What does it mean to lose a community when it is online?

How can we think about the way we design and use the internet as both ephemeral and also can be valuable - and consider this in our approach and the way we operate towards designing and collecting material?

Javascript (vanilla)

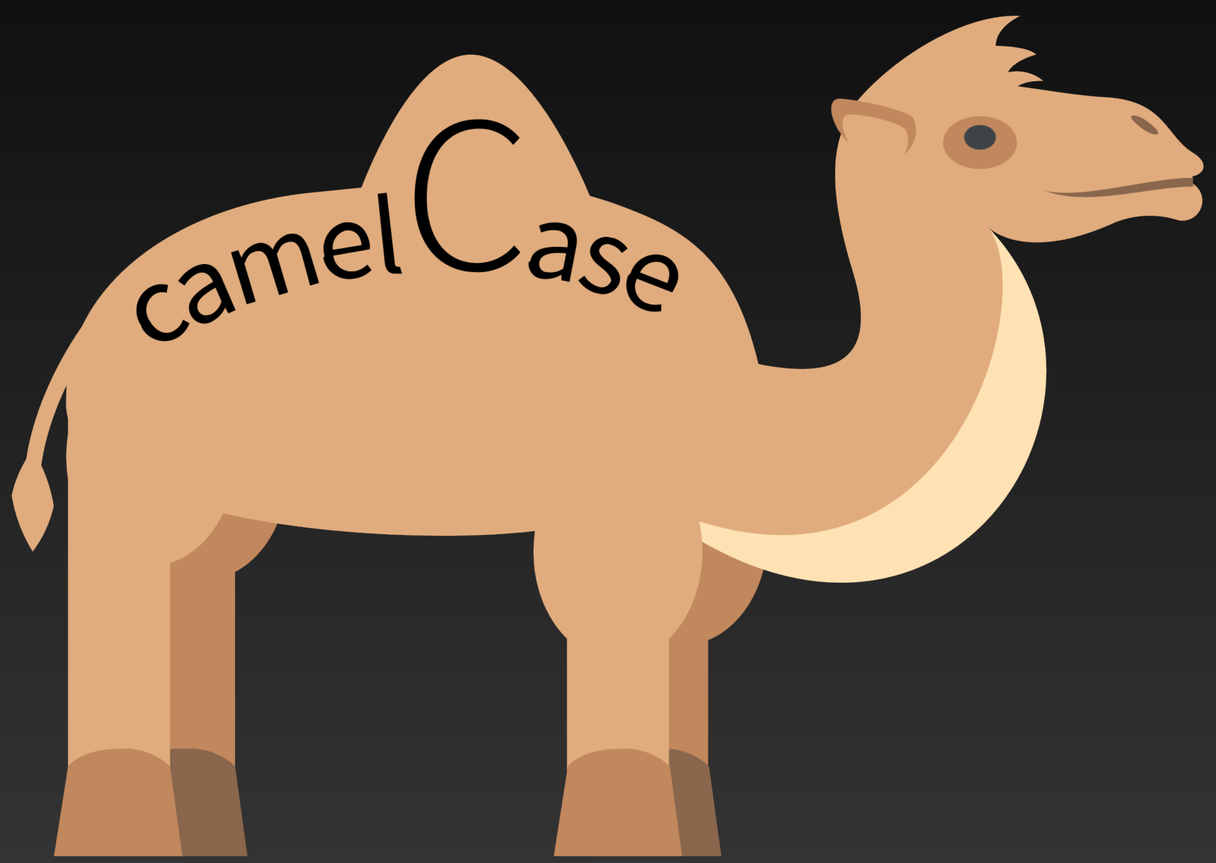
A mostly prototype based object oriented programming language - also used on the back end (node.js) so good for full stack development

In JavaScript order matters

