426.94,0ZM342.59,293.93L24.22,141.2L240,368.47,113.19,435.13L24.22-141.2-102.59-100,141.78-20.6L240,44.87L63.4,128.46,141.78,20.6Z"/>
  </g>
  <a href="https://www.nyu.edu">
    <g id="foreground">
      <polygon class="fg" points="240 97.34 286.35 191.26 390 206.32 315 279.43 332.7 382.66 240 333.92 147.29 382.66 165 279.43 90 206.32 193.65 191.26 240 97.34"/>
    </g>
  </a>
</svg>
```

```
<style>
  body {
    background-color: yellow;
  }

  #background {
    fill: blue;
  }

  #foreground {
    fill: purple;
  }

  #foreground:hover {
    fill: green;
  }

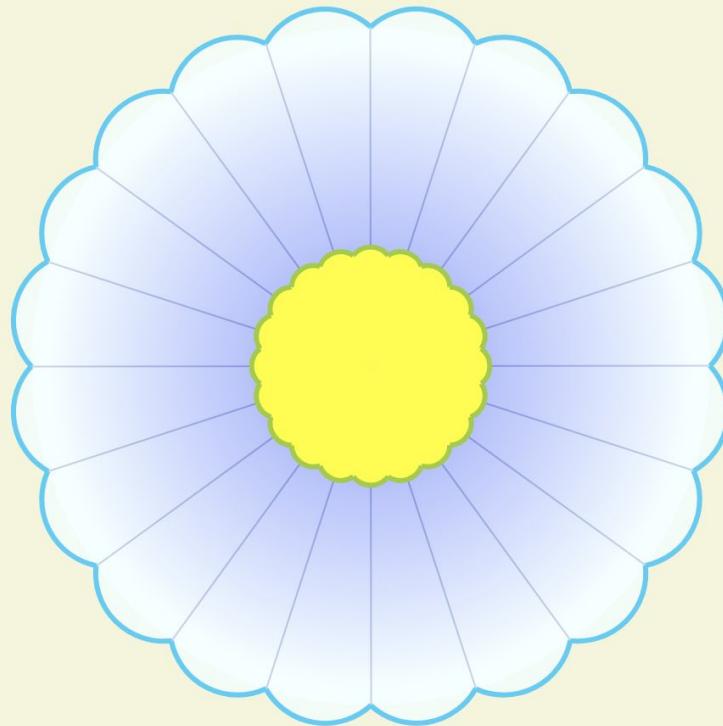
  #foreground:active {
    opacity: 0.5;
  }

  svg {
    width: 10em;
    height: auto;
  }
</style>
```

https://i6.cims.nyu.edu/~mr6465/web_design/svg/star-logo.html

SVG demonstration

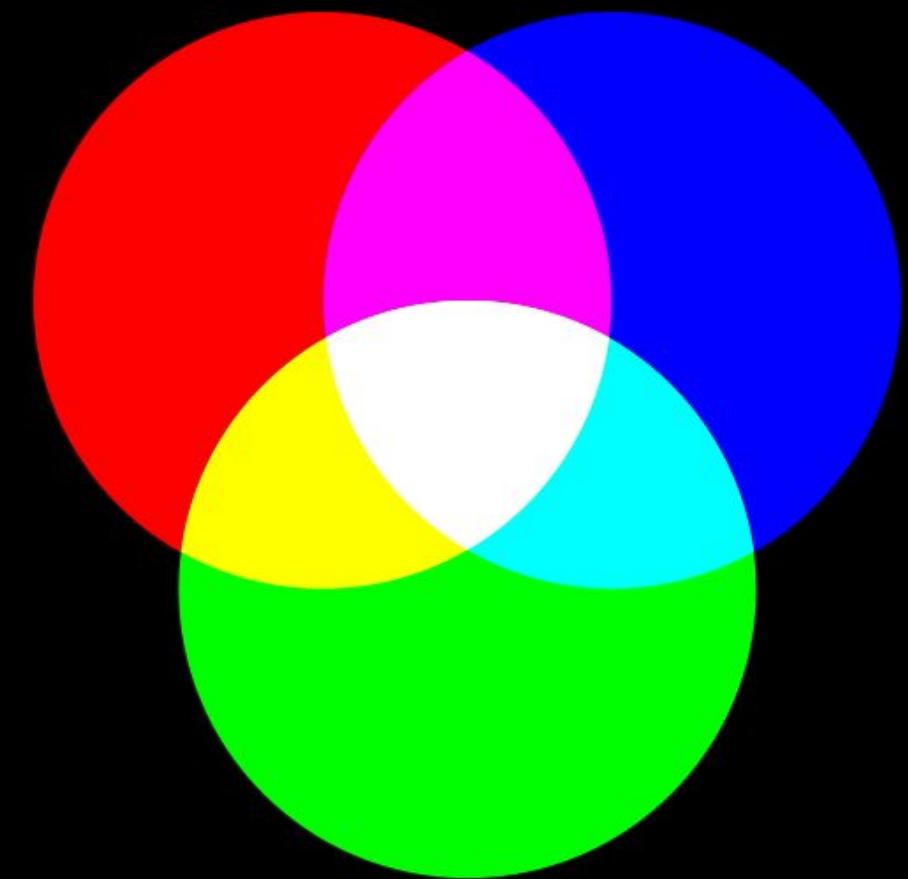
Move your mouse pointer over the flower.



https://i6.cims.nyu.edu/~mr6465/web_design/svg/example.html

Introduction to Web Design

Design and Accessibility



Introduction to Web Design

Design and Accessibility

Thought

Design is a process that involves thoughtfulness.

Design can encourage healthy (or unhealthy) habits, foster sustainable lifestyles (or not), and make us feel happy (or frustrated). Thoughtful design takes this important influence into account, and thoughtful designers carefully consider the implications on consumers' behaviour and habits when they craft brands, objects and interfaces. – Kristen Nozell, Ad Age

<https://www.youtube.com/watch?v=BBPjKqpODlc>

Introduction to Web Design

Form

Design and Accessibility

Image

Color

Typography

Composition

<https://www.youtube.com/watch?v=GJN7TemsZtY>

Introduction to Web Design

Form:
Image

Design and
Accessibility

Photography
Illustration
Line and shape
Texture

<https://blog.tubikstudio.com/web-design-basic-types-of-images-web-content/>

Introduction to Web Design

Form:
Color

Design and
Accessibility

Hue

Value

Intensity

<https://color.adobe.com/create/color-wheel>

Introduction to Web Design

Form: Typography

Design and Accessibility

Font selection

Type size

Alignment

Letter spacing

Line spacing

Grammar

<https://xd.adobe.com/ideas/principles/web-design/best-modern-fonts-for-websites/>

Introduction to Web Design

Form:
Composition

Design and
Accessibility

Rhythm
Proportion
Structure

Variation

Balance

Boundary

Space

<https://xd.adobe.com/ideas/process/ui-design/design-composition-key-principles/>

Introduction to Web Design

Context

Design and Accessibility

Device

Web browser

Age of visitor

Literacy

Geographic location

Language(s)

Ability

<https://xd.adobe.com/ideas/principles/web-design/12-dos-donts-web-design-2/>

Introduction to Web Design

Accessibility

Design and Accessibility

"When we talk about accessible code, what we are really talking about at its core is inclusiveness. The actual process . . . involves rules and standards, tests and tools; but inclusive development is more abstract than that. It's a shift in thinking . . . Inclusive development means making something valuable, not just accessible, to as many people as we can."

—Carie Fisher

[Getting to the Heart of Digital Accessibility](#)

Introduction to Web Design

Accessibility: Categories of Disability

Design and Accessibility

Vision impairment

Mobility impairment

Auditory impairment

Cognitive impairment

Introduction to Web Design

Design and Accessibility

Cultivating a mindful design approach allows you to do more with less.

Digital Accessibility

Digital Accessibility

Digital accessibility is the practice of ensuring that digital technology, including websites, mobile applications, immersive experiences, digital environments, and mixed reality, can be consumed by anybody or entity, regardless of visual, mobile, cognitive, and auditory abilities.



For example, somebody with weaker arms may need to use a mouthstick to type. A person with auditory issues uses captions to watch videos. Those using hearing aids, and those who are blind or have low vision will use a screen reader to read aloud what's on the screen. A person who has suffered a stroke may have difficulty using a mouse. An older individual with dexterity issues may have problems using a keyboard.

Universal Design

Needing accessible digital environments is also necessary for the non-disabled, and/or those compromised due to situations such as a broken limb, pregnancy, or simply juggling tasks and only having one hand to work with. Imagine being in a noisy airport and needing to watch a video concerning your flight and having lost your headphones. In this case, captions would be essential.

W3C

Access to digital technologies, including to the Internet, is a basic human right, according to the United Nations Convention on the Rights of Persons with Disabilities. Most of the international community has adopted this UN convention and other binding policies.

The World Wide Web Consortium (W3C) has created international standards for digital creations, in addition, the W3C Web Accessibility Initiative (WAI) has further developed standards and support materials to help designers, developers, content creators, designers and others in the field understand and implement accessibility.

"The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect," says Tim Berners-Lee, W3C Director and inventor of the World Wide Web W3C.

Perceivable

Provide text alternatives for non-text content.

Provide captions and other alternatives for multimedia.

Create content that can be presented in different ways, including by assistive technologies, without losing meaning.

Make it easier for users to see and hear content.

Operable

Make all functionality available from a keyboard.

Give users enough time to read and use the content.

Do not use content that causes seizures or physical reactions.

Help users navigate and find content.

Make it easier to use inputs other than the keyboard.

Understandable

Make text readable and understandable.

Make content appear and operate in predictable ways.

Help users avoid and correct mistakes.

Robust

Maximize compatibility with current and future user tools.

The practice of digital accessibility is deeply rooted in principles of Universal Design. Universal Design is the design and composition of an environment so that it can be accessed, understood, and used to the greatest extent possible by all people regardless of their age, size, ability, or disability. An environment (or any building, product, or service in that environment) should be designed to meet the needs of all people who wish to use it. This is not a special requirement, for the benefit of only a minority of the population. It is a fundamental condition of good design. (The Centre for Excellence in Universal Design.)

Who benefits from accessible websites? Identify those in your network that could benefit from universal design.

What are some social issues concerning creating inclusive spaces?

What is disability justice?

ac·ces·si·bil·i·ty



Noun

The quality of being easy to obtain, use, understand, reach, or enter.



What is ‘Web Accessibility’

- The inclusive practice of removing barriers that prevent interaction with, or access to, websites by people with disabilities.

[Web accessibility - Wikipedia](#)



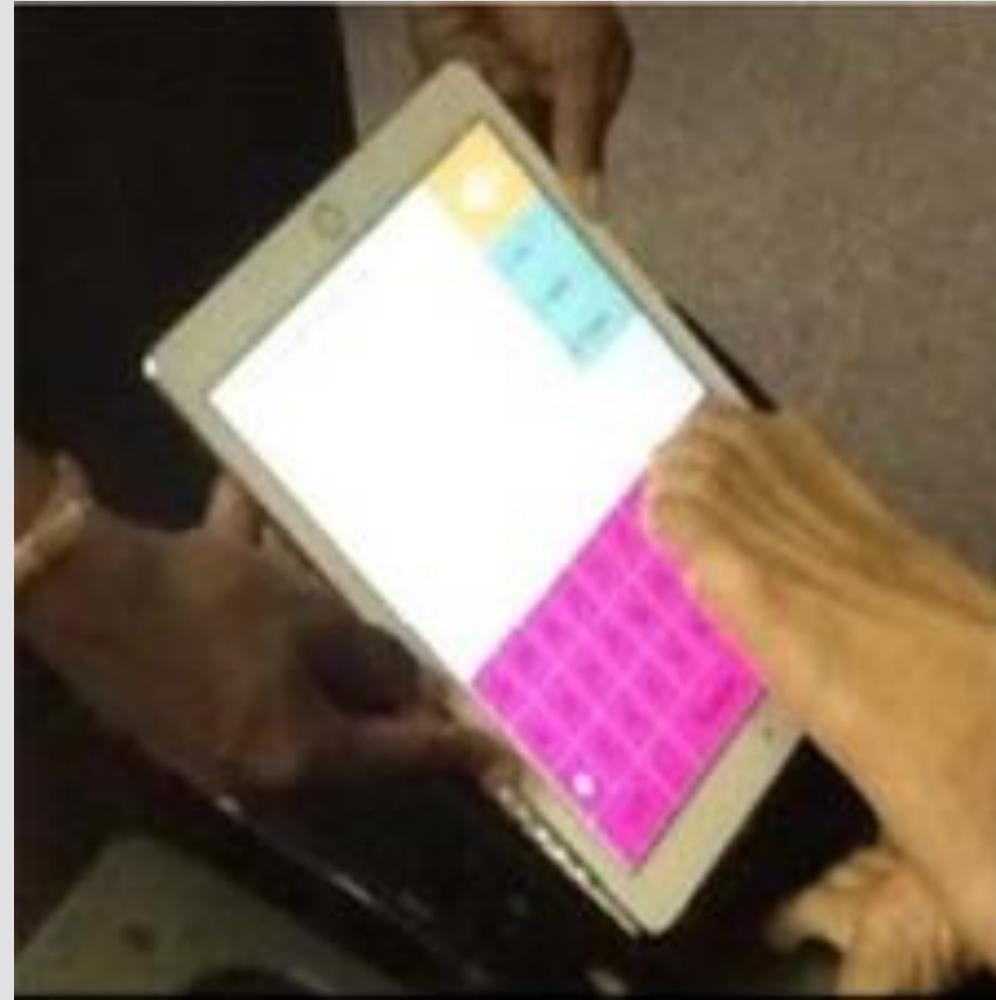
Assistive Technology

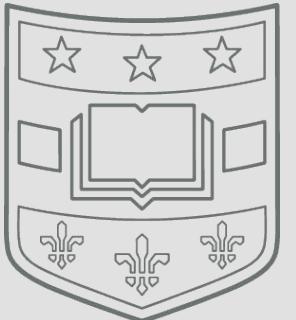
- Any item, piece of equipment, or product system, whether acquired commercially off the shelf, modified or customized, that is used to increase, maintain or improve functional capabilities of individuals with disabilities.
- Examples:
 - Screen Readers
 - Pointing devices
 - Switches
 - Alternate keyboards
 - Siri, Amazon Echo, etc.



Alternate Access Methods

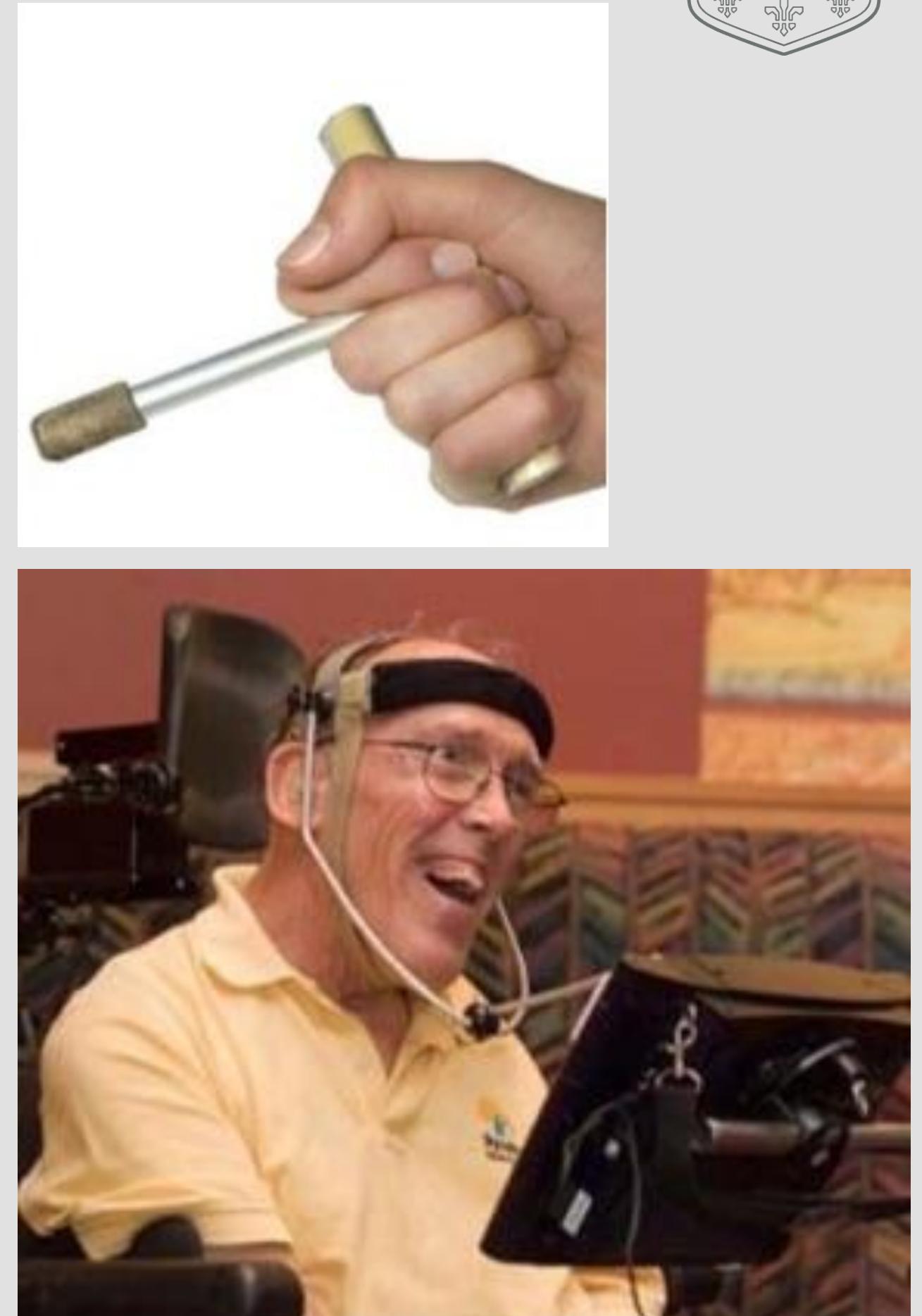
- Alternate keyboards
- Single button mice
- Trackballs / Joysticks
- Head pointing devices
- Pointing / Typing aides
- Head mice
- Switches / Onscreen keyboards
- Touch windows
- Eye gaze

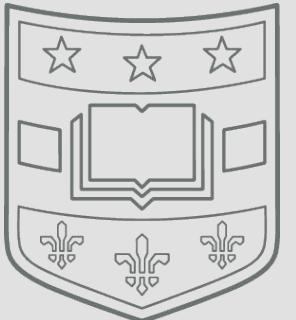




Low tech pointing devices can include:

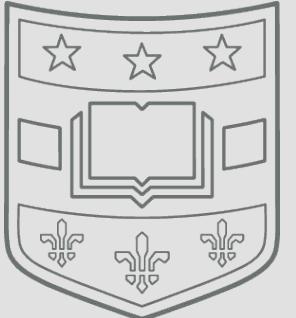
- Head / Chin pointers
- Styluses
- Adapted hand pointers
- Mouth sticks





2010 U.S. Census: Nearly 1 in 5 have disability

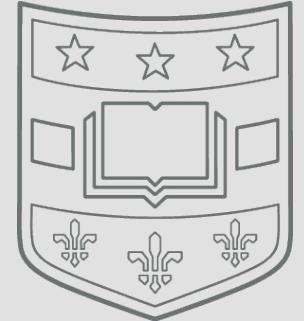
- About 56.7 million people, 19% of the population, had a disability in 2010, according to a broad definition of disability, with more than half of them reporting the disability was severe.
- About 8.1 million people had difficulty seeing, including 2.0 million who were blind or unable to see.
- About 7.6 million people experienced difficulty hearing, including 1.1 million whose difficulty was severe. About 5.6 million used a hearing aid.
- About 19.9 million people had difficulty lifting and grasping. This includes, for instance, trouble lifting an object like a bag of groceries, or grasping a glass or a pencil.



Web Content Accessibility Guidelines (WCAG) 2.0

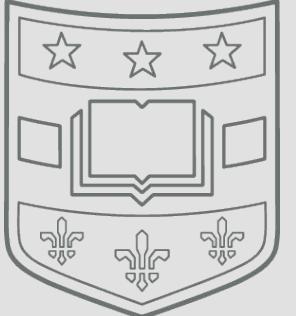
- (WCAG) 2.0 defines how to make Web content more accessible to people with disabilities. Accessibility involves a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities. Although these guidelines cover a wide range of issues, they are not able to address the needs of people with all types, degrees, and combinations of disability. **These guidelines also make Web content more usable by older individuals with changing abilities due to aging and often improve usability for users in general.**
- [Web Content Accessibility Guidelines \(WCAG\) 2.0](#)

WCAG 2.0

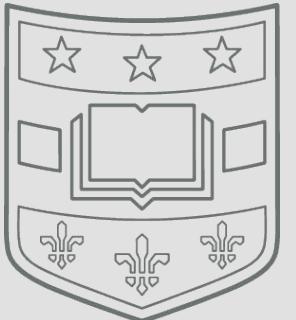


- The four guiding principles of accessibility in WCAG 2.0:
 - Perceivable
 - Operable
 - Understandable
 - Robust
- Three levels of conformance: A, AA, AAA
- Divided into 63 success criteria.

Section 508



- Applies to the federal government
- Adopted as part of the 2001 Amendment to the Rehabilitation Act
- Initially based on WCAG 1.0 (May 1999)
- Section508.gov



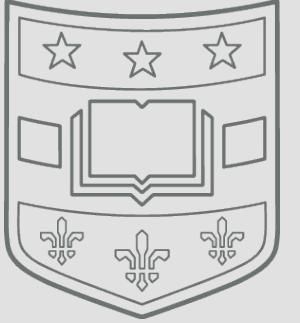
WCAG 2.0 vs. Section 508

- WCAG 2.0 Success Criteria are more explicit than the existing 508 Standards.
- WCAG 2.0 is written in a way that is technology neutral and is therefore directly applicable to a wide range of content types and formats.
- WCAG 2.0 has 38 Level A and AA Success Criteria. Of these, 22 are phrased differently but equivalent in substance to current 508 requirements.
- Section 508 is in the processes of being updated to align with WCAG 2.0 (level AA)
- [Comparison Table of WCAG 2.0 to Existing 508 Standards](#)



WCAG 2.1

- 28 additional success criteria are proposed that will be added to existing WCAG 2.0.
- Scheduled to be implemented in 2018
- [Web Content Accessibility Guidelines \(WCAG\) 2.1](#)

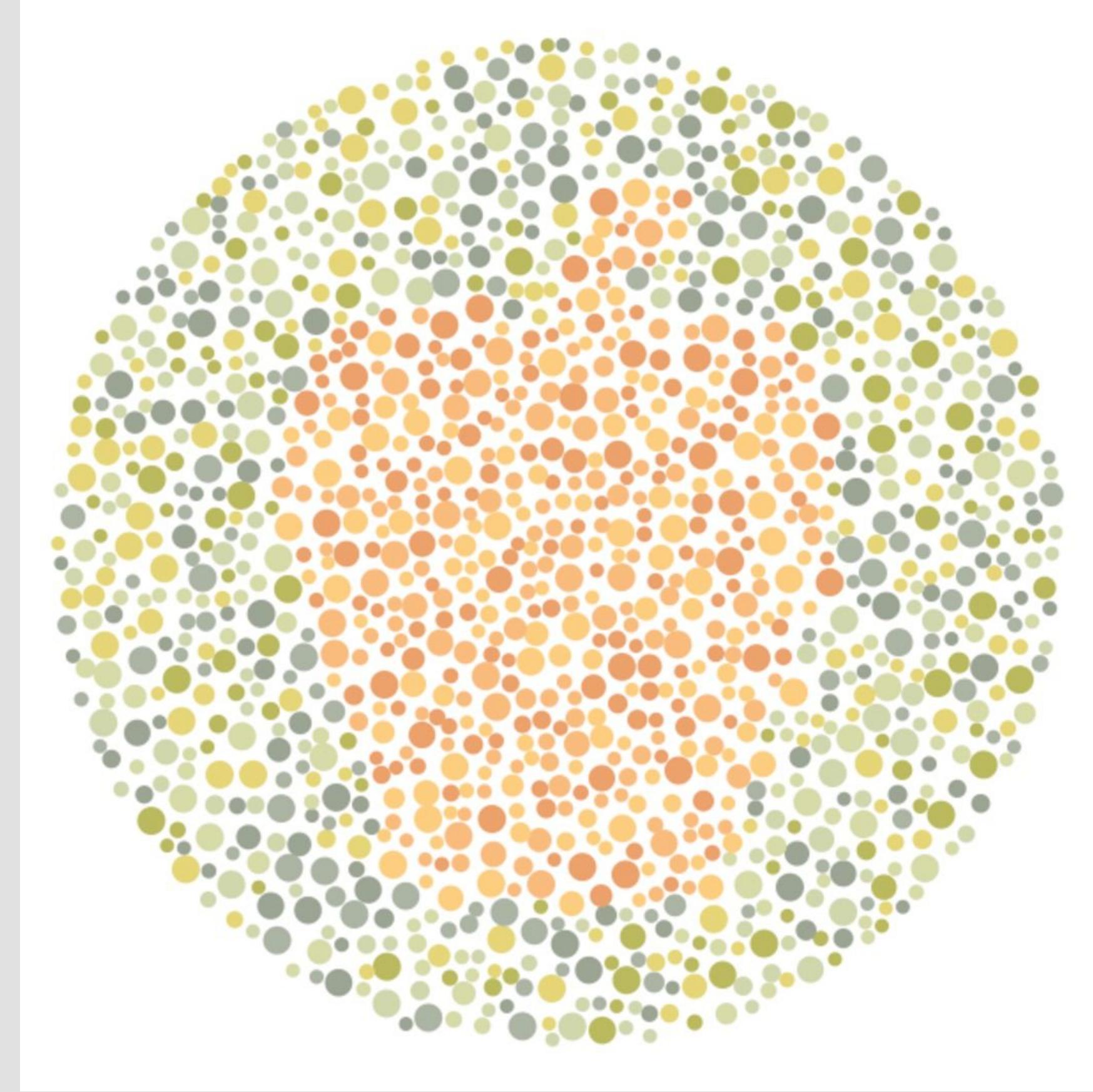


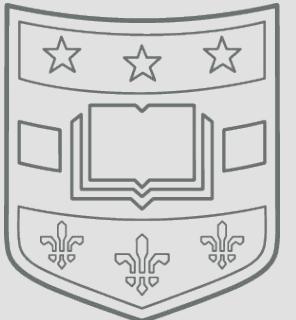
Spectrum of Disabilities

- Visual
- Auditory
- Physical
- Cognitive
- Speech

Visual Disabilities

- Low vision
 - Partial sight
 - Poor acuity
 - Tunnel vision
 - Clouded vision
- Color blindness
- Blindness





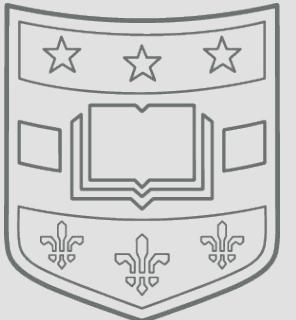
Examples of Good Practice for Visual Disabilities

- Images & controls should have equivalent text alternatives
- Text, images & page layouts can be resized without losing information.
- Video content has text or audio alternatives, or audio-description track.
- Text and images have sufficient contrast between foreground and background color.
- Provide consistent, predictable navigation.
- Avoid using color alone to identify links or controls.



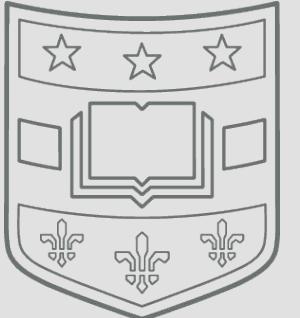
Auditory Disabilities

- Hard of hearing
- Deafness



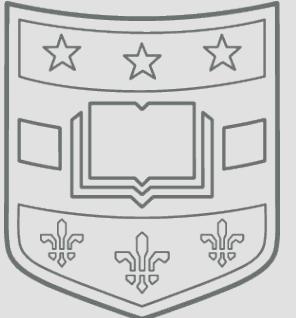
Examples of good practice for auditory disabilities

- Audio content, including videos, provide captions or transcripts.
- Media players provide volume controls.
- Media players provide options to adjust caption text size and colors.
- No interactions that rely on using voice only.



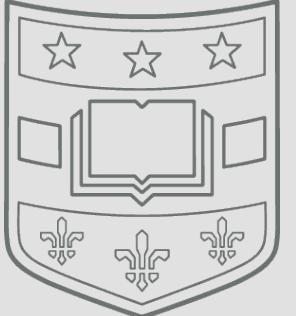
Examples of Physical Disabilities

- Amputation
- Arthritis
- Fibromyalgia
- Rheumatism
- Muscular dystrophy
- Repetitive stress injury
- Tremors and spasms
- Quadriplegia



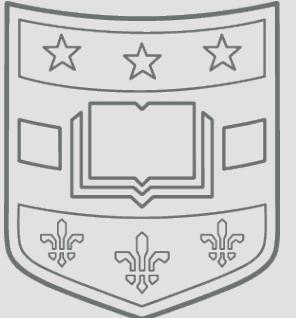
Good practice for physical disabilities

- Provide full keyboard support
 - All links, menu items, controls accessible via keyboard (Tab, Shift+Tab, & Return keys)
 - No keyboard traps
- Provide sufficient time to complete tasks.
- Provide consistent, predictable, simple navigation and page functions.
- Link targets, buttons should be of sufficient size.



Examples of cognitive disabilities

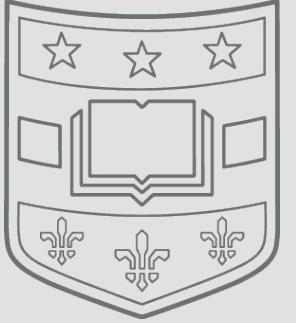
- Attention deficit hyperactivity disorder (ADHD)
- Autism spectrum disorder (ASD)
- Memory impairments
- Multiple sclerosis
- Perceptual or learning disorders
- Seizure disorders



Good practice for cognitive disorders

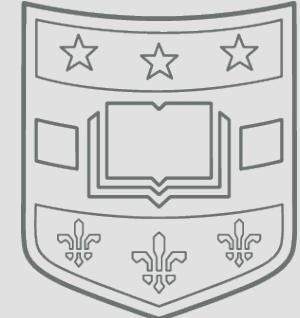
- Provide simple navigation and page layouts that are easy to understand and use.
- Avoid, when possible, complex sentences that are difficult to read or unusual words.
- Avoid moving, blinking, or flickering content. Or provide method to disable.
- Video, animations, or audio content can be paused or stopped.
- Simple text is supplemented by images, graphs, or illustrations.

Accessibility > Compliance



Your site can be compliant, yet inaccessible.

UC Berkeley Removes Online lectures



The image shows a screenshot of a YouTube channel page for 'UCBerkeley'. At the top, there's a purple square with a white 'U' and a banner that reads 'CONTENT UNAVAILABLE BEGINNING 3/15/2017'. Below this, a message states 'Beginning 3/15/2017 UC Berkeley course content will be removed from this site.' and provides a link 'For more information: ets.berkeley.edu/course-capture-changes'. On the right, there are links for the website (<http://webcast.berkeley.edu>) and Google+. The main channel area shows the title 'UCBerkeley' and navigation tabs for Home, Videos, Playlists, Channels, Discussion, and About. A search bar is also visible.

*In its review, the department looked at videos on UC-Berkeley's YouTube page, finding that automatically generated captions **weren't complete or accurate**, a barrier for those with hearing disabilities. Some videos also had issues that made them challenging for those with vision disabilities, such as low color contrast.*

The department [DOJ] found that the university was in violation of the federal disabilities law because “significant portions of its online content are not provided in an accessible manner when necessary to ensure effective communication with individuals with hearing, vision or manual disabilities.”

Source – [Why UC-Berkeley is restricting access to thousands of online lecture videos](#)



Checking for Accessibility

- Manual Testing
 - Keyboard check
 - Use with screen reader
- Online Tools
 - [WebAIM WAVE](#)

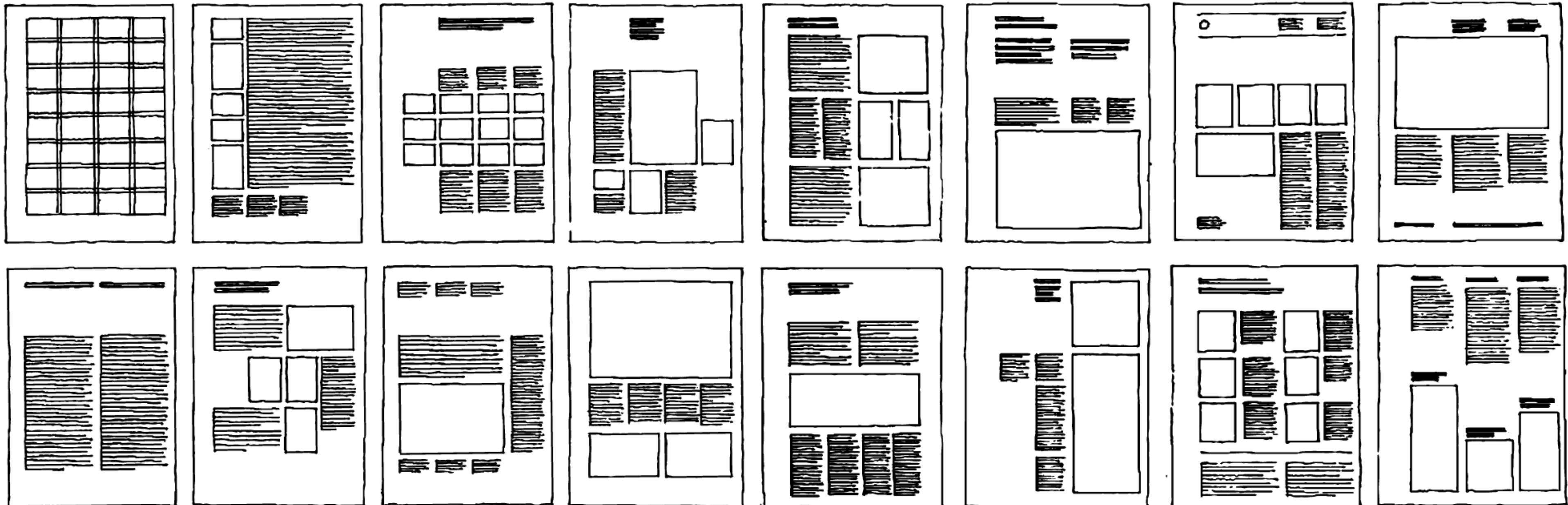


Accessibility Testing Demo

- Danforth University Center (DUC) homepage
- <http://wave.webaim.org/report?url=https://duc.wustl.edu>

Introduction to Web Design

Web Page Layout



Introduction to Web Design

Wireframing

Web Page Layout

Website wireframing allows you to plan the layout of your website.

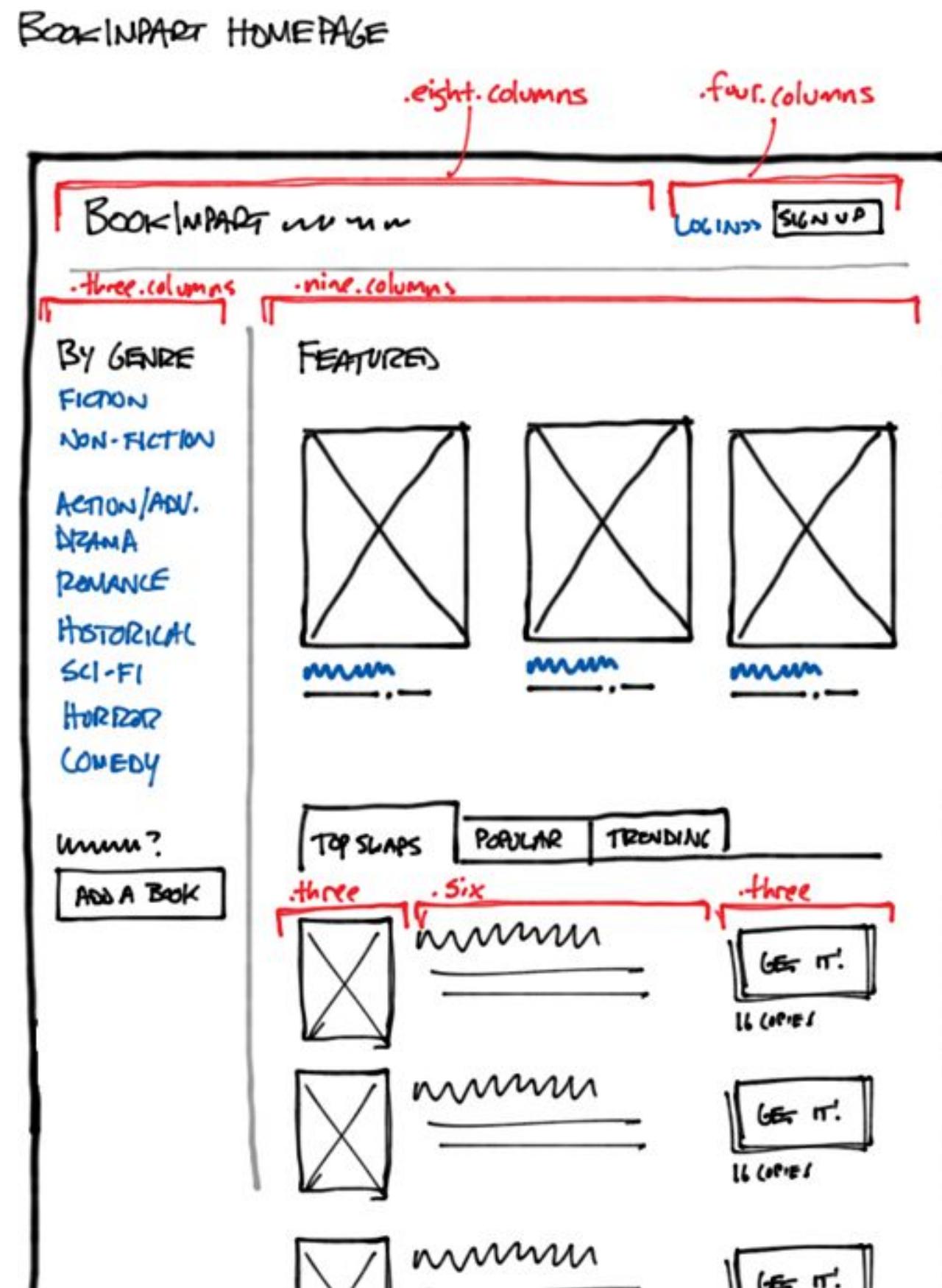
It is the process of making design decisions before they are implemented.

Wireframing can range from a simple skeletal framework to a detailed mockup of each page.

Spending time planning your site makes coding easier.

Introduction to Web Design

Web Page Layout



<https://www.figma.com>

Introduction to Web Design

Wireframing and Prototyping

Web Page Layout

Here is an approach to wireframing that can be adapted to a variety of design projects.

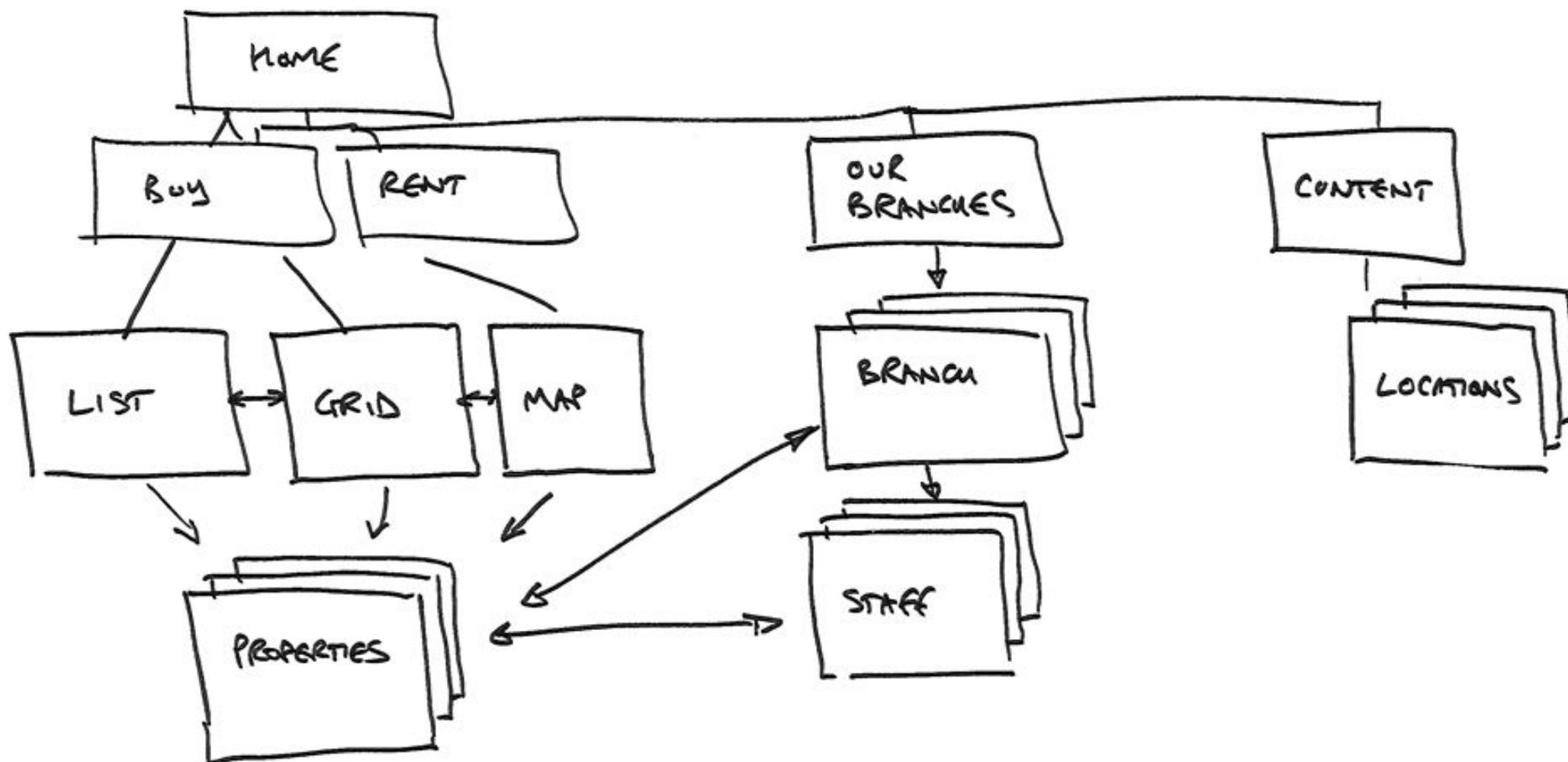
- Think
- Design
- Implement
- Revise

This sequence can be looped through as necessary.

Introduction to Web Design

Site Map

Web Page Layout



<https://octopus.do/>

Introduction to Web Design

Page Layout

Web Page Layout

There are several ways to design the layout of a web page with CSS.

- CSS float property
- CSS positioning
- CSS flexible box
- CSS grid

Introduction to Web Design

CSS Float Property

Web Page Layout

The CSS float property allows you to position block elements inline

This means that any element, block or inline, can be positioned alongside another element

The CSS float property is an outmoded technique of web page layout

https://i6.cims.nyu.edu/~mr6465/web_design/nyc/

Introduction to Web Design

CSS Positioning

https://i6.cims.nyu.edu/~mr6465/web_design/layout/css-positioning.html

Web Page Layout

The CSS position property specifies the type of positioning used for an element on a page.

static

Default document flow

absolute

Element is positioned relative to its first positioned (not static) parent element

fixed

Element is positioned relative to the browser window

relative

Element is positioned relative to its normal position

sticky

Positioned based on the user's scroll position

Introduction to Web Design

CSS Flexible Box

Web Page Layout

Use the CSS Flexible Box Layout Module (Flexbox) for arranging items along one axis.

Flexbox consists of flexible containers and flexible items within.

A flex container expands items to fill available free space or shrinks them to prevent overflow.

In practice, flexbox can accommodate different screen sizes and different display devices more easily than the CSS float property.

https://i6.cims.nyu.edu/~mr6465/web_design/layout/airports-main-2/

Introduction to Web Design

CSS Grid

Web Page Layout

Web pages are often laid out using grid systems.

CSS grids are intended to make this process more intuitive by defining a grid and then specifying where content should be placed within it.

The CSS Grid Layout Module can be used for the overall structure of a page.

https://i6.cims.nyu.edu/~mr6465/web_design/layout/seasons2/

Flexbox

flex containers + flex items

Flexbox

The “Flexible Box” or “Flexbox” layout mode offers an alternative to FLOATS for defining the overall appearance of a web page. Whereas floats only let us horizontally position our boxes, flexbox gives us *complete* control over the alignment, direction, order, and size of our boxes.

Flexbox vs Grid

CSS has always been used to lay out our web pages, but it's never done a very good job of it. First, we used tables, then floats, positioning and inline-block, but all of these methods were essentially hacks and left out a lot of important functionality (vertical centering, for instance). Flexbox helped out, but it's intended for simpler one-dimensional layouts, not complex two-dimensional ones (Flexbox and Grid actually work very well together).

Flexbox is generally used for smaller, simpler layouts and Grid for complicated 2-Dimensional ones. Flexbox is also very useful for handling a large amount of different content sizes.



ALIGNMENT



DIRECTION



ORDER

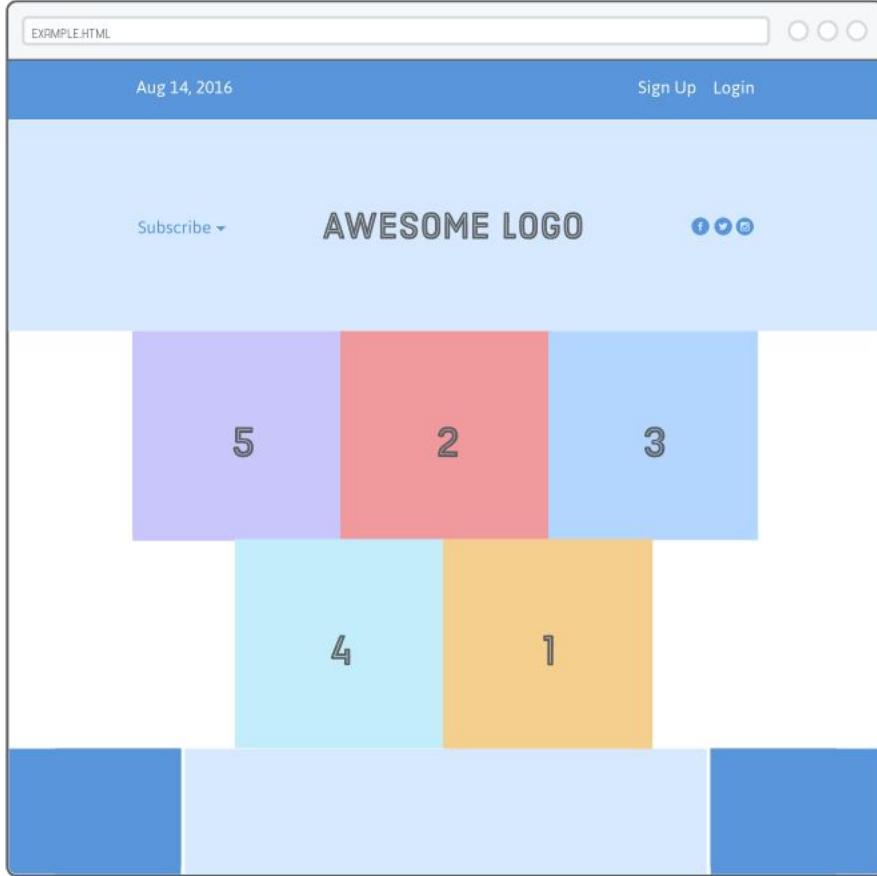


SIZE

Class files

class6/flexbox/flexbox.html

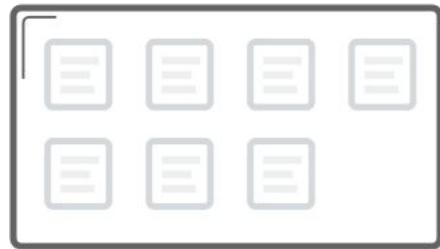
class6/flexbox/flexbox.css



Finished Example

Flexbox Overview

Flexbox uses two types of boxes: **flex containers** and **flex items**. The job of a flex container is to group a bunch of flex items together and define how they're positioned.



“FLEX CONTAINER”



“FLEX ITEMS”

Flex Containers

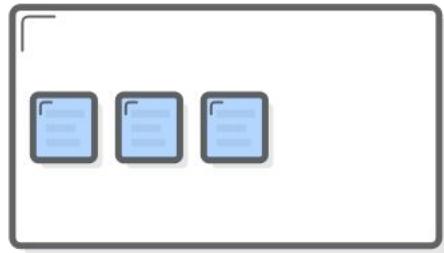
The first step in using flexbox is to turn one of our HTML elements into a flex container. We do this with the display property. By giving it a value of flex, we're telling the browser that everything in the box should be rendered with flexbox instead of the default box model.

```
.menu-container {  
  display: flex;  
  color: #fff;  
  background-color: #5995DA; /* Blue */  
  padding: 20px 0;  
}
```

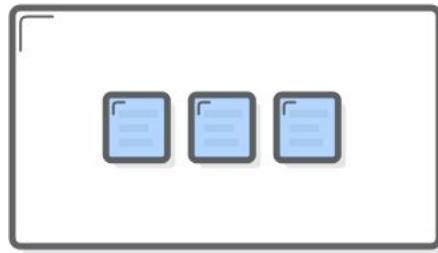
Aligning Flex Items

Now we can specify the horizontal alignment of the items:

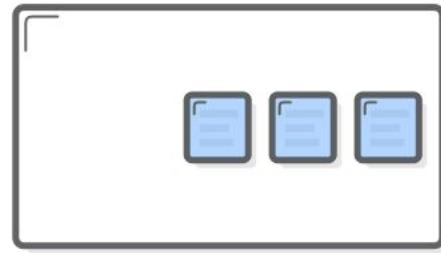
```
.menu-container {  
  display: flex;  
  justify-content: center;  
  color: #fff;  
  background-color: #5995DA; /* Blue */  
  padding: 20px 0;  
}
```



FLEX-START



CENTER



FLEX-END

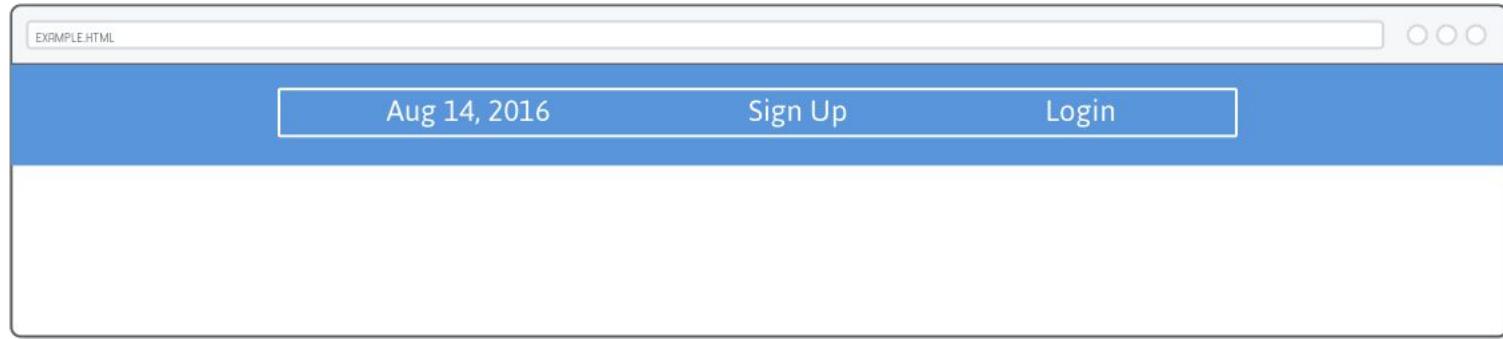
Other values for justify-content are shown below:

- center
- flex-start
- flex-end
- space-around
- space-between

Multiple Flex Items

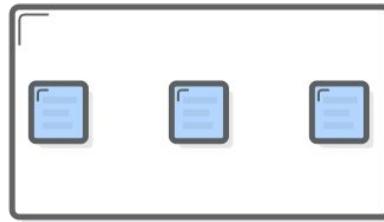
```
.menu {  
  display: flex;  
  justify-content: space-around;  
  border: 1px solid #fff; /* For debugging */  
  width: 900px;  
}
```

This turns our .menu into a nested flex container, and the space-around value spreads its items out across its entire width.



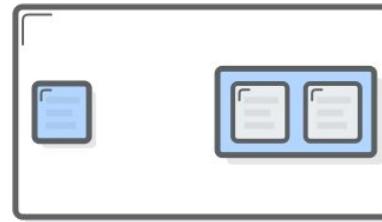
Grouping Flex Items

Flex containers only know how to position elements that are one level deep (i.e., their child elements). They don't care one bit about what's inside their flex items. This means that grouping flex items is another weapon in your layout-creation arsenal. Wrapping a bunch of items in an extra `<div>` results in a totally different web page.



NO GROUPING

(3 FLEX ITEMS)



GROUPED ITEMS

(2 FLEX ITEMS)

For example, let's say you want both the **Sign Up** and **Login** links to be on the right side of the page, as in the screenshot below. All we need to do is stick them in another `<div>`:

```
9   <div class='menu-container'>
10     <div class='menu'>
11       <div class='date'>Mar 18, 2019</div>
12       <div class='links'>
13         <div class='signup'>Sign Up</div>
14         <div class='login'>Login</div>
15       </div>
16     </div>
17   </div>
```



But, now we need to lay out the `.links` element because it's using the default block layout mode. The solution: more nested flex containers! Add a new rule to our `styles.css` file that turns the `.links` element into a flex container:

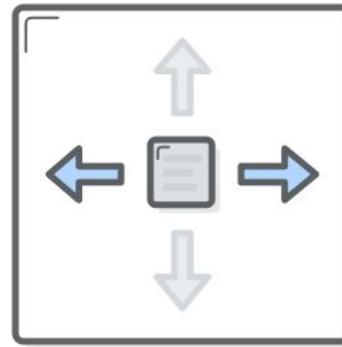
```
.links{
    border: 1px solid #fff; /* For debugging */
    display: flex;
    justify-content: flex-end;
}

.login{
    margin-left: 20px;
}
```

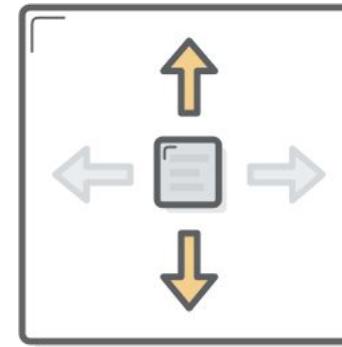


Cross-Axis (Vertical) Alignment

Flex containers can also define the vertical alignment of their items.

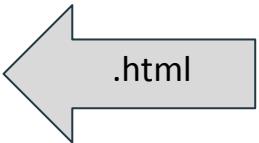


JUSTIFY-CONTENT

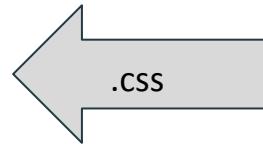


ALIGN-ITEMS

```
<div class='header-container'>
<div class='header'>
  <div class='subscribe'>Subscribe &#9662;</div>
  <div class='logo'><img src='images/awesome-logo.svg' /></div>
  <div class='social'><img src='images/social-icons.svg' /></div>
</div>
</div>
```



```
1 .header-container {
2   color: #5995DA;
3   background-color: #D6E9FE;
4   display: flex;
5   justify-content: center;
6 }
7
8 .header {
9   width: 900px;
10  height: 300px;
11  display: flex;
12  justify-content: space-between;
13  align-items: center;
14 }
```



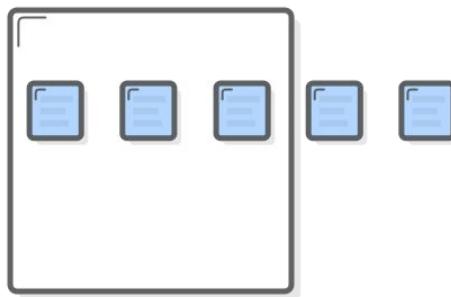
.html

.CSS



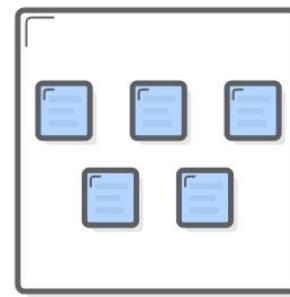
Wrapping flex items as a grid

Not only can it render items as a grid—it can change their alignment, direction, order, and size, too. To create a grid, we need the `flex-wrap` property.



NO WRAPPING

`FLEX-WRAP: NOWRAP;`



WITH WRAPPING

`FLEX-WRAP: WRAP;`

.html

```
<div class='photo-grid-container'>
  <div class='photo-grid'>
    <div class='photo-grid-item first-item'>
      <img src='images/one.svg'/>
    </div>
    <div class='photo-grid-item'>
      <img src='images/two.svg'/>
    </div>
    <div class='photo-grid-item'>
      <img src='images/three.svg'/>
    </div>
    <div class='photo-grid-item'>
      <img src='images/four.svg'/>
    </div>
    <div class='photo-grid-item last-item'>
      <img src='images/five.svg'/>
    </div>
  </div>
</div>
```

.css

```
.photo-grid-container {
  display: flex;
  justify-content: center;
}

.photo-grid {
  width: 900px;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
}

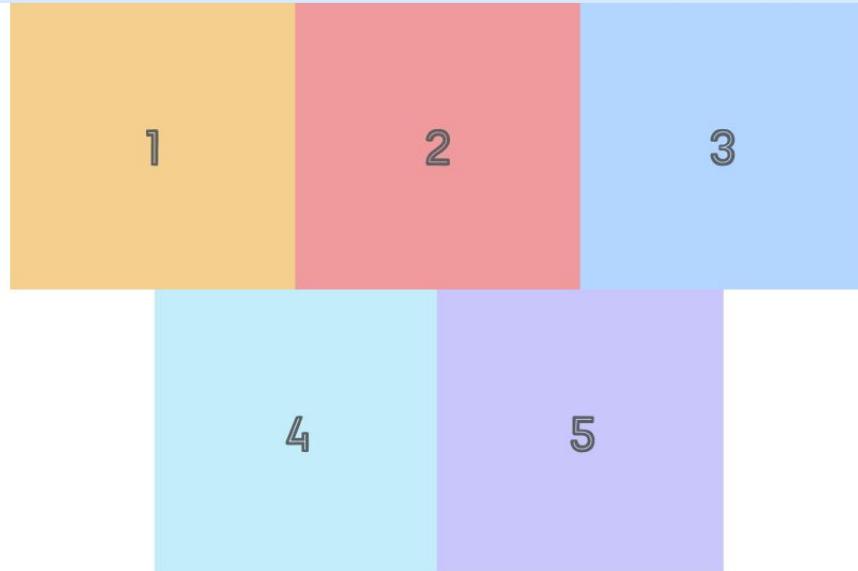
.photo-grid-item {
  border: 1px solid #fff;
  width: 300px;
  height: 300px;
}
```

Aug 14, 2016

[Sign Up](#) [Login](#)

[Subscribe](#) ▾

AWESOME LOGO



Individual items - order

Adding an `order` property to a flex item defines its order in the container without affecting surrounding items. Its default value is `0`, and increasing or decreasing it from there moves the item to the right or left, respectively.

This can be used, for example, to swap order of the `.first-item` and `.last-item` elements in our grid.

```
.first-item {  
  order: 1;  
}  
  
.last-item {  
  order: -1;  
}
```

Flex item alignment

We can do the same thing with vertical alignment. What if we want that **Subscribe** link and those social icons to go at the bottom of the header instead of the center? Align them individually! This is where the align-self property comes in. Adding this to a flex item overrides the align-items value from its container:

```
.social,  
.subscribe {  
  align-self: flex-end;  
  margin-bottom: 20px;  
}
```

Aug 14, 2016

Sign Up Login



AWESOME LOGO

Subscribe ▾



5

2

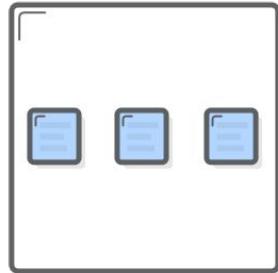
3

4

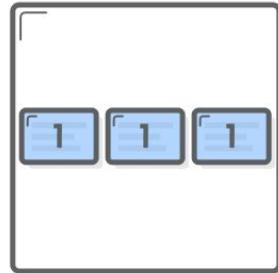
1

Flexible items

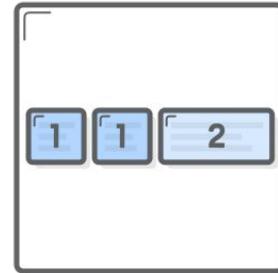
The flex property defines the width of individual items in a flex container. Or, more accurately, it allows them to have flexible widths. It works as a weight that tells the flex container how to distribute extra space to each item. For example, an item with a flex value of 2 will grow twice as fast as items with the default value of 1.



NO FLEX

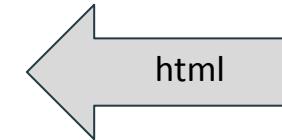


EQUAL FLEX



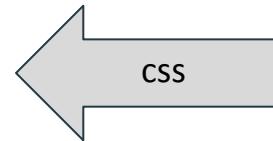
UNEQUAL FLEX

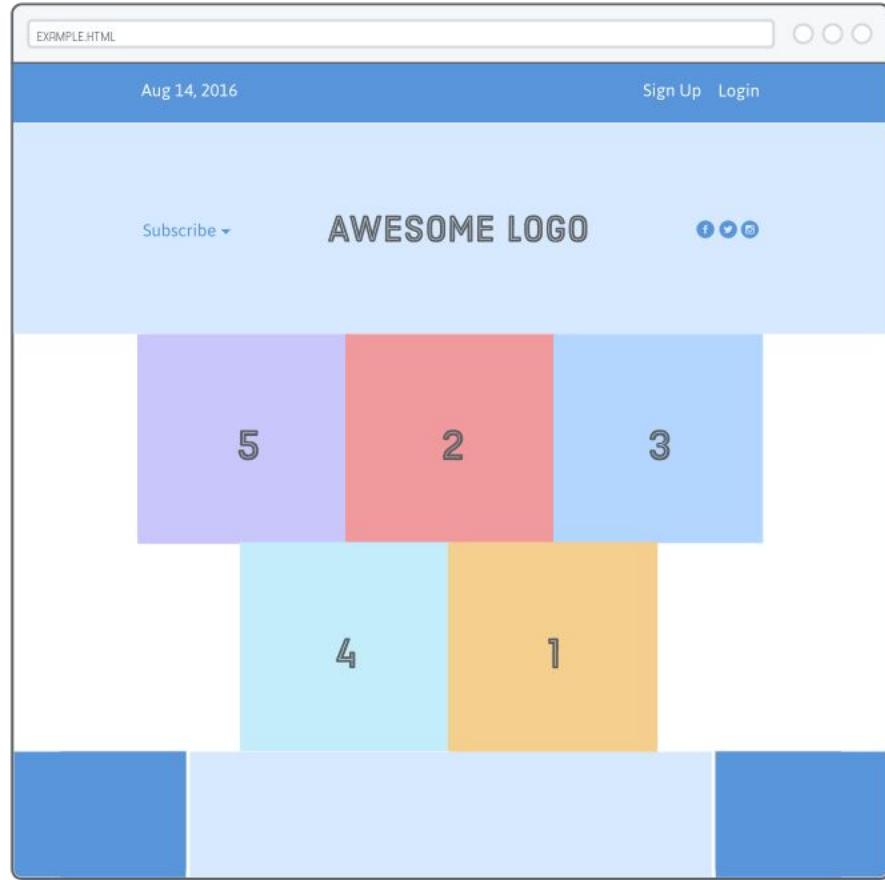
```
<div class='footer'>
  <div class='footer-item footer-one'></div>
  <div class='footer-item footer-two'></div>
  <div class='footer-item footer-three'></div>
</div>
```

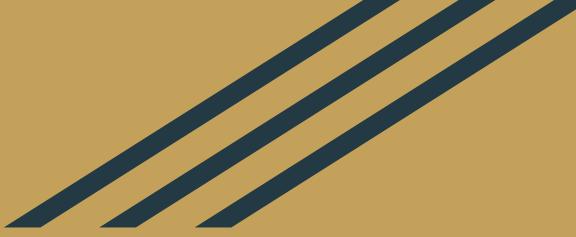


```
.footer {
  display: flex;
  justify-content: space-between;
}

.footer-item {
  border: 1px solid #fff;
  background-color: #D6E9FE;
  height: 200px;
  flex: 1;
}
.footer-two {
  flex: 3;
}
```

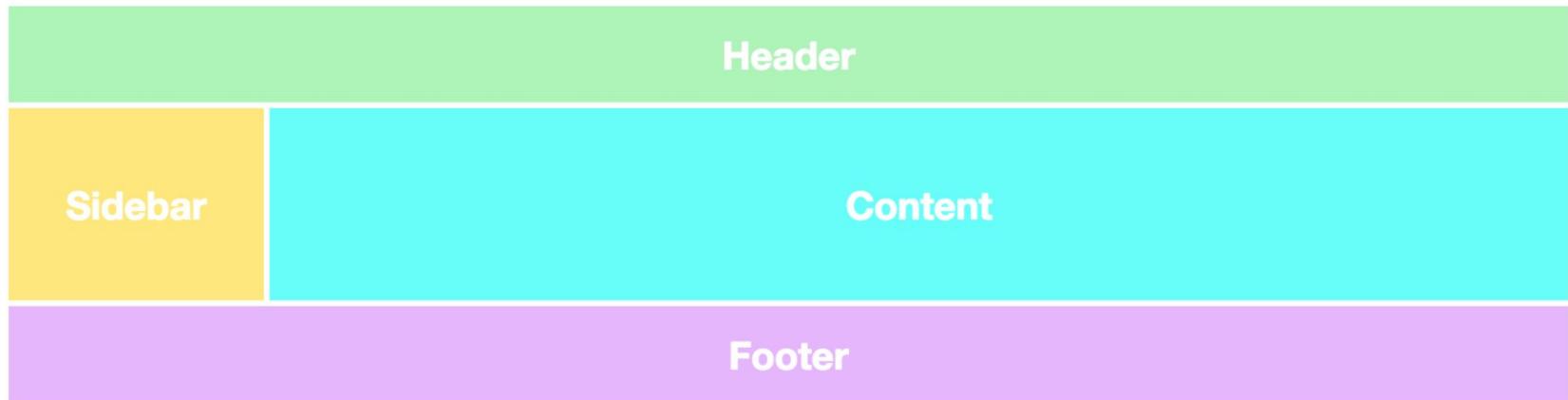






GRIDS

Grid layouts



Grid layouts are fundamental to the design of websites, and the CSS Grid module is the most powerful and easiest tool for creating it

Example Files

class5/floats_grid/grid.html

class5/floats_grid/grid_style/grid_style.css

Columns & Rows - 2 Dimensions

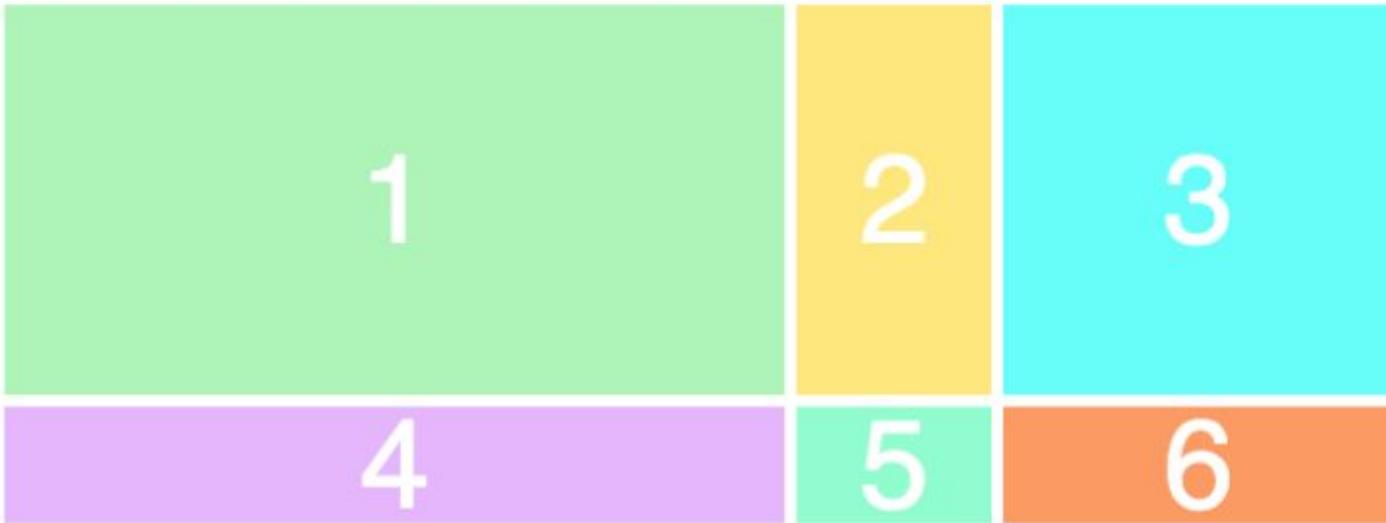
Let's change the display in the CSS file to grid and define columns and rows (similar to an .html table):

```
.wrapper {  
    display: grid;  
    grid-template-columns: 100px 100px 100px;  
    grid-template-rows: 50px 50px;  
}
```

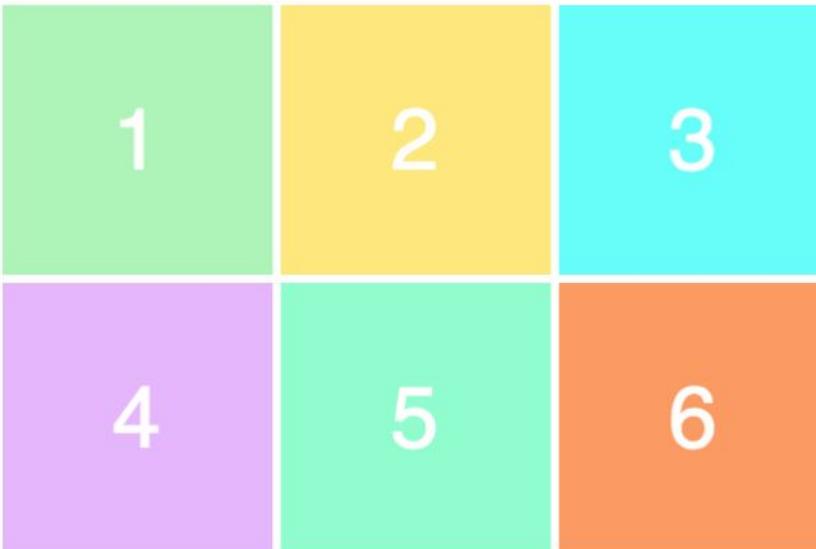
1	2	3
4	5	6

As we've written three values for `grid-template-columns`, we'll get three columns. We'll get two rows, as we've specified two values for the `grid-template-rows`.

The values dictate how wide we want our columns to be (100px) and how tall we'd want our rows to be (50px).



```
.wrapper {  
    display: grid;  
    grid-template-columns: 200px 50px 100px;  
    grid-template-rows: 100px 30px;  
}
```



```
.wrapper {  
    display: grid;  
    grid-template-columns: 100px 100px 100px;  
    grid-template-rows: 100px 100px 100px;  
}
```

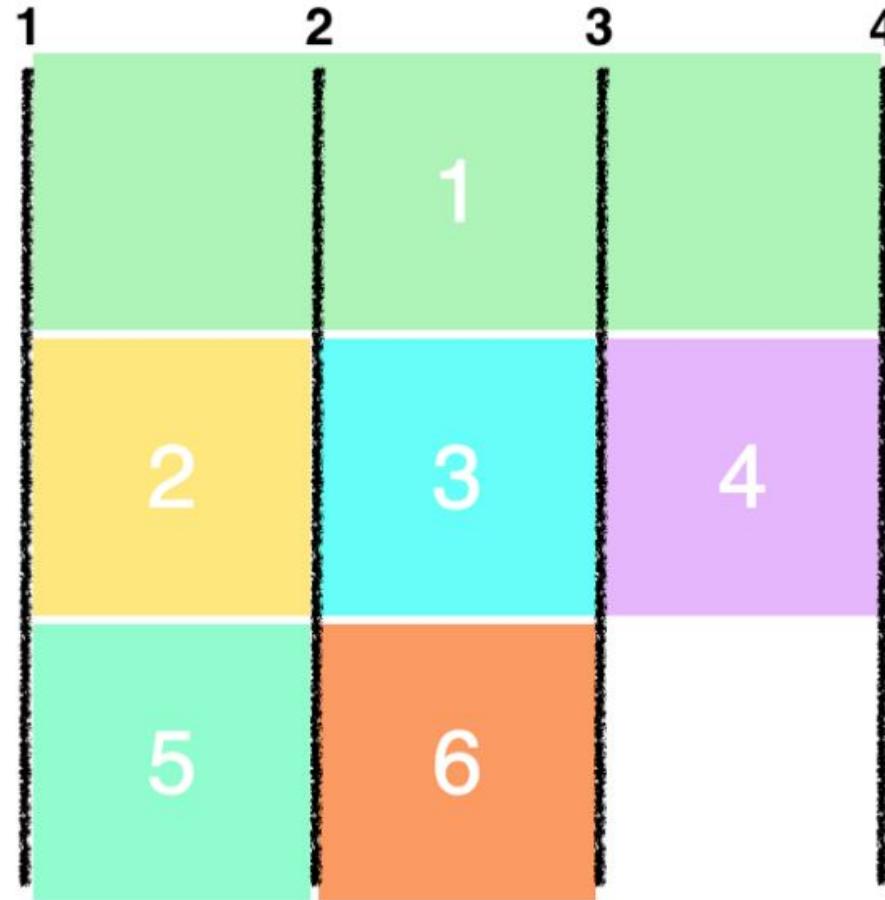
position & size

To position and resize the items we'll target them and use the `grid-column` and `grid-row` properties:

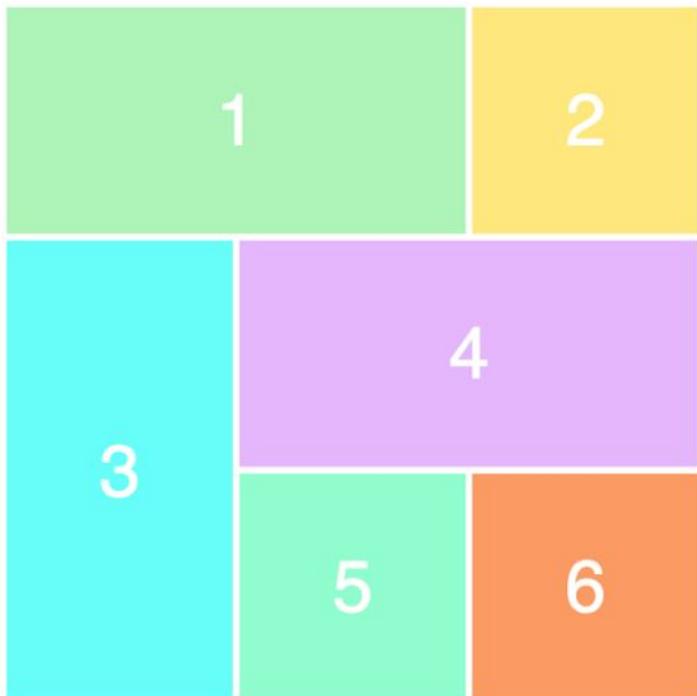
```
.item1 {  
    grid-column-start: 1;  
    grid-column-end: 4;  
}
```

What we're saying here is that we want item1 to start on the first grid line and end on the fourth column line. In other words, it'll take up the entire row.

Grid lines



```
.item1 {  
    grid-column-start: 1;  
    grid-column-end: 3;  
}  
  
.item3 {  
    grid-row-start: 2;  
    grid-row-end: 4;  
}  
  
.item4 {  
    grid-column-start: 2;  
    grid-column-end: 4;  
}
```



Static vs. Fluid Layouts

Static

A static page layout (sometimes called a “fixed” layout or “fixed width” layout) uses a preset page size and does not change based on the browser width. In other words, the page layout might have a permanent width of 960 pixels no matter what. This is how web pages were traditionally built for many years until modern influences like media queries and responsive design were introduced around the start of the 2010s.

Different devices will treat a static page layout in various ways, so the rendered page could be slightly unpredictable. For example, on a desktop browser, if the window is too small horizontally, then the page will be cut off and horizontal scroll bars will be displayed. On a mobile device like an iPhone, the page will be scaled automatically, allowing the user to zoom in on points of interest (provided that no metatags override this default behavior).

When new websites are built, most of them don’t opt for a static layout, because it means that the mobile experience will require a separate website. There are major exceptions, such as [the online Apple.com store](#), but Apple is a unique case because a selling point of their mobile devices is that they can view static layouts. In other words, Apple doesn’t seem to be adopting responsive design just yet.

Static Example File

class5/layouts/static-page-layout-example

Fluid

A liquid page layout (sometimes called “fluid” or “fluid width”) uses relative units instead of fixed units. Typically a liquid layout will use percentages instead of pixels, but any relative unit of measurement will work, such as ems.

A liquid layout often will fill the width of the page, no matter what the width of the browser might be. Liquid layouts don’t require as much thought as a responsive design, but there are some major drawbacks at very large or very small browser widths. If the browser is very wide, some content might be stretched too far. On large screens, a single paragraph might run across the page on a single line. Conversely, a multi-column layout on a small screen could be too crowded for the content.

Fluid Example File

class5/layouts/liquid-page-layout-example

Introduction to Web Design

Responsive Design



Introduction to Web Design

A Dao of Web Design

Responsive Design

"The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the fact that the web doesn't have the same constraints, and design for this flexibility. But first, we must 'accept the ebb and flow of things.'"

—John Allsopp, "A Dao of Web Design"

Introduction to Web Design

Responsive Design

Responsive Design

Mobile traffic is as (if not more) relevant as desktop traffic now.

We should build for the type of screens that will be used to access our sites.

Responsive web design uses “media queries” to figure out what resolution of device it’s being served on.

Flexible images and fluid grids size correctly to fit the screen.

Responsive web design is design for flexibility.

Introduction to Web Design

Foundations of Responsive Design

Responsive Design

Flexible grids (fluid layouts)

Media queries

Flexible, responsive images

Introduction to Web Design

Media Queries

Responsive Design

Features you can include in a media query include : width, height, device-width, device-height, orientation, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, scan grid

Most of the above can be combined with min- and max- prefixes.

The most common media queries assess min-width and max-width.

Media queries can be used to load an alternate style sheet or, more commonly, to offer alternate styles within an existing style sheet.

Introduction to Web Design

Media Query Syntax

Responsive Design

CSS media queries use the @media rule followed by two optional values: “only” or “not”

“only” screens out older browsers from reading the rest of the query.

“not” negates the result: “not screen” means everything except screen-based media.

A feature: value pair, enclosed by parentheses, comprises the essence of the media query.

Media features that can be assessed are predefined.

Multiple feature: value pairs can be combined with “and”.

Introduction to Web Design

CSS Rule Set

```
body {  
    background-color: orange;  
}
```

Responsive Design

CSS Rule Set with a Media Query

```
@media only screen and (min-width: 480px) {  
    body {  
        background-color: orange;  
    }  
}
```

CSS MEDIA QUERIES

|

A DAO OF WEB DESIGN

RESPONSIVE DESIGN

"The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the fact that the web doesn't have the same constraints, and design for this flexibility. But first, we must 'accept the ebb and flow of things.'"

—John Allsopp, "A Dao of Web Design"

<https://alistapart.com/article/dao/>

Media Queries

Responsive Design

Features you can include in a media query include :
width, height, device-width, device-height, orientation,
aspect-ratio, device-aspect-ratio, color, color-index,
monochrome, resolution, scan grid

Most of the above can be combined with min- and max- prefixes

The most common media queries assess min-width and max-width.

Media queries can be used to load an alternate style sheet or, more commonly, to offer alternate styles within an existing style sheet.

|

Media Query Syntax

Responsive Design

CSS media queries use the @media rule followed by two optional values: “only” or “not”

“only” screens out older browsers from reading the rest of the query.

“not” negates the result: “not screen” means everything except screen-based media.

A feature: value pair, enclosed by parentheses, comprises the essence of the media query.

Media features that can be assessed are predefined.

Multiple feature: value pairs can be combined with “and”.

|

Responsive Design

Media Query Syntax

CSS Rule Set

```
body {  
    background-color: orange;  
}
```

CSS Rule Set with a Media Query

```
@media only screen and  
(min-width: 480px) { body {  
    background-color:  
    orange;  
}  
}
```



HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1">
6   <title></title>
7 <style type="text/css">
8
9   body {
10     background-color: orange;
11   }
12
13 @media only screen and (max-width: 600px) {
14
15   body {
16     background-color: lightblue;
17   }
18 }
19
20 </style>
21 </head>
22 <body>
23   <p>Resize the browser window. When the width of
this document is 600 pixels or less, the background-
color is "lightblue", otherwise it is "orange".</p>
24 </body>
25 </html>
```

Result

Resize the browser window. When the width of this document is 600 pixels or less, the background-color is "lightblue", otherwise it is "orange".

```
media1.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <title></title>
7 <style type="text/css">
8
9     body {
10    background-color: orange;
11    font-size: 40pt;
12 }
13
14 @media only screen and (max-width: 600px) {
15
16     body {
17    background-color: lightblue;
18 }
19 p{
20     font-size: 20pt;
21 }
22 }
23
24
25
26 </style>
27 </head>
28 <body>
29     <p>Resize the browser window. When the width of this document is 600 pixels or less, the background-color is "lightblue", otherwise it is "orange".</p>
30
31 </body>
32 </html>
```

Common device breakpoints

- 320px – 480px: Mobile devices
- 481px – 768px: iPads, Tablets
- 769px – 1024px: Small screens, laptops
- 1025px – 1200px: Desktops, large screens
- 1201px and more – Extra large screens, TV

- There is no standard list of breakpoints. There are a ton of devices on the market (w/new devices released every year) so we can't and we shouldn't define fixed breakpoints for each of them.

SOME SAY IT IS BETTER TO DEFINE YOUR BREAKPOINTS WITH YOUR INDIVIDUAL CONTENT IN MIND ... RATHER THAN DESIGN FOR SPECIFIC DEVICES THAT MAY CHANGE.



ONE OF THE UTILITIES OF LIBRARIES LIKE BOOTSTRAP IS THE BREAKPOINTS ARE BUILT INTO THE TEMPLATE AND KEPT CURRENT: <https://getbootstrap.com/docs/5.0/layout/breakpoints/>

LAYOUT APPLICATIONS

A simple flexbox example: https://i6.cims.nyu.edu/~mr6465/media_queries/flex-media/

```
index.html
1 <!DOCTYPE html>
2 <html lang="en" >
3 <head>
4   <meta charset="UTF-8">
5   <title>Simple FlexBox Layout</title>
6   <link rel="stylesheet" href=".style.css">
7
8 </head>
9 <body>
10 <!-- partial:index.partial.html -->
11 <header class="header">Header </header>
12 <main class="main">Main
13   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tristique mollis i
14   venenatis, ex quis bibendum sagittis, turpis nunc tristique lectus, ut varius leo lo
15   quis. In hac habitasse platea dictumst. Pellentesque sit amet condimentum arcu, vel c
16   placerat elit, vitae sollicitudin neque. Donec lacinia hendrerit scelerisque. Ut sit
17   Suspendisse efficitur est in rhoncus hendrerit. Quisque ipsum augue, venenatis a mau
18   magna eget, tempus dictum ante. Vestibulum et rutrum augue. Aliquam consequat sceler
19   15 Nunc non malesuada dolor, tincidunt fermentum nisl. Donec semper interdum neque a vive
19   Integer nisl libero, lobortis eget sem vitae, interdum scelerisque tortor. Sed interdum
19   efficitur. Vestibulum elementum porttitor vestibulum. Pellentesque habitant morbi trist
19   eros id risus ullamcorper, nec convallis augue pellentesque. Fusce molestie blandit dap
19   cursus nulla ut dui fermentum sagittis at et felis. Nunc ac lorem bibendum, viverra urn
19   lorem ex auctor purus, vel aliquam tellus velit ac ex.</p>
20   
21
22 </main>
23 <aside class="sidebar">Sidebar</aside>
24 <footer class="footer">Footer</footer>
25 <!-- partial -->
```

```
style.css
1 body {
2   display: flex;
3   flex-wrap: wrap;
4 }
5 body > * { /*(greater than) > is a child selector that styles the immediate children*/
6
7   background: #eee;
8   padding: 2rem;
9   text-align: center;
10  border: 5px solid white;
11 }
12
13 .header{
14   flex: 1 100%;
15   order: 4;
16 }
17 .footer {
18   flex: 1 100%;
19   order: 2;
20 }
21 .sidebar {
22   flex: 1;
23   order: 3;
24 }
25 .main {
26   flex: 2;
27   order: 1;
28 }
29
30 @media screen and (max-width: 600px){
31   body {
32     -webkit-flex-direction: column;
33     flex-direction: column;
34     flex-wrap: nowrap;
35   }
36   .header{
37     order: 1;
38   }
39   .footer {
40     order: 4;
41   }
42   .sidebar {
43     order: 3;
44   }
45   .main {
46     order: 2;
47   }
48 }
```

USING BREAKPOINTS WITH GRID ENABLES RESPONSIVE MULTI-COLUMN LAYOUTS

```
1 .wrapper {  
2   display: grid;  
3   grid-template-columns: 50% 30% 20%;  
4   grid-template-rows: 30% 30% 30%;  
5   grid-gap: 1%;  
6  
7   text-align: center;  
8   color:white;  
9   font-family: Arial,Helvetica,sans-serif;  
10  font-size:1.5000em;  
11  
12 }  
13 }
```

Default liquid layout with 3x3 grid

```
128  
129 @media (max-width: 600px) {  
130  
131   .wrapper {  
132     grid-template-columns: 1fr 1fr;  
133   }  
134  
135   #item3{  
136     background-color: pink;  
137     grid-row-start: 1;  
138     grid-row-end: 2;  
139     font-size: 10pt;  
140 }
```

Smaller screen width: switch to 2 column grid!

Examples:

- https://i6.cims.nyu.edu/~mr6465/media_queries/kpop/index_breakpoints.html
- https://i6.cims.nyu.edu/~mr6465/web_design/responsive/seasons/
- https://i6.cims.nyu.edu/~mr6465/media_queries/grid/

Introduction to Web Design

Web Forms

A screenshot of a web form interface. The form is contained within a light gray rounded rectangle. It features three text input fields, each with a black label above it: "Name", "Email", and "Website". Below these input fields is a single "Submit" button. All input fields have a thin yellow border.

Name	<input type="text"/>
Email	<input type="text"/>
Website	<input type="text"/>
<input type="button" value="Submit"/>	

Introduction to Web Design

Forms

Web Forms

Web pages are good not just for providing information to visitors, but also gathering information from them.

The HTML <form> element is used to define a form for getting user input.

A variety of form elements are used to provide an interface for the input.

These form elements include text fields, checkboxes, drop-down menus, and buttons.

HTML Form Element Example

HTML

```
<form action="page.php" method="GET">
  <label for="textbox">Text Box</label><br>
  <input type="text" name="textbox"><br><br>
  <label for="password">Password Input</label><br>
  <input type="password" name="password"><br><br>
  <label for="textbox">Text Area</label><br>
  <textarea name="textarea"></textarea><br><br>
  <label for="dropdown">Dropdown</label><br>
  <select id="dropdown">
    <option value="1">Option 1</option>
    <option value="2">Option 2</option>
    <option value="3">Option 3</option>
  </select><br><br>
  <label for="checkbox">Checkbox</label><br>
  <input type="checkbox" name="checkbox"><br><br>
  <label for="radio">Radio Select</label><br>
  <input type="radio" name="radio">
  <input type="radio" name="radio">
  <input type="radio" name="radio"><br><br>
  <label for="file">File</label><br>
  <input type="file" name="file"><br><br>
  <input type="submit" value="Submit Button">
```

Result

Text Box

Password Input

Text Area

Dropdown
Option 1 ▾

Checkbox

Radio Select

File
 Choose File No file chosen

https://i6.cims.nyu.edu/~mr6465/web_design/javascript/form/form.html

Introduction to Web Design

HTML Form

Web Forms

Forms always begin with the `<form>` element.

The `<form>` element's `action` attribute specifies how the form will be processed.

The `<input>` element is used for various kinds of user input.

The `<input>` element's `type` attribute determines what kind of input is received from users.

Each `<input>` element must also have a `name` attribute and value in order for the data to be sent.

Form elements can be used to collect data

SyncFiddle - Collaborate HTML X +

← → C syncfiddle.net/fiddle/-MpE0mAOdMEpHP4Y4k0I

SyncFiddle Run Add library Export Lint Off Share

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h2>The input Element</h2>
5 <input type="text" placeholder="Type something..." id="myInput">
6 <button type="button" onclick="getInputValue();">Get
7 Value</button>
8 <br>
9 <div id="display">
10 </div>
11 <script type="text/javascript">
12     function getInputValue(){
13         // Selecting the input element and get its value
14         var inputVal =
15             document.getElementById("myInput").value;
16         //Displaying the Value on the page
17         document.getElementById("display").innerHTML = "<br>" +
18             inputVal ;
19     }
20 </script>
21 </body>
22 </html>
```

Result

The input Element

this is something

this is something

<https://syncfiddle.net/fiddle/-MpE0mAOdMEpHP4Y4k0I>

Front-end processing

SyncFiddle - Collaborate HTML

syncfiddle.net/fiddle/-MpE5yJROeqfVgYkMtaR

SyncFiddle

Run Add library Export Lint Off Share

HTML

```
<!DOCTYPE html>
<html>
<head>
<title>seasons</title>
</head>
<body>
<input type="text" placeholder="What season..." id="myInput">
<button type="button" onclick="getInputValue();">Get Value</button>
<div id="display">
</div>
<br>
<div id="seasonal">
</div>
<br>
<script>
function getInputValue(){
    // Selecting the input element and get its value
    var inputVal = document.getElementById("myInput").value;
    //Displaying the Value on the page
    document.getElementById("display").innerHTML = "<br>" +
inputVal ;
}
var season = inputVal;
if (season === 'spring') {
console.log('It\'s spring! The trees are budding!');
document.getElementById("seasonal").innerHTML= "<img
src='https://i6.cms.nyu.edu/~mr6465/web_design/javascript/seasons_js/image
s/spring-2x.jpg'>";
}
18:33:30.576 | It's sunny and warm because it's summer!
```

Result

summer Get Value

summer



Console Clear

<https://syncfiddle.net/fiddle/-MpE5yJROeqfVgYkMtaR>

Introduction to Web Design

Web Forms

Form Validation

Before form data gets sent, it's important to validate the input.

- You may want to make certain form fields required.
- You probably want to make sure that certain fields are completed properly.
- You should also verify that malicious code is not sent along with form input.

Form validation can be done client-side, server-side, or both.

Save form Data in a Text File u x +

i6.cims.nyu.edu/~mr6465/web_design/javascript/form/data_parse.html

Enter your name

Enter your age

Enter your email address

-- Choose the country -- ▾

Write some message ...

https://i6.cims.nyu.edu/~mr6465/web_design/javascript/form/data_parse.html

Often this data is sent to a php file on the server for processing via a http get or post requests (security and functionality)

Introduction to Web Design

Web Forms

HTML Form

```
<form action="my-script.php">  
  
    First name:  
    <input type="text" name="firstname">  
  
    Last name:  
    <input type="text" name="lastname">  
  
    <input type="submit" value="Submit">  
  
</form>
```

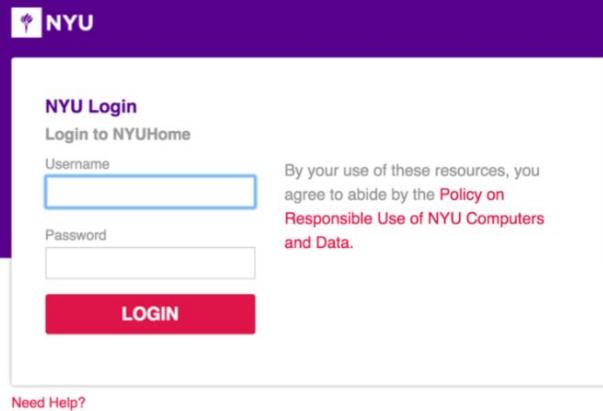
Introduction to Web Design

Web Forms



Introduction to Web Design

Web Forms



The image shows a screenshot of the NYU Login page. At the top left is the NYU logo. Below it, the text "NYU Login" and "Login to NYUHome". There are two input fields: "Username" and "Password", both outlined in blue. To the right of these fields is a red block of text: "By your use of these resources, you agree to abide by the [Policy on Responsible Use of NYU Computers and Data](#)." At the bottom center is a red "LOGIN" button. At the very bottom, there is a link "Need Help?".

NYU

NYU Login
Login to NYUHome

Username

Password

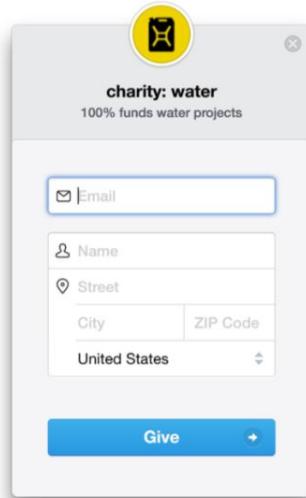
By your use of these resources, you agree to abide by the [Policy on Responsible Use of NYU Computers and Data](#).

LOGIN

Need Help?

Introduction to Web Design

Web Forms



A screenshot of a modal window for the charity: water website. The window has a yellow circular logo at the top left. Below it, the text "charity: water" and "100% funds water projects". The main area contains a form with fields for "Email" (with a placeholder icon), "Name" (placeholder icon), "Street" (placeholder icon), "City" and "ZIP Code" (separated by a vertical line), and a dropdown menu set to "United States". At the bottom is a large blue "Give" button with a white arrow icon.

charity: water
100% funds water projects

Email

Name

Street

City ZIP Code

United States

Give →

Introduction to Web Design

Form Processing

Web Forms

Normally, forms are sent to the server to be processed.

This requires a server-side application written in a back-end language.

Since server-side coding is beyond the scope of this class, we will use a free service that receives form data and sends it to you via email.

A screenshot of a web browser window showing the Formspree website. The title bar reads "Custom Forms with No Server" and the address bar shows "formspree.io". The page features a red header with the Formspree logo and navigation links for Home, Plans, Blog, Library, and Help. On the right, there are "Get started" and "Sign in" buttons. The main content area has a large heading "The form solution for any developer" and a subtext "Use your own frontend code. Submit to our API. We'll handle the rest." Below this is a red "Get started" button. A dark callout box contains sample HTML code for a form submission. At the bottom, a message states "Formspree forms can perfectly match your website by inheriting your website's CSS".

The form solution for any developer

Use your own frontend code. Submit to our API. We'll handle the rest.

[Get started](#)

```
<form action="https://formspree.io/f/{form_id}" method="post">
  <label for="email">Your Email</label>
  <input name="Email" id="email" type="email">
  <button type="submit">Submit</button>
</form>
```

Formspree forms can perfectly match your website
by inheriting your website's CSS

Web Form

view-source:https://i6.cims.nyu.edu/~mr6465/web_design/javascript/form/

First Name:

Last Name:

Email:

Message

Enter your message here.

https://i6.cims.nyu.edu/~mr6465/web_design/javascript/form/

FORM ART COMPETITION



who: Alexei Shulgin

what: Form

where: c3

browser: Netscape 3.0

Contact

THANKS TO

Introduction to Web Design

Web Audio and Video



Introduction to Web Design

Sound

Web Audio and Video

Sound consists of pressure waves moving through air.

Without air, there is no sound.

Our ears are sensitive to pressure waves and transmit these signals to the brain.

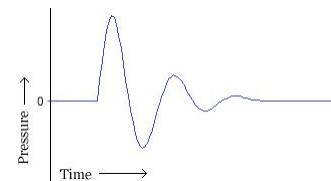


https://www.youtube.com/watch?v=6FI5Z_aw3Fc

Introduction to Web Design

Hand Clap

Web Audio and Video

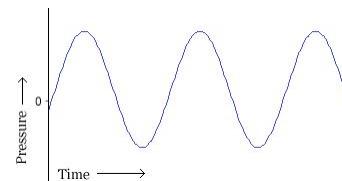


<https://freesound.org/people/mikaelfernstrom/sounds/68698/>

Introduction to Web Design

Periodic Wave

Web Audio and Video



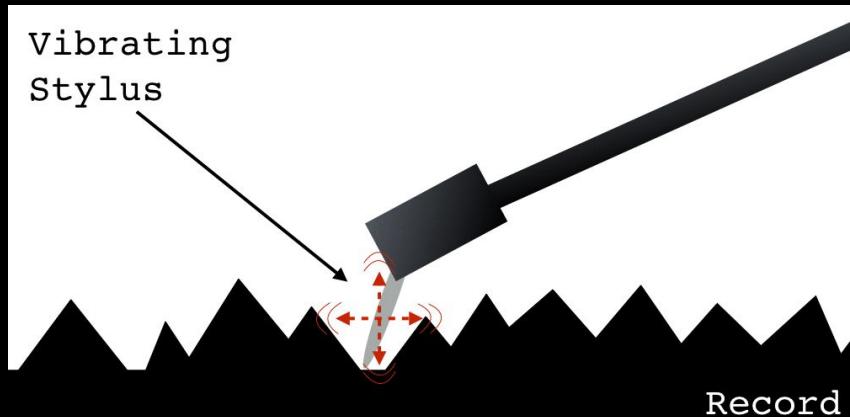
<https://freesound.org/people/Jovica/packs/489/>

The Sound of the Internet

<https://freesound.org/people/lintphishx/sounds/65786/>

Introduction to Web Design

Sound Recording



Web Audio and Video

When sound is recorded, acoustic waves are converted to electrical waves.

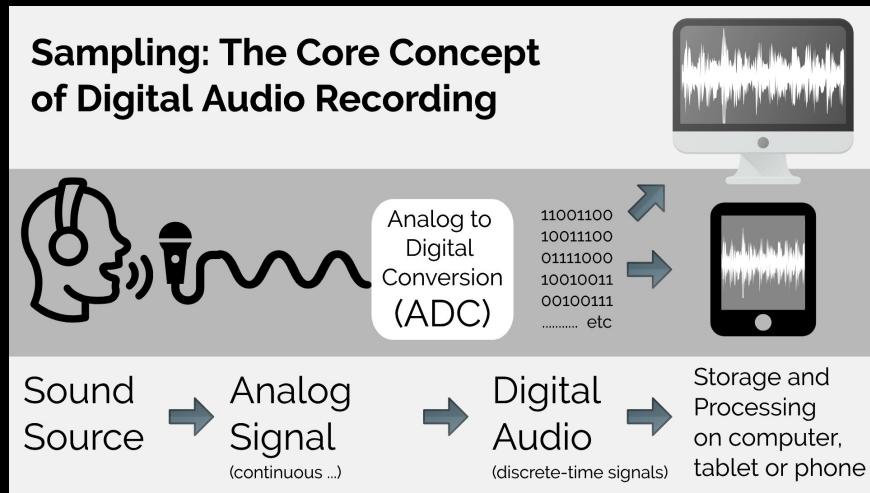
A microphone consists of a small membrane that vibrates.

Movements of the membrane are translated into electrical signals.

Higher pressure typically corresponds to higher voltage.

Introduction to Web Design

Digital Audio



Web Audio and Video

An audio signal is an analog (continuous) format.

The electrical waves must be converted to digital information for computational processing.

Digital recording is accomplished with an analog-to-digital converter (ADC).

The ADC captures a snapshot of the electric voltage on an audio line and represents it as a digital number.

Capturing the voltage thousands of times per second creates a good approximation of the original audio.

Introduction to Web Design

Digital Audio Playback

Web Audio and Video

All computers must give us analog signals to be useful.

The screen converts digital information to light.

The digital-to-analog converter (DAC) takes the sample and sets a certain voltage on the analog outputs to recreate the signal.

This voltage is conveyed to the speakers which create pressure waves in the air.

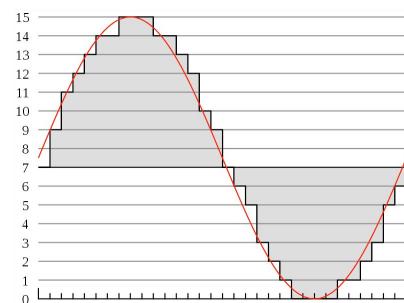
Introduction to Web Design

Digital/Analog Sound

<https://www.bhphotovideo.com/c/buy/AD-DA-Converters/ci/14831>

<https://www.audioadvice.com/videos-reviews/what-is-a-dac/>

Web Audio and Video



<https://www.avid.com/pro-tools>

<https://www.apple.com/mac/garageband/>

<https://www.audacityteam.org/>

Introduction to Web Design

Digital Video

Web Audio and Video

Analog video uses a continuous electrical signal to capture footage on magnetic tape. Examples: VHS or Hi-8.

In theory, analogue video can capture more information leading than digital video but dv is now so widely implemented it would be rare to find analogue video still being used.

Digital Video is encoded in binary information and can be reproduced infinitely with no quality loss.

mp4 , mov , avi , mov , .wmv are some formats found on the web

VIDEO CODECS + FORMATS FOR THE WEB

MP4 MOV WMV H264

RECOMMENDED: MP4 using H264 Codec

Can also use quicktime for .mov if an alpha channel is needed

<https://www.freecodecamp.org/news/video-formats-for-the-web/>

Free video compression and conversion

<https://handbrake.fr/>

Rendering: <https://www.adobe.com/products/media-encoder.html>

Introduction to Web Design

HTML Audio and Video

Web Audio and Video

HTML5 supports audio and video natively in the web browser.

For years, it was necessary to rely on a third party to deliver this kind of content.

Now we can use the `<audio>` and `<video>` tags.

The `<audio>` and `<video>` tags use `src` attribute or the `<source>` tag to specify one or more media resources.

https://i6.cims.nyu.edu/~mr6465/web_design/media/

https://i6.cims.nyu.edu/~mr6465/web_design/media/video-player.html

Introduction to Web Design

HTML Inline Frames

Web Audio and Video

Another way to embed media on a web page is with the HTML inline frame element: <iframe>

An inline frame represents a nested browsing context, embedding another HTML page into the current one.

Embedding all or part of one web page into another is way to present content on a website.

<https://syncfiddle.net/fiddle/-MpYt1hUaWa1XL-mUp6X>

Introduction to Web Design

JavaScript





HTML

CSS

JavaScript

Introduction to Web Design

JavaScript

JavaScript

You can think of a web page as consisting of three layers: structure, presentation, and behavior.

- HTML is the structure layer.
- CSS is the presentation layer.
- JavaScript is the behavior layer.

JavaScript is a programming language for creating interactivity and functionality in web browsers.

hello_world!

Increment Red Green

```
<body>  
<p id="myP">hello_world!</p>  
  
<button type="button" id="increment">Increment</button>  
<button type="button" id="red">Red</button>  
<button type="button" id="green">Green</button>  
  
<div id="output"></div>
```

```
<style type="text/css">  
  #myP{  
    color: orange;  
    font-size: 150px;  
  }  
  
  #myP:hover{  
    color: pink;  
    font-size: 50px;  
}  
</style>
```

```
<script>  
  
let a;  
a = 75;  
let r;  
r = 0;  
let b;  
b = 0;  
  
let increment = document.getElementById("increment");  
let red = document.getElementById("red");  
let green = document.getElementById("green");  
  
increment.addEventListener("click", function(){  
  
  a = a +1;  
  document.getElementById("myP").style.fontSize = a+"px";  
  console.log(a);  
  document.getElementById("output").innerHTML = a;  
  
});  
  
red.addEventListener("click", function(){  
  
  r = r+10;  
  document.getElementById("myP").style.color = "rgb"+("("+r+",0,0)");  
});
```

Introduction to Web Design

Background

JavaScript

JavaScript was invented by Brendan Eich and introduced by Netscape in 1995.

At that time, the Java language was ascendant and the name "JavaScript" was an attempt to ride this popularity.

Eventually, browsers other than Netscape began to support JavaScript functionality, calling it "ECMAScript."

Today, JavaScript is not only a lingua franca of the web but a basis for many other computational media projects.



ES6 is the latest version of JavaScript (ECMA 2015)

ECMAScript 2015 was the second major revision to JavaScript.

ECMAScript 2015 is also known as ES6 and ECMAScript 6.

Some of the major updates included:

- [The let keyword](#)
- [The const keyword](#)
- [Arrow Functions](#)
- [Promises](#)

Introduction to Web Design

Front-End Language

JavaScript

Like HTML and CSS, JavaScript is usually rendered in the web browser.

Because it's rendered in the browser rather than on a server, JavaScript is considered a "front-end" language.

A browser's "JavaScript engine" interprets and executes JavaScript code in the browser.

There are different JavaScript engines for different browsers.

Introduction to Web Design

Capability

JavaScript

Computationally speaking, there isn't much JavaScript can't do; it's a robust programming language.

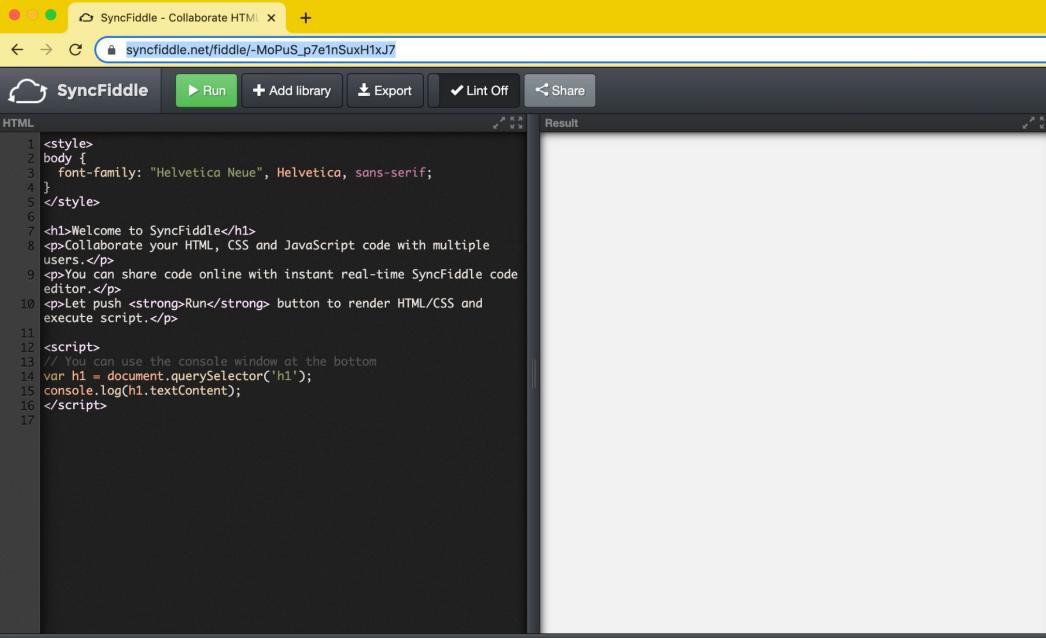
Core functionality includes modifying HTML and CSS, communicating with the server, and storing data.

We will use JavaScript to modify page content and style.

As with any technology, it's good to consider when to use it and when it's unnecessary.

Let's say "Hello World" with JS together!
Using (console.log, alert, + .innerHTML methods) and then style our innerHTML output!

https://syncfiddle.net/fiddle/-MoPuS_p7e1nSuxH1xJ7



The screenshot shows the SyncFiddle web application interface. At the top, there is a yellow header bar with three dots, a title 'SyncFiddle - Collaborate HTML...', and a '+' button. Below the header is a toolbar with a cloud icon, the 'SyncFiddle' logo, a 'Run' button, an 'Add library' button, an 'Export' button, a 'Lint Off' button, and a 'Share' button. The main area is divided into two panes: 'HTML' on the left and 'Result' on the right. The 'HTML' pane contains the following code:

```
1 <style>
2 body {
3   font-family: "Helvetica Neue", Helvetica, sans-serif;
4 }
5 </style>
6
7 <h1>Welcome to SyncFiddle</h1>
8 <p>Collaborate your HTML, CSS and JavaScript code with multiple
users.</p>
9 <p>You can share code online with instant real-time SyncFiddle code
editor.</p>
10 <p>Let push <strong>Run</strong> button to render HTML/CSS and
execute script.</p>
11
12 <script>
13 // You can use the console window at the bottom
14 var h1 = document.querySelector('h1');
15 console.log(h1.textContent);
16 </script>
17
```

The 'Result' pane is currently empty, showing a plain white space.

Introduction to Web Design

Application

JavaScript

As with CSS, JavaScript targets HTML elements to do something with them.

There are three ways you can apply JavaScript to HTML.

- Inline JavaScript
- Embedded JavaScript
- External JavaScript

External and embedded JavaScript are preferable for their separation of content and behavior.

JavaScript

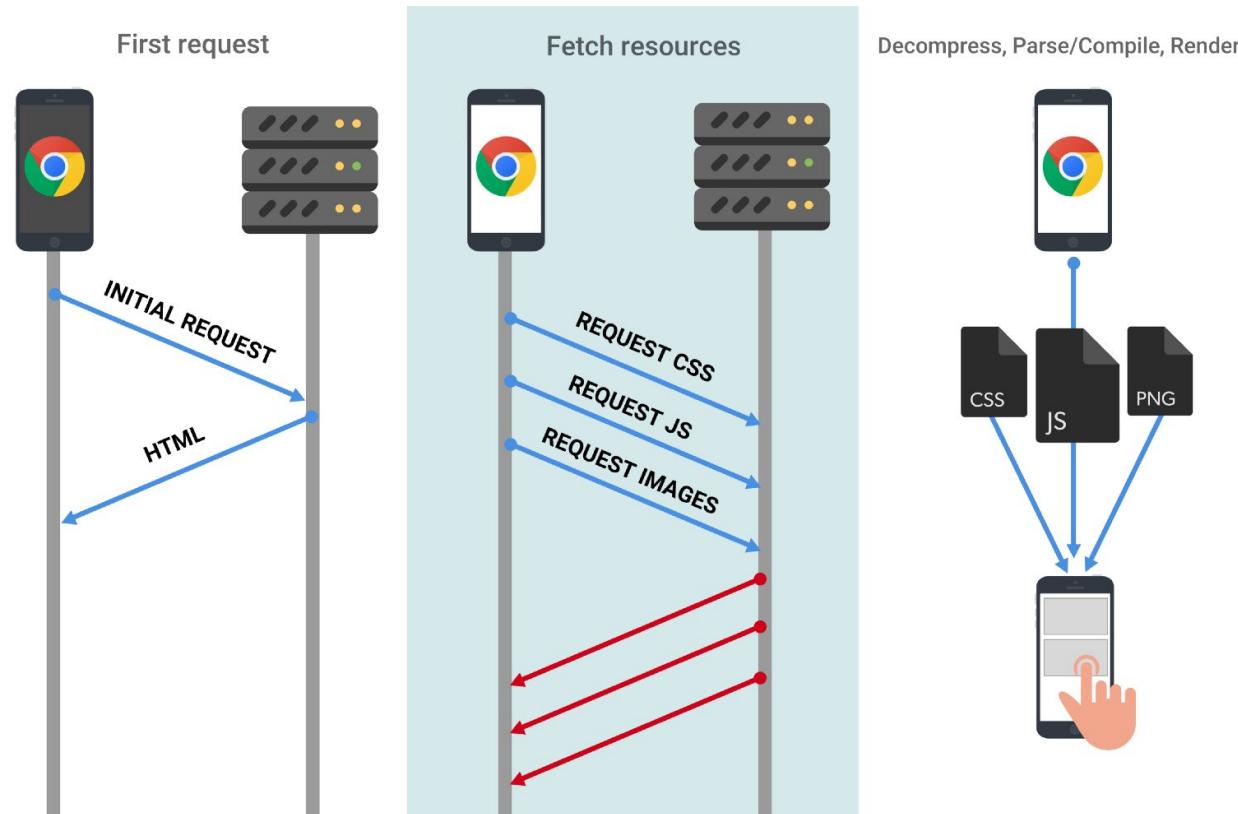
The native language of modern-web browsers

A large, bold, orange logo consisting of the letters "JS" in a sans-serif font.

JavaScript ('dʒɑ:və,skrɪpt),[8] often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a language that is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web.[9] JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it,[10] and all major web browsers have a dedicated JavaScript engine to execute it.

JavaScript is executed by the Browser



JavaScript Console

The console is a panel that displays important messages, like errors, for developers. Much of the work the computer does with our code is invisible to us by default. If we want to see things appear on our screen, we can print, or log, to our console directly.

It's very useful for us to be able to print values to the console, so we can see the work that we're doing and debug.



Chrome File Edit

View History Bookmarks Window Help

Always Show Bookmarks Bar ⌘⌘B

Stop ⌘.

Reload This Page ⌘R

Enter Presentation Mode ⌘⌘F

Enter Full Screen ⌘⌘F

Actual Size ⌘0

Zoom In ⌘+

Zoom Out ⌘-

Encoding ▶

Developer ▶

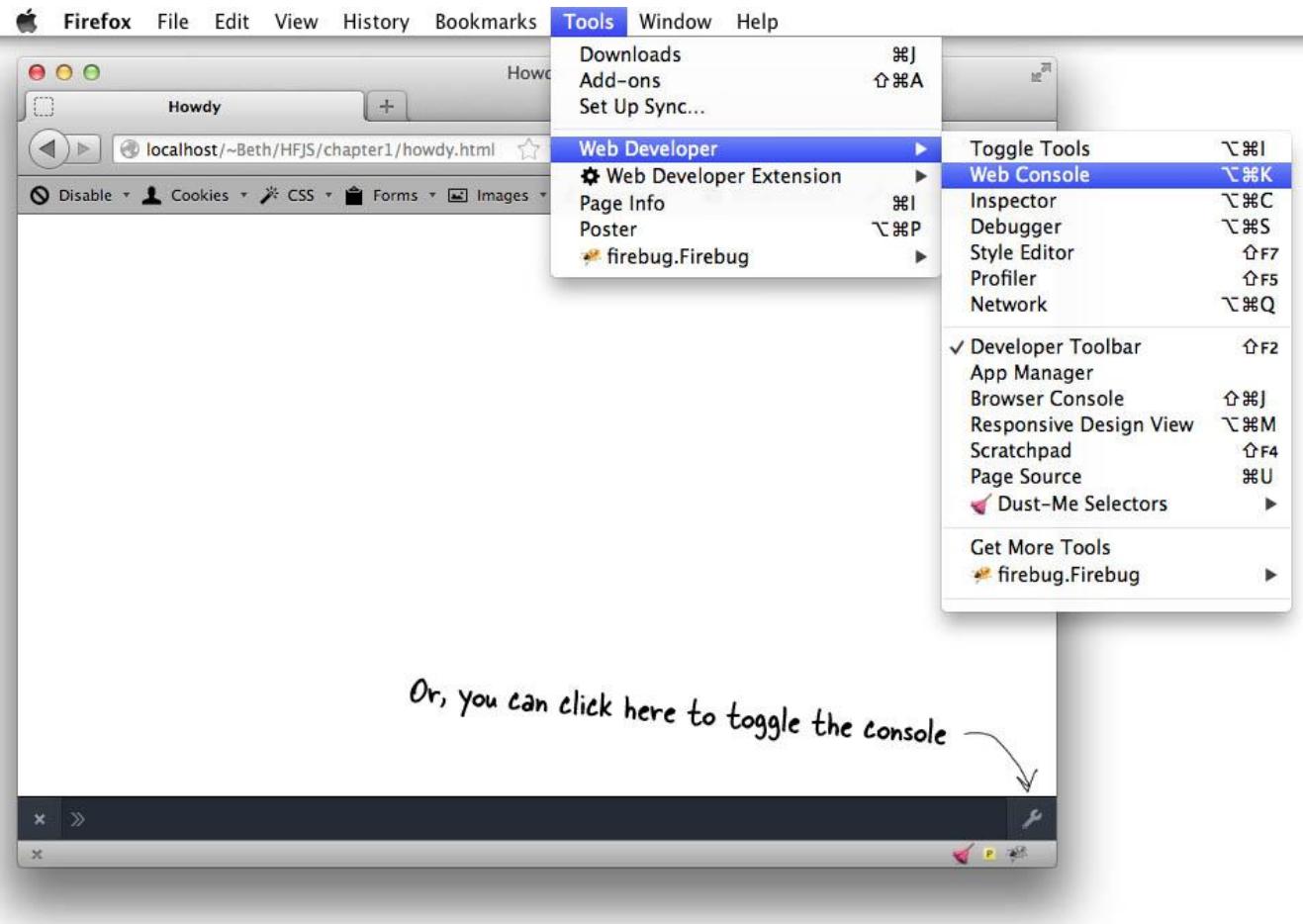
⋮

★ » ⌄

View Source ⌘⌘U

Developer Tools ⌘⌘I

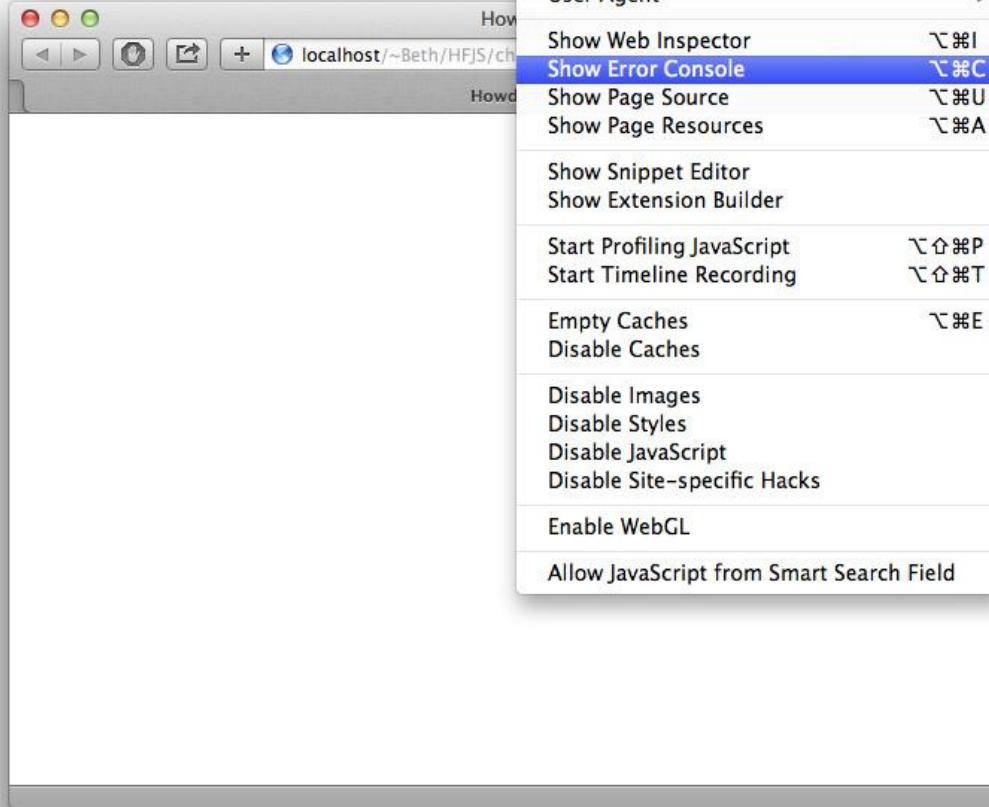
JavaScript Console ⌘⌘J



Safari File Edit View History Bookmarks

Develop Window Help

- Open Page With ▶
- User Agent ▶
- Show Web Inspector ⌘I
- Show Error Console ⌘C
- Show Page Source ⌘U
- Show Page Resources ⌘A
- Show Snippet Editor
- Show Extension Builder
- Start Profiling JavaScript ⌘P
- Start Timeline Recording ⌘T
- Empty Caches ⌘E
- Disable Caches
- Disable Images
- Disable Styles
- Disable JavaScript
- Disable Site-specific Hacks
- Enable WebGL
- Allow JavaScript from Smart Search Field



Printing to the console

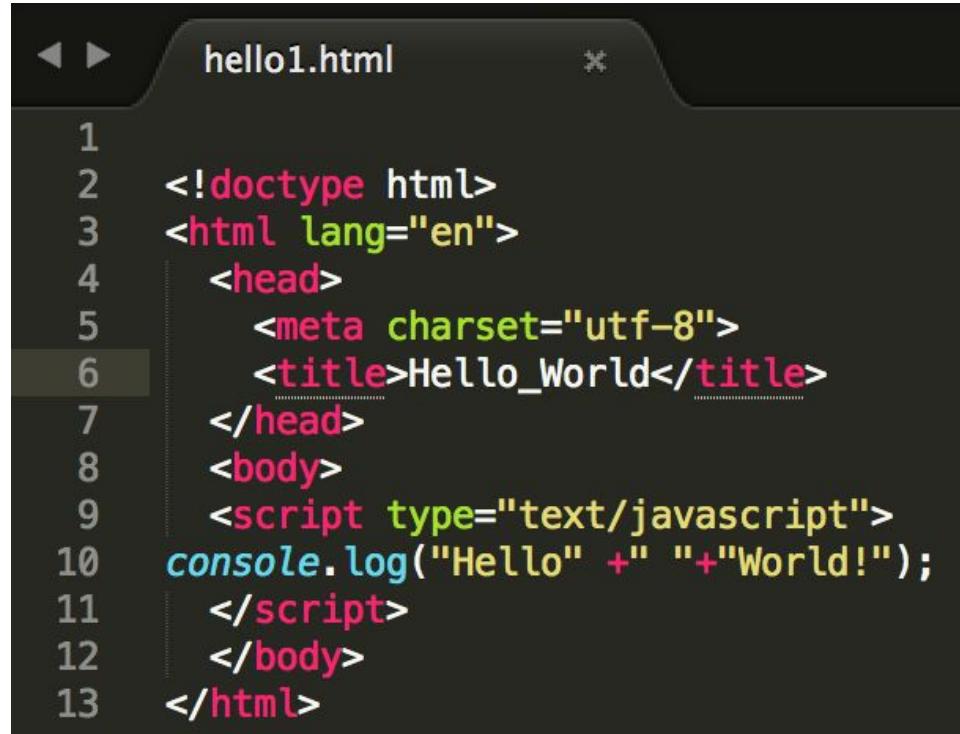
The Console method `log()` outputs a message to the web console. The message may be a single string (with optional substitution values), or it may be any one or more JavaScript objects.

Syntax:

```
console.log(obj1 [, obj2, ..., objN]);  
console.log(msg [, subst1, ..., substN]);
```

<https://developer.mozilla.org/en-US/docs/Web/API/Console/log>

Let's write the following code and preview it in Chrome:



```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Hello_World</title>
6   </head>
7   <body>
8     <script type="text/javascript">
9       console.log("Hello" +" "+"World!");
10    </script>
11  </body>
12</html>
```

*Comments

In JavaScript (like other C-based languages) comments are designated using the double backslash:

Comments in both PHP and JavaScript

// Single Line Comment

```
/*
    Multi-line
    comment
*/
```

Available in PHP Only

PHP single line comment

Data Types

JS uses 7 fundamental data types

1. *Number*: Any number, including numbers with decimals: 4, 8, 1516, 23.42.
2. *String*: Any grouping of characters on your keyboard (letters, numbers, spaces, symbols, etc.) surrounded by single quotes: ' ... ' or double quotes " ... ". Though we prefer single quotes. Some people like to think of string as a fancy word for text.
3. *Boolean*: This data type only has two possible values— either true or false (without quotes). It's helpful to think of booleans as on and off switches or as the answers to a "yes" or "no" question.
4. *Null*: This data type represents the intentional absence of a value, and is represented by the keyword null (without quotes).
5. *Undefined*: This data type is denoted by the keyword undefined (without quotes). It also represents the absence of a value though it has a different use than null.
6. *Symbol*: A newer feature to the language, symbols are unique identifiers, useful in more complex coding. No need to worry about these for now.
7. *Object*: Collections of related data.

Data Primitives

The first 6 of those types are considered primitive data types. They are the most basic data types in the language. Objects are more complicated and we'll look at them as we progress.

Two of the most common data types are Strings and Numbers.

Strings are expressed in quotes and have no numerical value.

Numbers are expressed without quotes and hold a mathematical value.

Try inputting the following into your code

```
1 console.log("JavaScript");
2 console.log(2011);
```

The first statement outputs a string. The second, a number.

Class Exercise - Strings & Numbers

Modify your previous hello_world code. Save as a new file and add the following:

- 1) Output a string to the console
- 2) Output an integer to the console
- 3) Output the string: `Woohoo! I love to code! #javascript` to the console.
- 4) Output a float value to the console

Arithmetic Operators

Operator Statements

An operator is a character that performs a task in our code. JavaScript has several built-in arithmetic operators, that allow us to perform mathematical calculations on numbers. These include the following operators and their corresponding symbols:

1. Add: +
2. Subtract: -
3. Multiply: *
4. Divide: /
5. Remainder: %

```
console.log(3 + 4); // Prints 7  
console.log(5 - 1); // Prints 4  
console.log(4 * 2); // Prints 8  
console.log(9 / 3); // Prints 3
```

Your Turn

To your code, add the following console statements:

1. Inside of a `console.log()`, add `3.5` to your age. This is the age you'll be when we start sending people to live on Mars.
2. On a new line write another `console.log()`. Inside the parentheses, take the current year and subtract `1969`.

The answer is how many years it's been since the 1969 moon landing.

3. Create another `console.log()`. Inside the parentheses divide `65` by `240`.
4. Create one last `console.log()`. Inside the parentheses, multiply `0.2708` by `100`.

That's the percent of the sun that is made up of helium. Assuming we could stand on the sun, we'd all sound like chipmunks!

String Concatenation

Operators aren't just for numbers! When a + operator is used on two strings, it appends the right string to the left string:

```
console.log('hi' + 'ya'); // Prints  
'hiya'  
console.log('wo' + 'ah'); // Prints  
'woah'  
console.log('I love to ' + 'code.')  
// Prints 'I love to code.'
```

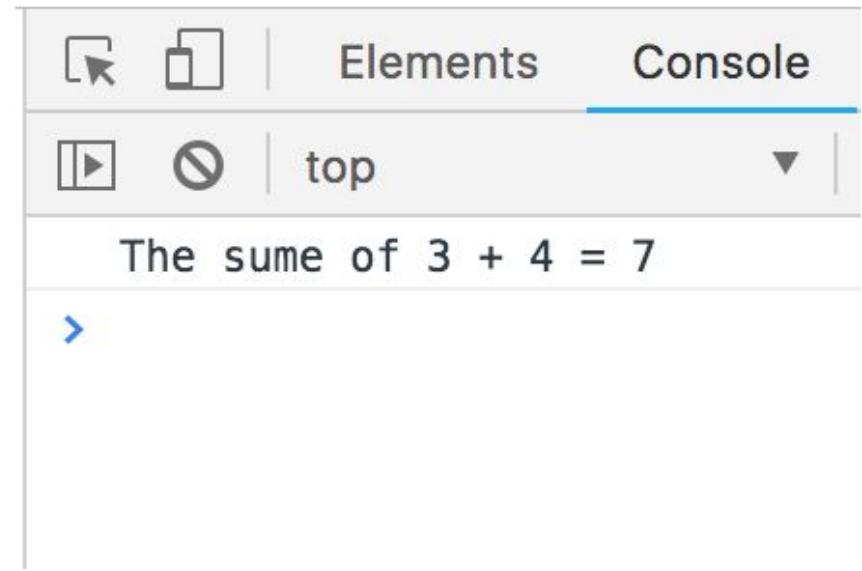
```
console.log('front ' + 'space');  
// Prints 'front space'  
console.log('back' + ' space');  
// Prints 'back space'  
console.log('no' + 'space');  
// Prints 'nospace'  
console.log('middle' + ' ' + 'space');  
// Prints 'middle space'
```

This process of appending one string to another is called concatenation. Notice in the third example we had to make sure to include a space at the end of the first string. The computer will join the strings exactly, so we needed to make sure to include the space we wanted between the two strings.

Mixing Numbers + Strings

We can concatenate output by paying attention to expressing strings + numbers correctly:

```
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8">
5      <title>Hello_World</title>
6  </head>
7  <body>
8      <script type="text/javascript">
9
10     //prints the string + integer result to the console
11     console.log("The sume of 3 + 4 = " + (3 + 4));
12
13     </script>
14  </body>
15 </html>
```



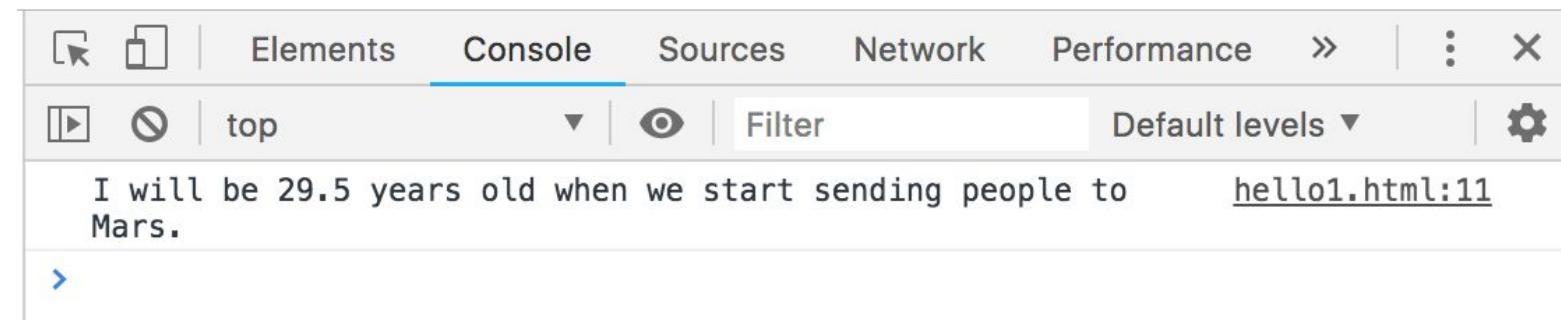
Class Exercise

Rewrite your arithmetic operators code to include the textual context.

For example, the console output to question #1 should read:

I will be 29.5 years old when we start sending people to Mars.

```
10 //prints the string + integer result to the console  
11 console.log("I will be " + (26 + 3.5) +" years old when we start sending people to Mars.");  
12
```



Properties

The “.” operator

When you introduce a new piece of data into a JavaScript program, the browser saves it as an instance of the data type.

Every data type has certain properties associated with it.

Every string instance has a property called `length` that stores the number of characters in that string. You can retrieve property information by appending the string with a period and the property name:

```
console.log('Hello'.length); // Prints 5
```

Methods

Remember that methods are actions we can perform. JavaScript provides a number of string methods.

We call, or use, these methods by appending an instance with a period (the dot operator), the name of the method, and opening and closing parentheses: ie. 'example string'.methodName().

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/prototype

Add the following to your code

```
1 console.log('JavaScript'.toUpperCase());  
2 console.log('      Remove whitespace  
  '.trim());
```

Built-in objects

In addition to console, there are other objects built into JavaScript. Down the line, you'll build your own objects, but for now these "built-in" objects are full of useful functionality.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

Practice with global objects

For example, if you wanted to perform more complex mathematical operations than arithmetic, JavaScript has the built-in Math object.

The great thing about objects is that they have methods! Let's call the `.random()` method from the built-in Math object:

```
console.log(Math.random()); // Prints a  
random number between 0 and 1
```

Arithmetic operators with objects

To generate a random number between 0 and 50, we could multiply this result by 50, like so:

```
Math.random() * 50;
```

The example above will likely evaluate to a decimal. To ensure the answer is a whole number, we can take advantage of another useful `Math` method called `Math.floor()`.

`Math.floor()` takes a decimal number, and rounds down to the nearest whole number. You can use `Math.floor()` to round down a random number like this:

```
Math.floor(Math.random() * 50);
```

Your turn

- 1) Inside of a `console.log()`, create a random number with `Math.random()`, then multiply it by `100`.
- 2) Now, use `Math.floor()` to make the output a whole number. Inside the `console.log` you wrote in the last step, put `Math.random() * 100` inside the parentheses of `Math.floor()`.
- 3) Find a method on the [JavaScript Math object](#) that returns the smallest integer greater than or equal to a decimal number. Use this method with the number `you generated`. Log the answer to the console.
- 4) Use the [JavaScript documentation](#) to find a method on the built-in `Number` object that checks if a number is an integer.

Put the number `you generated from #3` in the parentheses of the method and use `console.log()` to print the result.

```
1 console.log(Math.floor(Math.random()  
*100));  
2 console.log(Math.ceil(43.8));  
3 console.log(Number.isInteger(2017)); //  
Prints false
```

Output

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using `innerHTML`.
- Writing into the HTML output using `document.write()`.
- Writing into an alert box, using `window.alert()`.
- Writing into the browser console, using `console.log()`.

Using innerHTML

To access an HTML element, JavaScript can use the `document.getElementById(id)` method:

<https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById>

The `id` attribute defines the HTML element. The `innerHTML` property defines the HTML content:

```
document.getElementById("id_name").innerHTML = "your output";
```

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <style type="text/css">
5          #demo{
6              background-color: yellow;
7          }
8      </style>
9  </head>
10 <body>
11
12 <h1>This is a heading written with html</h1>
13 <p>This is a paragraph written with html</p>
14
15 <p id="demo"></p>
16
17 <script>
18 document.getElementById("demo").innerHTML = 5 + 6;
19 </script>
20
21 </body>
22 </html>
```

Try this code on your own.

Then, create another div, assign it an id, and add your own string message to display within that div using the innerHTML method with JavaScript.

Try it with an image:

.innerHTML= ;

window.alert()

You can use an alert box to display data:

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h2>Web Page</h2>
6  <p>paragraph.</p>
7
8  <script>
9  window.alert(5 + 6);
10 </script>
11
12 </body>
13 </html>
```

Try this code and then add a second window alert displaying a string message.

document.write()

The `document.write()` method writes a string of text to a document stream. For testing purposes, it is convenient to use `document.write()` but please keep in mind this function deletes existing html after the document is loaded.

We will practice a bit with it, but its real utility beyond testing & learning lies with **AJAX (Asynchronous Javascript and XML)** programming. A technique for dynamic web programming we will look at later in class.

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1>Web Page</h1>
6  <p>Paragraph.</p>
7
8  <script>
9  document.write(5 + 6);
10 document.write("Hello World");
11 </script>
12
13 </body>
14 </html>
```

Test this code out.

Use `document.write` to output the results of your own arithmetic operation and then a string message.

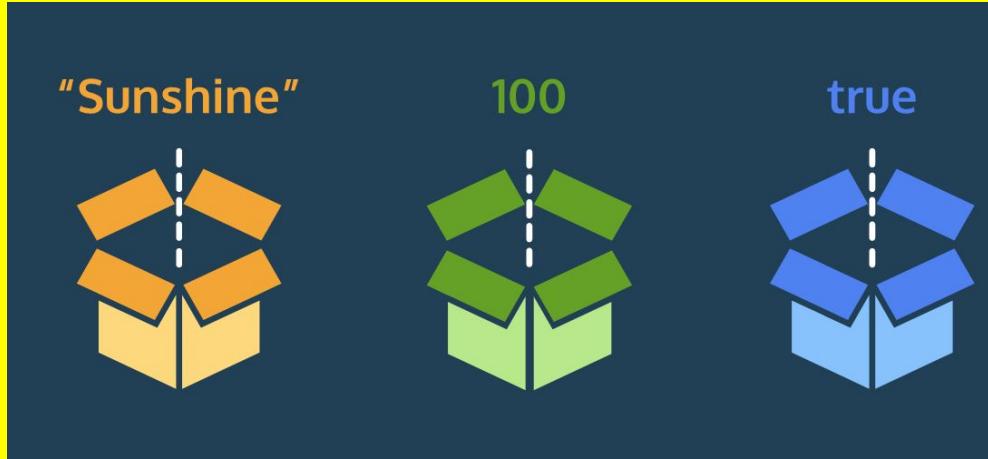
LAST CLASS 4 EXERCISE

- 1) Take your console output from the exercises on slide 15 and use the inner.HTML method to output each answer into it's own div
- 2) Do the same with your answers from slide 21
- 3) And the same with your answers from slide 29
- 4) Use CSS to position your divs as a horizontal and vertical grid and give each background a different color:

<https://gridbyexample.com/examples/>

JavaScript II

Variables & Conditionals



Variables

Variables - containers for data

In programming, a variable is a container for a value. You can think of variables as little containers for information that live in a computer's memory. Information stored in variables, such as a username, account number, or even personalized greeting can then be found in memory.

In short, variables label and store data in memory.

It is important to distinguish that variables are not values; they contain values and represent them with a name.

Declaring variables

There are three keywords used to declare variables in JavaScript:

- **var** - the most common, standard in JavaScript and other C-based languages
- **let** - new in ES2015 (ES6). Introduces block scope
- **const** - similar to let, but cannot be reassigned. Also introduced in ES6

Syntax

```
var myName = 'Arya';
console.log(myName);
// Output: Arya
```

1. `var`, short for variable, is a JavaScript *keyword* that creates, or *declares*, a new variable.
2. `myName` is the variable's name. Capitalizing in this way is a standard convention in JavaScript called *camel casing*. In camel casing you group words into one, the first word is lowercase, then every word that follows will have its first letter uppercased. (e.g. `camelCaseEverything`).
3. `=` is the *assignment operator*. It assigns the value (`'Arya'`) to the variable (`myName`).
4. `'Arya'` is the *value* assigned (`=`) to the variable `myName`. You can also say that the `myName` variable is *initialized* with a value of `'Arya'`.

Let's Practice

- Create a new .js file called “variables.js” and save it in a directory called “Class_5”
- Declare a variable named `favoriteFood` using the `var` keyword and assign to it the string '`pizza`'.
- Declare a variable named `numOfSlices` using the `var` keyword and assign to it the number `8`.
- use `console.log()` to print the value saved to `favoriteFood`.
- use `console.log()` to print the value saved to `numOfSlices`.

Your Turn!

- Create your own variable and assign it a string value
- console.log the value of this variable
- Create another variable and assign it a float value
- Console.log the value of this new variable

Browser Practice!

Let's use JavaScript variables to change the style properties of text.

Create a file called js1.html and add the following code:

```
1  <html>
2  <body>
3
4  <p id="p2">JavaScript!</p>
5
6  <script>
7  </script>
8
9  </body>
10 </html>
```

Now let's use JavaScript to change the style properties:

```
1 <html>
2 <body>
3
4 <p id="p2">JavaScript!</p>
5
6 <script>
7
8 document.getElementById("p2").style.color = "red";
9 document.getElementById("p2").style.fontSize = "150px";
10
11 </script>
12
13 </body>
14 </html>
```

Instead of hard-coding in the color “red” and size “150px”, let’s make variables called color and size to hold this data:

```
1  <html>
2  <body>
3
4  <p id="p2">JavaScript!</p>
5
6  <script>
7  var color ="red";
8  var size ="150";
9
10 document.getElementById("p2").style.color = color;
11 document.getElementById("p2").style.fontSize = size+"px";
12
13 </script>
14
15 </body>
16 </html>
```

Your turn

What happens when we change the data held by the variables size and color?

Change var color="red"; to var color="green";

Change var size="150"; to var size ="300";

Our text changes style properties!

Now, write your own text in a separate <p> tag and assign it another ID.

Use JavaScript variables to change its color and size.

Variables are dynamic (most of them)

The power and flexibility of variables really lies in their ability to hold dynamic data values. This means, we can change the values they hold for a variety of purposes.

```
1 var meal = 'Enchiladas';
2 console.log(meal); // Output: Enchiladas
3 meal = 'Burrito';
4 console.log(meal); // Output: Burrito|
```

In this example, we initiate the value of the variable meal first as Enchiladas, then we change it to Burrito.

Let's write this code back in our variables.js file to see it in action.

Your turn!

Create another variable called “calories” and assign it an integer value.

Print the value of calories to the console.

After your console.log statement, re-assign the value of calories to a float number

Print out the new value of calories to the console

Browser Practice!

We can extend this concept to our browser exercise in js1.html

```
1  <html>
2  <body>
3
4  <p id="p2">JavaScript!</p>
5  <p id="p3">Variables!</p>
6
7  <script>
8  var color ="red";
9  var size ="150";
10
11 document.getElementById("p2").style.color = color;
12 document.getElementById("p2").style.fontSize = size+"px";
13
14 color ="blue";
15 size ="75";
16
17 document.getElementById("p3").style.color = color;
18 document.getElementById("p3").style.fontSize = size+"px";
19 </script>
20
21 </body>
22 </html>
```

Create another `<p>` element and assign it a unique ID. Add the text “JS is Awesome!” to this paragraph.

Change the value of variables color and size a third time to make the text purple and 10px.

Create a new `<p>` element and write in any text you would like.

Now, change the value of the variables you created on your own in the previous exercises to add another color and size.

Mathematical Assignment Operators

Changing a variable's value

Let's consider how we can use variables and math operators to calculate new values and assign them to a variable. Check out the example below:

```
9  <script type="text/javascript">
10 var w = 4;
11 w = w + 1;
12
13 console.log(w); // Output: 5
```

In the example above, we created the variable `w` with the number `4` assigned to it. The following line, `w = w + 1`, increases the value of `w` from `4` to `5`

Other arithmetic operators

We also have access to other mathematical assignment operators: `-`, `*`, and `/` which work in a similar fashion.

```
15  var x = 20;
16  x = x - 5; // Can be written as x -= 5;
17  console.log(x); // Output: 15
18
19  var y = 50;
20  y = y * 2; // Can be written as y *= 2;
21  console.log(y); // Output: 100
22
23  var z = 8;
24  z /= 2; // Can be written as z = z / 2
25  console.log(z); // Output: 4
```

Let's practice

Start with the following code:

```
9  <script type="text/javascript">
10 var levelUp = 10;
11 var powerLevel = 9001;
12 var multiplyMe = 32;
13 var quarterMe = 1152;
14
15 console.log('The value of levelUp:', levelUp);
16 console.log('The value of powerLevel:', powerLevel);
17 console.log('The value of multiplyMe:', multiplyMe);
18 console.log('The value of quarterMe:', quarterMe);
19 </script>
```

Do the following:

- Use the `+` mathematical assignment operator to increase the value stored in `levelUp` by `5`.
- Use the `-` mathematical assignment operator to decrease the value stored in `powerLevel` by `100`.
- Use the `*` mathematical assignment operator to multiply the value stored in `multiplyMe` by `11`.
- Use the `/` mathematical assignment operator to divide the value stored in `quarterMe` by `4`.
- Print out the new values to the console.

Increment and Decrement Operators

Other mathematical assignment operators include the *increment operator* (++) and *decrement operator* (--).

The increment operator will increase the value of the variable by 1. The decrement operator will decrease the value of the variable by 1. For example:

```
20  var a = 10;  
21  a++;  
22  console.log(a); // Output: 11  
23  
24  var b = 20;  
25  b--;  
26  console.log(b); // Output: 19
```

Just like the previous mathematical assignment operators (+=, -=, *=, /=), the variable's value is updated *and* assigned as the new value of that variable.

Your turn again!

Add the following variables to your code:

```
var gainedDollar = 3;  
var lostDollar = 50;
```

Using the increment operator, increase the value of `gainedDollar`.

Using the decrement operator, decrease the value of `lostDollar`.

Print out their new values to the console.

String Concatenation with Variables

In previous exercises, we assigned strings to variables. Now, let's go over how to connect, or concatenate, strings in variables.

The `+` operator can be used to combine two string values even if those values are being stored in variables:

```
1 var myPet = "armadillo";
2 console.log('I own a pet ' + myPet);
```

In the example above, we assigned the value `'armadillo'` to the `myPet` variable. On the second line, the `+` operator is used to combine three strings: `'I own a pet'`, the value saved to `myPet`, and `'. '`. We log the result of this concatenation to the console as: `I own a pet armadillo`

Create a variable named `favoriteAnimal` and set it equal to your favorite animal

Use `console.log()` to print `'My favorite animal: ANIMAL'` to the console. Use string concatenation so that `ANIMAL` is replaced with the value in your `favoriteAnimal` variable

typeof Operator

While writing code, it can be useful to keep track of the data types of the variables in your program. If you need to check the data type of a variable's value, you can use the `typeof` operator.

The `typeof` operator checks the value to its right and *returns*, or passes back, a string of the data type.

```
const unknown1 = 'foo';
console.log(typeof unknown1); // Output:
string

const unknown2 = 10;
console.log(typeof unknown2); // Output:
number

const unknown3 = true;
console.log(typeof unknown3); // Output:
boolean
```

Practice checking types

Create a new variable:

```
var newVariable = 'Experimenting with typeof';
```

Use `console.log()` to print the `typeof newVariable`.

Below the `console.log()` statement, reassign `newVariable` to 1.

Since you assigned this new value to `newVariable`, it has a new type! On the line below your reassignment, use `console.log()` to print `typeof newVariable` again.

Review - your turn!

Let's review what we've discussed about variables and arithmetic operations:

On the following slide is code that uses button clicks to change the value of variables that affect the properties of size and color.

We'll go over it together, and then you'll need to complete its functionality.

```
variables_review.html ●
```

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <p id="myP">JavaScript!</p>
6
7 <button type="button" id="increment">Increment</button>
8 <button type="button" id="red">Red</button>
9
10 <script>
11
12 var a;
13 a = 75;
14 var r;
15 r = 0;
16
17
18 var increment = document.getElementById("increment");
19 var red = document.getElementById("red");
20
21 document.getElementById("myP").style.fontSize = a+"px";
22
23 increment.addEventListener("click", function(){
24
25     a = a +1;
26     document.getElementById("myP").style.fontSize = a+"px";
27     console.log(a);
28
29 });
30
31 red.addEventListener("click", function(){
32
33     r = r+10;
34     document.getElementById("myP").style.color = "rgb"+("+"r+",0,0)";
35
36 });
37
38
39 </script>
40
41 </body>
42 </html>
```

In the html structure of this code we have a paragraph with some text: “JavaScript!” and we create two buttons and give them unique ID’s so we can target them in JS.

In the JavaScript, we create variables for the font-size (a) and red color (r)

Then, we create variables to target each button.

We use JavaScript’s addEventListener to function to bind a function to the appropriate button-click.

In the increment function, we increase the value of by 1, increasing the size.

In the red function, we increase the value of r by 10, increasing the red value of rgb.

We use string concatenation with variables to pass the values to the html structure.

Your turn!

- Create a button and accompanying function that does the opposite of increment. Clicking this button should shrink the text.
- Create a button for “green” and “blue” to affect the values of those colors. You can refer to the setup for “red” as a reference.

Conditional Statements



What are conditional statements?

In life, we make decisions based on circumstances. Think of an everyday decision as mundane as falling asleep— if we are tired, we go to bed, otherwise, we wake up and start our day.

These if-else decisions can be modeled in code by creating *conditional statements*. A conditional statement checks specific condition(s) and performs a task based on the condition(s).

Let's explore how programs make decisions by evaluating conditions and introduce logic into our code!

Types of Conditionals in JavaScript

In JavaScript we have the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

If statements

We often perform a task based on a condition. For example, if the weather is nice today, then we will go outside. If the alarm clock rings, then we'll shut it off. If we're tired, then we'll go to sleep. In programming, we can also perform a task based on a condition using an `if` statement:

```
if (true) {  
    console.log('This message will  
print!');  
}  
// Prints "This message will print!"
```

- The `if` keyword followed by a set of parentheses `()` which is followed by a *code block*, or *block statement*, indicated by a set of curly braces `{}`.
- Inside the parentheses `()`, a condition is provided that evaluates to `true` or `false`.
- If the condition evaluates to `true`, the code inside the curly braces `{}` runs, or executes.
- If the condition evaluates to `false`, the block won't execute.

Let's make an “if” statement

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>conditionals</title>
5  </head>
6  <body>
7  <script type="text/javascript">
8
9
10 </script>
11 </body>
12 </html>
```

In the <script> section of a new .html file:

Declare a variable named `sale`. Assign the value `true` to it.

Now create an `if` statement. Provide the `if` statement a condition of `sale`. Inside the code block of the `if` statement, `console.log()` the string `'Time to buy!'`.

Run the code

Notice that the code inside the `if` statement ran, since `'Time to buy!'` was logged to the console. reassigned `sale` to `false`. Run your code and observe what happens.

Browser-based example

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>HTML div</title>
5 <style type="text/css">
6   #div1{
7     width: 100px;
8     float:left;
9     height:100px;
10    background:red;
11    margin:10px;
12  }
13 </style>
14 </head>
15 <body>
16
17 <div id="div1">
18 First DIV
19 </div>
20
21 <script type="text/javascript">
22
23
24 </script>
25 </body>
26 </html>
```

Here's some code that draws a `<div>` as red box onto a browser page.

In the `<script>` section, create a variable called "color" and assign it the value of true.

Write an if statement that changes the box to green if color is true.

You'll need to use this within the code block section of your statement:

```
document.getElementById("div1").style.background = "green";
```

if . . . else statements

In the previous exercises, we used an `if` statement that checked a condition to decide whether or not to run a block of code. If the condition evaluated to `false`, the program just didn't run the code.

But, in many cases, we'll have code we want to run if our condition evaluates to `false`. If we wanted to add some default behavior to the `if` statement, we can add an `else` statement to run a block of code when the condition evaluates to `false`. Take a look at the inclusion of an `else` statement:

```
if (false) {  
    console.log('The code in this block  
will not run.');
```

```
} else {  
    console.log('But the code in this block  
will!');
```

```
}
```

```
// Prints "But the code in this block  
will!"
```

An `else` statement must be paired with an `if` statement, and together they are referred to as an `if...else` statement. The code inside the `else` statement code block will execute when the `if` statement's condition evaluates to `false`. `if...else` statements allow us to automate solutions to yes-or-no questions, also known as *binary decisions*.

Your turn:

Modify your first “if” statement, the one that outputs time to buy if sale=true:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>conditionals</title>
5  </head>
6  <body>
7
8  <script type="text/javascript">
9  var sale = true;
10
11 if(sale){
12     console.log('Time to buy!');
13 }
14
15 </script>
16
17 </body>
18 </html>
```

Set sale to false.

Then, add an `else` statement to the existing `if` statement. Inside the code block of the `else` statement, `console.log()` the string 'Time to wait for a sale.'

See what happens as sale changes from true to false.

Browser-based example

Here's our code that changes the background color of a div to green if color = true.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>HTML div</title>
5 <style type="text/css">
6   #div1{
7     width: 100px;
8     float:left;
9     height:100px;
10    background:red;
11    margin:10px;
12  }
13 </style>
14 </head>
15 <body>
16
17 <div id="div1">
18 First DIV
19 </div>
20
21 <script type="text/javascript">
22   var color = true;
23
24   if(color){
25
26     document.getElementById("div1").style.background = "green";
27   }
28
29 </script>
30 </body>
31 </html>
```

Add an “else” statement that changes the color to blue whenever color= false;

Change color to equal false to test your code.

Operators

Comparison Operators

When writing conditional statements, sometimes we need to use different types of operators to compare values. These operators are called *comparison operators*.

Here is a list of some handy comparison operators and their syntax:

- Less than: <
- Greater than: >
- Less than or equal to: <=
- Greater than or equal to: >=
- Is equal to: == / === (identity operator)
- Is NOT equal to: !=

Comparison operators compare the value on the left with the value on the right. For instance:

```
10 < 12 // Evaluates to true
```

It can be helpful to think of comparison statements as questions. When the answer is "yes", the statement evaluates to `true`, and when the answer is "no", the statement evaluates to `false`. The code above would be asking: is 10 less than 12? Yes! So `10 < 12` evaluates to `true`.

We can also use comparison operators on different data types like strings:

```
'apples' === 'oranges' // false
```

In the example above, we're using the *identity operator* (`==`) to check if the string `'apples'` is the same as the string `'oranges'`. Since the two strings are not the same, the comparison statement evaluates to `false`.

All comparison statements evaluate to either `true` or `false` and are made up of:

- Two values that will be compared.
- An operator that separates the values and compares them accordingly (`>`, `<`, `<=`, `>=`, `==`, `!=`).

Let's Practice

Using `var`, create a variable named `hungerLevel` and set it equal to `7`.

Write an `if...else` statement using a comparison operator. The condition should check if `hungerLevel` is greater than `7`. If so, the conditional statement should log, `'Time to eat!'`. Otherwise, it should log `'We can eat later!'`.

Browser-based example

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>HTML div</title>
5  <style type="text/css">
6      #div1{
7          width: 100px;
8          float:left;
9          height:100px;
10         margin:10px;
11         border-style: solid;
12     }
13 </style>
14 </head>
15 <body>
16
17 Enter a number: <input type="text" id="myText">
18 <button onclick="myFunction()">submit</button>
19
20
21 <div id="div1">
22 First DIV
23 </div>
24
25 <script type="text/javascript">
26     //var color = 9;
27 var color;
28
29 function myFunction() {
30     color = document.getElementById("myText").value;
31     console.log(color);
32 }
33
34
35 </script>
36 </body>
37 </html>
```

Here's our previous code slightly modified to accept input from an html text field.

Notice when we submit the number entered by the user, we assign that value to our color variable.

Modify this code so that the square will be red if a number > 50 is entered and green if a number < 50 is entered. If the number $\ == 50$ the square will be blue.

Your if statements should go within the function brackets.

Logical Operators

Logical operators are often used in conditional statements to add another layer of logic to our code.

Logical Operators

We can use logical operators to add more sophisticated logic to our conditionals. There are three logical operators:

- the *and* operator (`&&`)
- the *or* operator (`||`)
- the *not* operator, otherwise known as the *bang* operator (`!`)

&& (logical and)

When we use the `&&` operator, we are checking that two things are `true`:

```
if (stopLight === 'green' && pedestrians
    === 0) {
  console.log('Go!');
} else {
  console.log('Stop');
}
```

When using the `&&` operator, both conditions *must* evaluate to `true` for the entire condition to evaluate to `true` and execute. Otherwise, if either condition is `false`, the `&&` condition will evaluate to `false` and the `else` block will execute.

|| (logical or)

When using the `||` operator, only one of the conditions must evaluate to `true` for the overall statement to evaluate to `true`

```
if (day === 'Saturday' || day ===  
    'Sunday') {  
    console.log('Enjoy the weekend!');  
} else {  
    console.log('Do some work.');//  
}
```

In the code example above, if either `day === 'Saturday'` or `day === 'Sunday'` evaluates to `true` the `if`'s condition will evaluate to `true` and its code block will execute. If the first condition in an `||` statement evaluates to `true`, the second condition won't even be checked. Only if `day === 'Saturday'` evaluates to `false` will `day === 'Sunday'` be evaluated. The code in the `else` statement above will execute only if both comparisons evaluate to `false`.

! (not!)

The `!` *not operator* reverses, or *negates*, the value of an operator.:

```
let excited = true;  
console.log(!excited); // Prints false  
  
let sleepy = false;  
console.log(!sleepy); // Prints true
```

Essentially, the `!` operator will either take a `true` value and pass back `false`, or it will take a `false` value and pass back `true`.

Practice!

Create two variables:

```
var mood = sleepy;
```

```
var tirednessLevel = 6;
```

Let's create an `if...else` statement that checks if `mood` is `'sleepy'` and `tirednessLevel` is greater than `8`.

If both conditions evaluate to `true`, then `console.log()` the string `'time to sleep'`. Otherwise, we should `console.log()` `'not bed time yet'`.

Browser-based example

Let's modify our box-changing-color code to make the logic more sophisticated by using logical operators.

Modify the if statements so that:

When a number ≥ 0 and ≤ 25 is entered the box will be red.

When a number > 25 and ≤ 50 is entered the box will be green.

When a number > 50 and ≤ 75 is entered the box will be blue

When a number > 75 and ≤ 100 is entered the box will be yellow

When the string 'purple' or 'Purple' is entered the box will be purple

When the string 'orange' or 'pumpkin' is entered the box will be orange

Else If Statement

The `else if` statement allows for more than two possible outcomes. You can add as many `else if` statements as you'd like, to make more complex conditionals!

The `else if` statement always comes after the `if` statement and before the `else` statement. The `else if` statement also takes a condition. Let's take a look at the syntax:

```
let stopLight = 'yellow';

if (stopLight === 'red') {
  console.log('Stop!');
} else if (stopLight === 'yellow') {
  console.log('Slow down.');
} else if (stopLight === 'green') {
  console.log('Go!');
} else {
  console.log('Caution, unknown!');
}
```

The `else if` statements allow you to have multiple possible outcomes. `if/else if/else` statements are read from top to bottom, so the first condition that evaluates to `true` from the top to bottom is the block that gets executed.

In the example above, since `stopLight === 'red'` evaluates to `false` and `stopLight === 'yellow'` evaluates to `true`, the code inside the first `else if` statement is executed. The rest of the conditions are not evaluated. If none of the conditions evaluated to `true`, then the code in the `else` statement would have executed.

Practice!

Create a var called “season” and assign it a starting value of “summer”.

Let’s write an if statement to check if it’s Spring:

```
var season = 'summer';

if (season === 'spring') {
  console.log('It\'s spring! The trees are budding!');
}
```

add an `else if` statement, and a `console.log()` that prints the string `'It's winter! Everything is covered in snow.'`. If `season = "winter"`.

Add another `else if` statement that checks if `season` is equal to `'fall'`.

Inside the code block of the `else if` statement you just created, add a `console.log()` that prints the string `'It's fall! Leaves are falling!'`.

Add an `else if` statement that checks if `season` is equal to `'summer'`.

Inside the code block of the `else if` statement you just created, add a `console.log()` that prints the string `'It's sunny and warm because it's summer!'`.

And finally, add an `else` statement if to `console.log ('Invalid entry')` if `season` isn't spring, summer, winter, or fall.

Browser Practice!

Take your else if code and modify it so that instead of just a console.log message, a seasonal picture appears when the variable season is changed.

Hint: you'll need to create a div in the body of your html and get the element by ID. You can use the innerHTML method to display the image.

Switch

A `switch` statement provides an alternative syntax to `else if` that is easier to read and write. A `switch` statement looks like this:

```
let groceryItem = 'papaya';

switch (groceryItem) {
  case 'tomato':
    console.log('Tomatoes are $0.49');
    break;
  case 'lime':
    console.log('Limes are $1.49');
    break;
  case 'papaya':
    console.log('Papayas are $1.29');
    break;
  default:
    console.log('Invalid item');
    break;
}

// Prints 'Papayas are $1.29'
```

- The `switch` keyword initiates the statement and is followed by `(...)`, which contains the value that each `case` will compare. In the example, the value or expression of the `switch` statement is `groceryItem`.
- Inside the block, `{ ... }`, there are multiple `cases`. The `case` keyword checks if the expression matches the specified value that comes after it. The value following the first `case` is `'tomato'`. If the value of `groceryItem` equalled `'tomato'`, that `case`'s `console.log()` would run.
- The value of `groceryItem` is `'papaya'`, so the third `case` runs—`Papayas are $1.29`is logged to the console.
- The `break` keyword tells the computer to exit the block and not execute any more code or check any other cases inside the code block. Note: Without the `break` keyword at the end of each `case`, the program would execute the code for all matching cases and the `default` code as well. This behavior is different from `if/else` conditional statements which execute only one block of code.
- At the end of each `switch` statement, there is a `default` statement. If none of the `cases` are true, then the code in the `default` statement will run.

Practice!

Let's write a `switch` statement to decide what medal to award an athlete.

Create a `var athleteFinalPosition;`

start by writing a `switch` statement with `athleteFinalPosition` as its expression.

Inside the `switch` statement, add three `cases`:

- The first `case` checks for the value '`first place`'
 - If the expression's value matches the value of the `case` then `console.log()` the string '`You get the gold medal!`'
- The second `case` checks for the value '`second place`'
 - If the expression's value matches the value of the `case` then `console.log()` the string '`You get the silver medal!`'
- The third `case` checks for the value '`third place`'
 - If the expression's value matches the value of the `case` then `console.log()` the string '`You get the bronze medal!`'

Remember to add a `break` after each `console.log()`.

Now, add a `default` statement at the end of the `switch` that uses `console.log()` to print '`No medal awarded.`'. If `athleteFinalPosition` does not equal any value of our `cases`, then the string '`No medal awarded.`' is logged to the console.

Remember to add the `break` keyword at the end of the `default` case.

Browser Practice

Write another version of your seasons code that uses a switch statement instead of else if.

JavaScript III

functions

Functions

A JavaScript function is a block of code designed to perform a particular task.

```
function myFunction(p1, p2) {  
    return p1 * p2;    // The function returns the product of p1 and p2  
}
```

A JavaScript function is executed when "something" invokes it (calls it).

```
myFunction(4, 3);
```

Syntax

There are a few ways to use and declare functions in JavaScript, including function expression, concise expression, arrow expression, etc.

This is the most basic and common function declaration:

```
FUNCTION  
KEYWORD      IDENTIFIER  
|             |  
function greetWorld() {  
    console.log('Hello, World!');  
}
```

KEY

● Function body

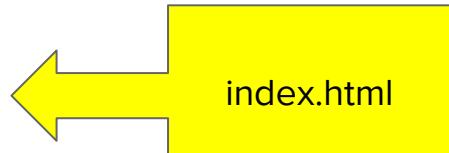
A function declaration consists of:

- The `function` keyword.
- The name of the function, or its identifier, followed by parentheses.
- A function body, or the block of statements required to perform a specific task, enclosed in the function's curly brackets, `{ }`.

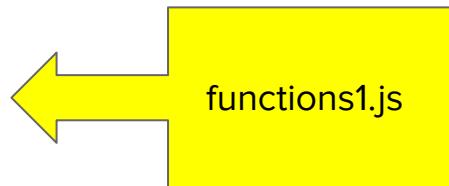
Basic Example

Let's write a Hello World function:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>JS Functions</title>
5  </head>
6
7  <body>
8
9  <div id="section1">
10 </div>
11
12 <script type="text/javascript" src="functions1.js"></script>
13
14 </body>
15 </html>
```



```
1  function HelloWorld(){
2      document.getElementById("section1").innerHTML = "Hello World!";
3  }
```



Calling a function

In order to execute the code in a function we have to call it.

We can call a function just by referring to it's name and adding an opening and closing parentheses + ending semi-colon:



5 myFunction();

Call your function

In your “functions1.js” call your Hello World function.
And preview the “index.html” file to see how it works.

Now, delete the HelloWorld(); function call from functions1.js

Now, let's trigger the function call with a button click, Create a “functions1.html” file:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>functions1</title>
5  </head>
6  <body>
7
8      <div id="section1">
9
10     </div>
11
12     <button onclick="HelloWorld();>submit</button>
13
14
15     <script type="text/javascript" src="functions1.js"></script>
16
17 </body>
18 </html>
```

* hoisting

The convention in most C-based languages is that functions have to be defined and declared before they can be called. JavaScript is an exception to this rule.

The **hoisting** feature in JavaScript which allows access to function declarations before they're defined.

```
1  HelloWorld();  
2  
3  function HelloWorld(){  
4      document.getElementById("section1").innerHTML = "Hello World!";  
5 }
```

A large yellow starburst graphic with several points, positioned to the right of the code snippet. It contains the text "This works too!" in black.

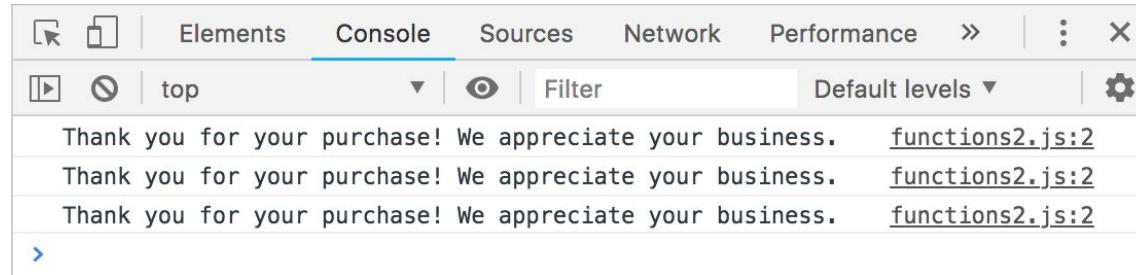
Your turn

- In your JS file create another function called `getReminder()`
- This function should use the `.innerHTML` method to output the string ‘Water the Plants’ into a div with the id of “section2”. (You’ll need to create this div in your `.html` file.)
- In your JS file create another function called `greetInSpanish()`
- This function should use the `.innerHTML` method to display an appropriate image into a div with the id of “section3”. (You’ll need to create this div in your `.html` file.)
- In your html, create three more buttons to trigger each of these functions.

Functional utility

One of the reasons functions are powerful and important concepts in programming is because we can wrap up complicated or laborious code as a single function and call it as many times as we need to. Let's demo this in a functions2.js file:

```
1  function sayThanks(){
2    console.log('Thank you for your purchase! We appreciate your business.');
3  }
4  sayThanks();
5  sayThanks();
6  sayThanks();
```

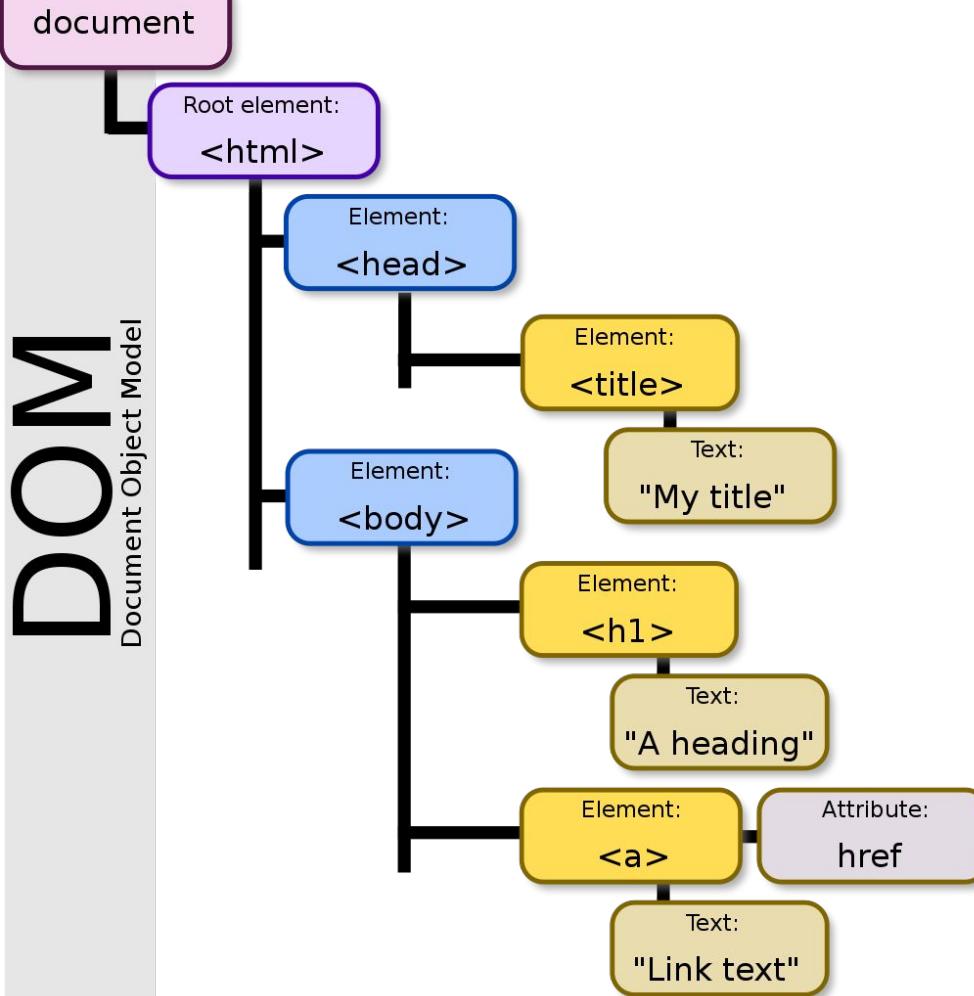


DOM STRUCTURE

Document Object Model Manipulation

DOM

Document Object Model



Document Object Model

The **Document Object Model (DOM)** is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document. Nodes can have event handlers attached to them. Once an event is triggered, the event handlers get executed.

Manipulating the DOM

The DOM was designed to be independent of any particular programming language, making the structural representation of the document available from a single, consistent API. Though we focus exclusively on JavaScript in this reference documentation, implementations of the DOM can be built for any language:

```
<p>One</p>
<p>Two</p>
<p>Three</p>
```

```
var paragraphs = document.getElementsByTagName("p"); // paragraphs[0] is the
first <p> element // paragraphs[1] is the second <p> element, etc.
alert(paragraphs[0].nodeName);
```

Property Accesors

Property accessors provide access to an object's properties by using the dot notation or the bracket notation:

```
1 var person = {};
2 person['firstname'] = 'Mario';
3 person['lastname'] = 'Rossi';
4
5 console.log(person.firstname);
6 // expected output: "Mario"
7
8 person = { 'firstname': 'John', 'lastname': 'Doe' }
9
10 console.log(person['lastname']);
11 // expected output: "Doe"
12
```

JavaScript HTML DOM Elements

```
document.getElementById(" ");
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementbyid

```
document.getElementsByTagName(" ");
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbytagname2

```
document.getElementsByClassName(" ");
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_getelementsbyclassname

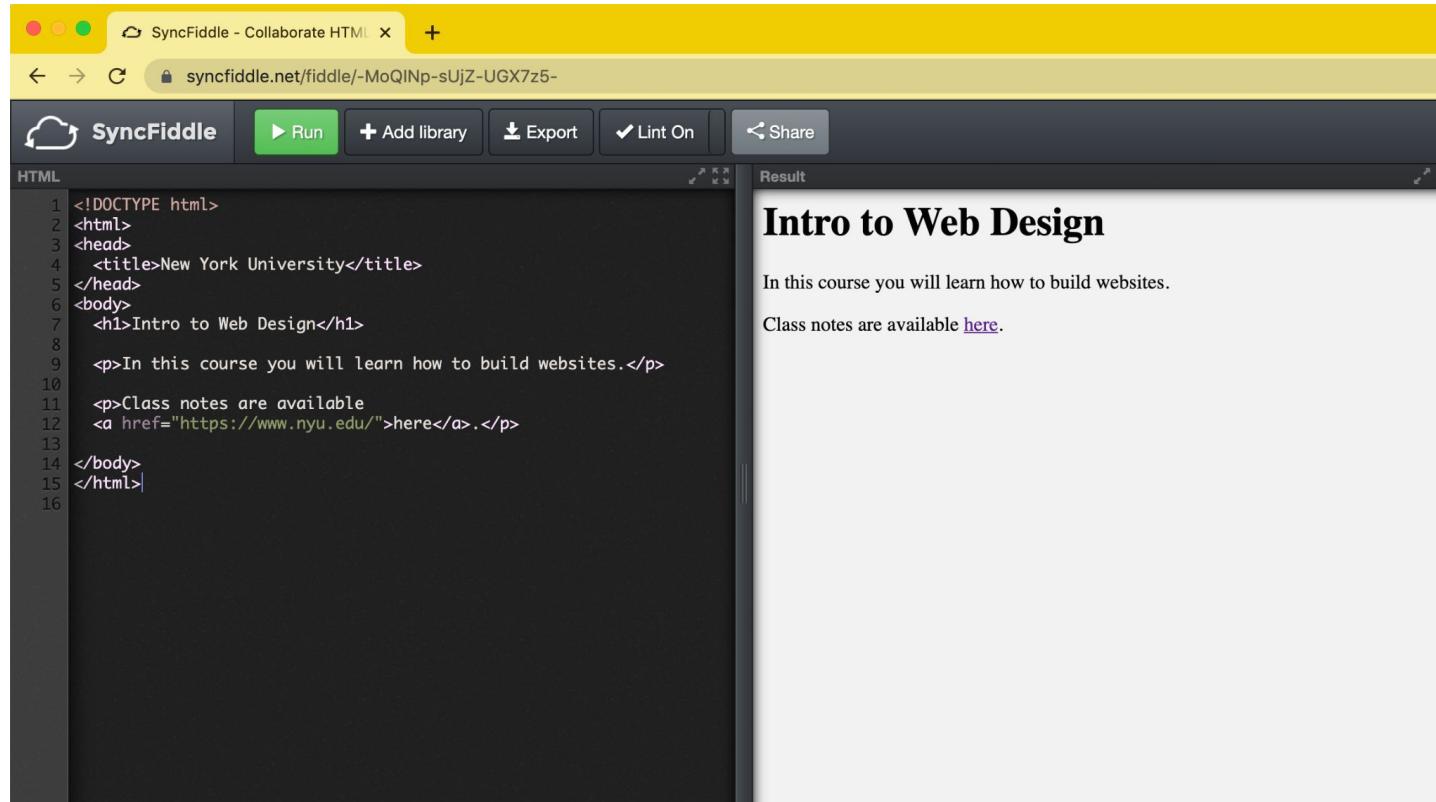
```
document.querySelectorAll(" "); (CSS selector)
```

https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_queryselectorall

The following HTML objects (and object collections) are also accessible:

- [document.anchors](#)
- [document.body](#)
- [document.documentElement](#)
 - [document.embeds](#)
 - [document.forms](#)
 - [document.head](#)
 - [document.images](#)
 - [document.links](#)
 - [document.scripts](#)
 - [document.title](#)

Let's Practice: <https://syncfiddle.net/fiddle/-MoQINp-sUjZ-UGX7z5->



The screenshot shows the SyncFiddle web application interface. The top navigation bar includes a yellow header with three dots, a back/forward button, a search bar containing the URL, and a plus sign for creating new fiddles. Below the header is a toolbar with a cloud icon labeled "SyncFiddle", a "Run" button, an "Add library" button, an "Export" button, a "Lint On" checkbox, and a "Share" button.

The main area is divided into two sections: "HTML" on the left and "Result" on the right. The "HTML" section contains the following code:

```
<!DOCTYPE html>
<html>
<head>
<title>New York University</title>
</head>
<body>
<h1>Intro to Web Design</h1>
<p>In this course you will learn how to build websites.</p>
<p>Class notes are available
<a href="https://www.nyu.edu/">here</a>.</p>
</body>
</html>
```

The "Result" section displays the rendered HTML output:

Intro to Web Design

In this course you will learn how to build websites.

Class notes are available [here](https://www.nyu.edu/).

The DOM

Introduction to Web Design

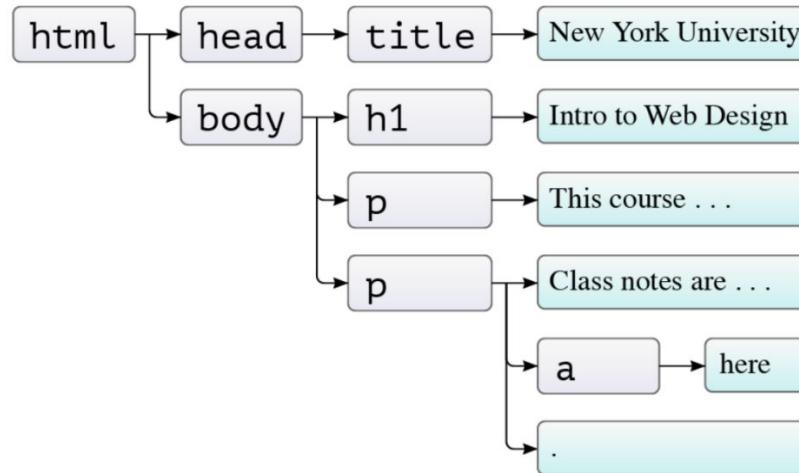
Document Object Model



Adapted from Eloquent JavaScript: The Document Object Model

Introduction to Web Design

Document Object Model



Adapted from Eloquent JavaScript: The Document Object Model

Introduction to Web Design

DOM Queries

Document Object Model

JavaScript methods that find elements in the DOM tree are called “DOM queries.”

DOM queries may return one element, or they may return a “node list.”

Which DOM query you use depends on what you want to do and the scope of browser support required.

Introduction to Web Design

Document Object Model

DOM Queries

Methods that return a single element node:

- `getElementById()`
- `querySelector()`

Introduction to Web Design

DOM Queries

Document Object Model

Methods that return one or more elements as a node list:

- `getElementsByClassName()`
- `getElementsByTagName()`
- `querySelectorAll()`

Let's change the link via the <a> element

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>New York University</title>
5  </head>
6  <body>
7      <h1>Intro to Web Design</h1>
8
9      <p>In this course you will learn how to build websites.</p>
10
11     <p>Class notes are available
12     <a href="notes.html">here</a>.</p>
13
14 <script type="text/javascript">
15
16     let link= document.querySelector('a');
17     link.textContent = "Mozilla Developers Network";
18     link.href = "https://developer.mozilla.org/en-US/"
19
20
21 </script>
22
23 </body>
24 </html>
```

Creating new elements

```
let sect= document.querySelector('section');
let para= document.createElement('p');
para.textContent = "JavaScript is fun!";
sect.appendChild(para);
```

Generate some CSS styling with JS

Use JS to create a variable selecting the ‘para.style’ element.

Change the following:

- color='red'
- backgroundColor='yellow'
- padding='10px'
- width ='250px'
- textAlign='center'

When you are done, use the element inspector in your browser to see how the code is rendered.

JavaScript III

functions

Functions

A JavaScript function is a block of code designed to perform a particular task.

```
function myFunction(p1, p2) {  
    return p1 * p2;    // The function returns the product of p1 and p2  
}
```

A JavaScript function is executed when "something" invokes it (calls it).

```
myFunction(4, 3);
```

Syntax

There are a few ways to use and declare functions in JavaScript, including function expression, concise expression, arrow expression, etc.

This is the most basic and common function declaration:

```
FUNCTION  
KEYWORD      IDENTIFIER  
|             |  
function greetWorld() {  
    console.log('Hello, World!');  
}
```

KEY

● Function body

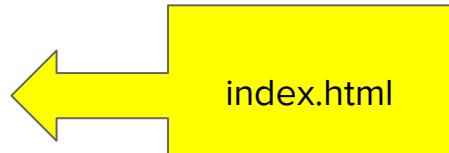
A function declaration consists of:

- The `function` keyword.
- The name of the function, or its identifier, followed by parentheses.
- A function body, or the block of statements required to perform a specific task, enclosed in the function's curly brackets, `{ }`.

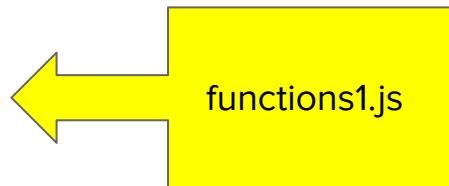
Basic Example

Let's write a Hello World function:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>JS Functions</title>
5  </head>
6
7  <body>
8
9  <div id="section1">
10 </div>
11
12 <script type="text/javascript" src="functions1.js"></script>
13
14 </body>
15 </html>
```



```
1  function HelloWorld(){
2      document.getElementById("section1").innerHTML = "Hello World!";
3  }
```



Calling a function

In order to execute the code in a function we have to call it.

We can call a function just by referring to it's name and adding an opening and closing parentheses + ending semi-colon:



A screenshot of a code editor interface. On the left, there is a dark grey sidebar with the number '5' displayed in white. To the right of the sidebar, there is a large, semi-transparent blue rectangular overlay. Inside this overlay, the text 'myFunction();' is written in a light blue font. The background of the code editor shows some faint, illegible code.

Call your function

In your “functions1.js” call your Hello World function.
And preview the “index.html” file to see how it works.

Now, delete the HelloWorld(); function call from functions1.js

Now, let's trigger the function call with a button click, Create a “functions1.html” file:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>functions1</title>
5  </head>
6  <body>
7
8      <div id="section1">
9
10     </div>
11
12     <button onclick="HelloWorld();>submit</button>
13
14
15     <script type="text/javascript" src="functions1.js"></script>
16
17 </body>
18 </html>
```

* hoisting

The convention in most C-based languages is that functions have to be defined and declared before they can be called. JavaScript is an exception to this rule.

The **hoisting** feature in JavaScript which allows access to function declarations before they're defined.

```
1  HelloWorld();  
2  
3  function HelloWorld(){  
4      document.getElementById("section1").innerHTML = "Hello World!";  
5 }
```

A large yellow starburst graphic with several points, positioned to the right of the code snippet. It contains the text "This works too!" in black.

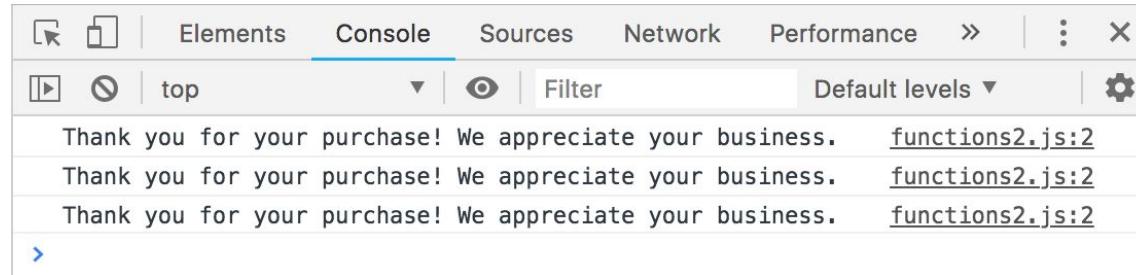
Your turn

- In your JS file create another function called `getReminder()`
- This function should use the `.innerHTML` method to output the string ‘Water the Plants’ into a div with the id of “section2”. (You’ll need to create this div in your `.html` file.)
- In your JS file create another function called `greetInSpanish()`
- This function should use the `.innerHTML` method to display an appropriate image into a div with the id of “section3”. (You’ll need to create this div in your `.html` file.)
- In your html, create three more buttons to trigger each of these functions.

Functional utility

One of the reasons functions are powerful and important concepts in programming is because we can wrap up complicated or laborious code as a single function and call it as many times as we need to. Let's demo this in a functions2.js file:

```
1  function sayThanks(){
2    console.log('Thank you for your purchase! We appreciate your business.');
3  }
4  sayThanks();
5  sayThanks();
6  sayThanks();
```



Introduction to Web Design

Events

NYU

nyu.edu

Information For: Students Faculty Alumni Employees Community COVID-19 Info Login to NYU Home All NYU

About NYU Admissions Academics University Life Research Search

NYU

More Than 20 NYU Faculty Named to List of Highly Cited Researchers

Application Deadlines are Approaching

Scientists Identify New Force Behind Past Mass Extinction Event

Academic Programs

Bilingualism Comes Naturally to Our Brains

Graduate Admissions

@nyuniversity

Financial Aid and Scholarships

Your Anxiety Isn't Going Anywhere. Here's How To Put

Facebook

Financial Aid and Scholarships

Your Anxiety Isn't Going Anywhere. Here's How To Put

Facebook

Introduction to Web Design

UI Events

Events

load

unload

error

resize

scroll

UI Event Examples

load : <https://syncfiddle.net/fiddle/-Mp2wZzhGZwl4g1xGPBu>

unload: <https://syncfiddle.net/fiddle/-Mp2xO8gquF1FNHZKRj->

error: <https://syncfiddle.net/fiddle/-Mp2yArZdXmx7cHXESX3>

resize: <https://syncfiddle.net/fiddle/-Mp2zANiD5AZU5S5kwsd>

scroll: <https://syncfiddle.net/fiddle/-Mp2zrEeSQJXVTOOpkND>

Introduction to Web Design

Events

Keyboard Events

keydown

keyup

keypress

https://syncfiddle.net/fiddle/-Mp302VK2YX79J31x_am

Introduction to Web Design

Events

Mouse Events

click

dblclick

mousedown

mouseup

mousemove

mouseover

mouseout

https://syncfiddle.net/fiddle/-Mp37W_05Usl-kKKCV9i

Introduction to Web Design

Events

Focus Events

focus

blur

<https://tools.knowledgewalls.com/realtimejavascripteditor/How-to-use-focus-and-blur-method-in-Javascript-with-Example>

Introduction to Web Design

Form Events

Events

input

change

submit

reset

cut

copy

paste

select

Form Events

change: <https://syncfiddle.net/fiddle/-Mp3Ds0BAJo3RPymk0TO>

submit: <https://syncfiddle.net/fiddle/-Mp3EBaQyEReJggFLE6f>

select: <https://syncfiddle.net/fiddle/-Mp3EgCALB25PS3KWu-B>

Introduction to Web Design

Events

Mutation Events

DOMSubtreeModified

DOMNodeInserted

DOMNodeRemoved

DOMNodeInsertedIntoDocument

DOMNodeRemovedFromDocument

<https://syncfiddle.net/fiddle/-Mp3FMOEyZIL7mB50beh>

<https://syncfiddle.net/fiddle/-Mp3GP41Q9LoAOJQQMOf>

https://syncfiddle.net/fiddle/-Mp3HBB_ByFyrHZWRNkD

Introduction to Web Design

Events

Touch Events

touchstart

touchmove

touchend

touchcancel

Introduction to Web Design

Event Handling



Events

1. Select an element for the script to respond to.
2. Specify which event will trigger the response.
3. Run code specific to that event.

Introduction to Web Design

Binding

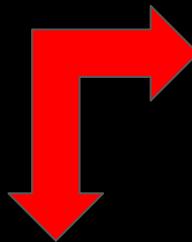
HTML events are "things" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "react" on these events.

HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document.

OLD

```
Single-Quote  
<element onclick='Javascript Code''></element>  
Double-Quote  
<element onclick="Javascript Code"></element>
```



Events

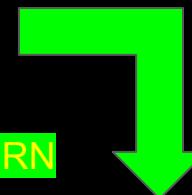
Specifying which event will trigger the response is also known as "binding."

There are three different ways to bind an event to an element.

- HTML event handlers
- DOM event handlers
- DOM event listeners

MODERN

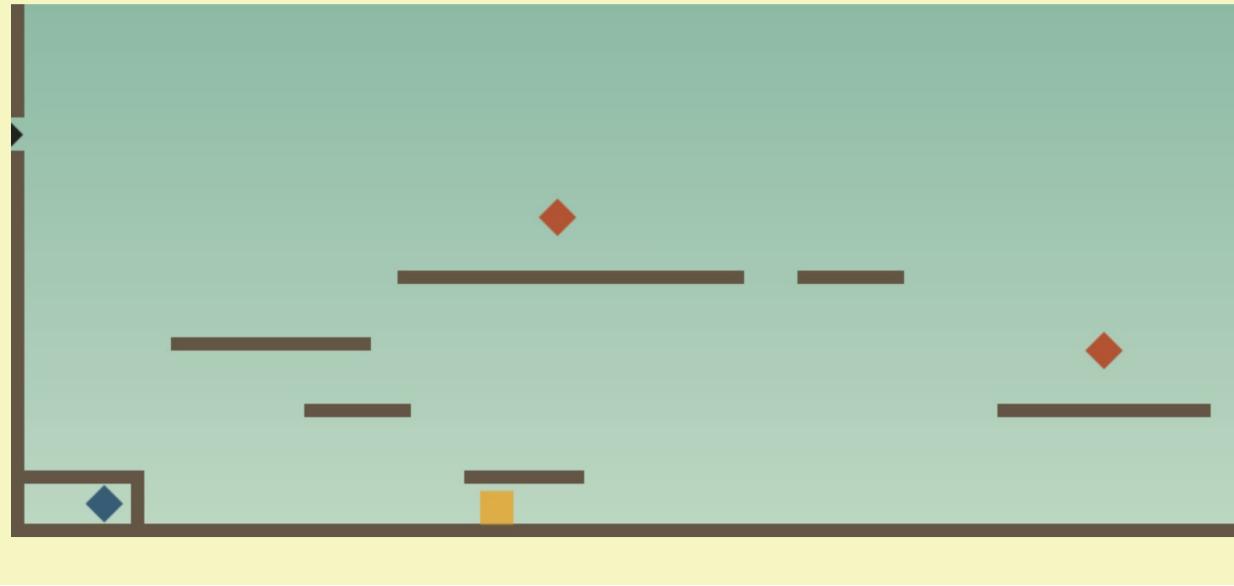
```
element.addEventListener(event, function, useCapture);
```



https://syncfiddle.net/fiddle/-Mp3NJbHph0EWR_X3ALI

Simple Platformer

Use the arrow keys and your platforming skills to reach the blue diamond



https://i6.cims.nyu.edu/~mr6465/web_design/javascript/platformer/

Introduction to Web Design

Version
Control



Introduction to Web Design

Version Control



Version Control

A system that records changes to a file or set of files over time so that you can recall specific versions later

Commonly used for software source code but any type of file can be placed under version control

A Version Control System (VCS) allows you to:

- Revert files back to a previous state
- Review changes made over time
- Collaborate more efficiently
- Maintain project backups

Introduction to Web Design

Version Control



my-project-1.txt



my-project-2.txt



my-project-3.txt

Version control software **keeps track of every modification to the code in a special kind of database**. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members

Introduction to Web Design

Centralized Version Control

Version Control

Centralized Version Control Systems were developed to allow collaboration with developers on other systems.

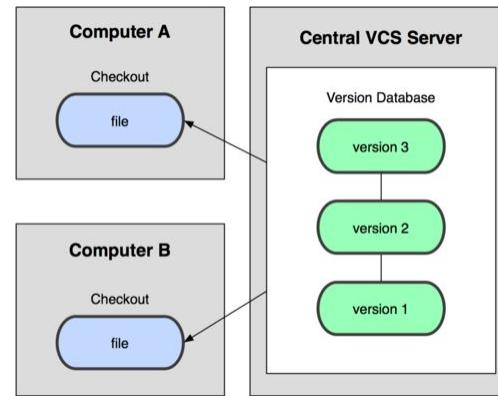
With a CVCS, a single server contains all the versioned files and clients “check out” files from that central place.

For many years, this has been the standard for version control.

The downside of centralized version control is the vulnerability of having the entire history of a project in one place.

Introduction to Web Design

Version Control



Introduction to Web Design

Distributed Version Control

Version Control

With Distributed Version Control Systems, clients don't just check out the latest snapshot of files, they fully mirror the entire history of the project.

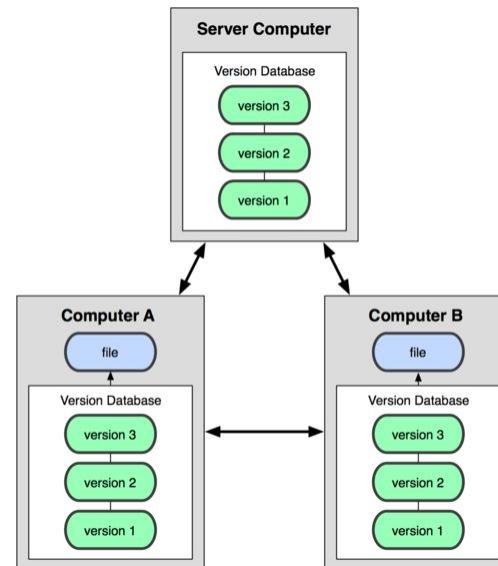
If a server dies, anyone with a copy of all the versioned files can restore it to the server.

Every checkout is really a full backup of all the data.

You can also collaborate with different groups of people in different ways simultaneously within the same project.

Introduction to Web Design

Version Control



Introduction to Web Design

Git History

Version Control

Git was created in 2005 by Linus Torvalds and the Linux development community for Linux kernel maintenance

Linux is an open source operating system project of fairly large scope

Its goal was to be a fully distributed VCS with a simple design, support for non-linear development, and the ability to handle large projects efficiently

Introduction to Web Design

Git Basics

Version Control

Git thinks of its data like a set of snapshots of a mini file system.

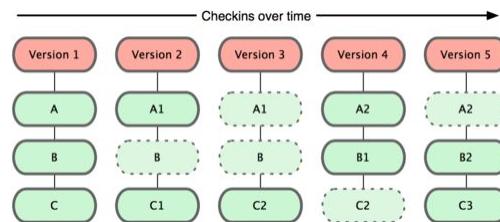
Every time you save the state of your project, it basically takes a picture of what all your files look like then and stores a reference to that snapshot.

To be efficient, if files have not changed, Git doesn't store the file again—just a link to the previous identical file it has already stored.

This makes Git more like a mini file system with some powerful tools built on top of it.

Introduction to Web Design

Version Control



Introduction to Web Design

Git States

Version Control

Git has three main states that your files can reside in: modified, staged, and committed.

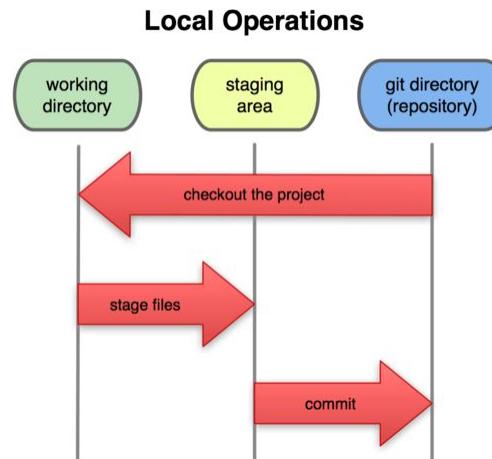
Modified means that you have changed the file but have not committed it to your database yet.

Staged means that you have marked a modified file in its current version to go into your next commit snapshot.

Committed means that the data is safely stored in your local database.

Introduction to Web Design

Version Control



Introduction to Web Design

Git Workflow

Version Control

1. Modify files in your working directory.
2. Stage the files, adding snapshots of them to your staging area.
3. Commit changes, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.



hello-git — bash — 80x24

```
Last login: Sun Nov 28 16:30:19 on ttys000
(base) Marks-MBP:~ markramos$ cd Documents/NYU_WebI
(base) Marks-MBP:NYU_WebI markramos$ mkdir hello-git
(base) Marks-MBP:NYU_WebI markramos$ cd hello-git
(base) Marks-MBP:hello-git markramos$ git init
Initialized empty Git repository in /Users/markramos/Documents/NYU_WebI/hello-git/.git/
(base) Marks-MBP:hello-git markramos$ echo "Hello_Git" >> README.md
(base) Marks-MBP:hello-git markramos$ git add README.md
(base) Marks-MBP:hello-git markramos$ git commit -m "Initial Commit"
[master (root-commit) 2ca29dd] Initial Commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
(base) Marks-MBP:hello-git markramos$
```

Introduction to Web Design

GitHub

Version Control

GitHub is a web-based hosting service that uses the Git VCS.

The site also provides social networking functionality such as feeds, followers, wikis, and statistics.

The company was founded in 2008 and is located in San Francisco.

In addition to computer programmers, architects, musicians, municipal governments, and academics are among its users.

mhr1235/hello-git

github.com/mhr1235/hello-git

Search or jump to... Pull requests Issues Marketplace Explore

mhr1235 / hello-git Public

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/mhr1235/hello-git.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# hello-git" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/mhr1235/hello-git.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/mhr1235/hello-git.git
git branch -M main
git push -u origin main
```

Personal Access Tokens x +

github.com/settings/tokens

Search or jump to... /

Pull requests Issues Marketplace Explore

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the GitHub API.

NYU Web I — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, Last used within the last week
admin:public_key, admin:repo_hook, delete:packages, delete_repo, gist, notifications, repo, user, workflow, write:discussion, write:packages

Expires on Tue, Dec 28 2021. Delete

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

© 2021 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

```
[base] Marks-MBP:hello-git markramos$ git remote add origin https://github.com/mhr1235/hello-git.git  
[base] Marks-MBP:hello-git markramos$ git remote -v  
origin https://github.com/mhr1235/hello-git.git (fetch)  
origin https://github.com/mhr1235/hello-git.git (push)
```

```
[base] Marks-MBP:hello-git markramos$ git branch -M main  
[base] Marks-MBP:hello-git markramos$ git push -u origin main
```

```
Username for 'https://github.com': mhr1235  
Password for 'https://mhr1235@github.com': Use token as pwd  
Enumerating objects: 3, done.  
Counting objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 236 bytes | 236.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
To https://github.com/mhr1235/hello-git.git  
 * [new branch]      main -> main  
Branch 'main' set up to track remote branch 'main' from 'origin'.  
(base) Marks-MBP:hello-git markramos$
```



github:pages

<https://pages.github.com/>

Pages

github.com/mhr1235/airports-main/settings/pages

Search or jump to... / Pull requests Issues Marketplace Explore

mhr1235 / airports-main Public Unwatch 1 Star 0 Fork

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

✓ Your site is published at <https://mhr1235.github.io/airports-main/>

Source

Your GitHub Pages site is currently being built from the main branch. [Learn more.](#)

Branch: main / (root) Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme. [Learn more.](#)

Choose a theme

Custom domain

Custom domains allow you to serve your site from a domain other than mhr1235.github.io. [Learn more.](#)

Save Remove

Enforce HTTPS

- Options
- Manage access
- Security & analysis
- Branches
- Webhooks
- Notifications
- Integrations
- Deploy keys
- Actions
- Environments
- Secrets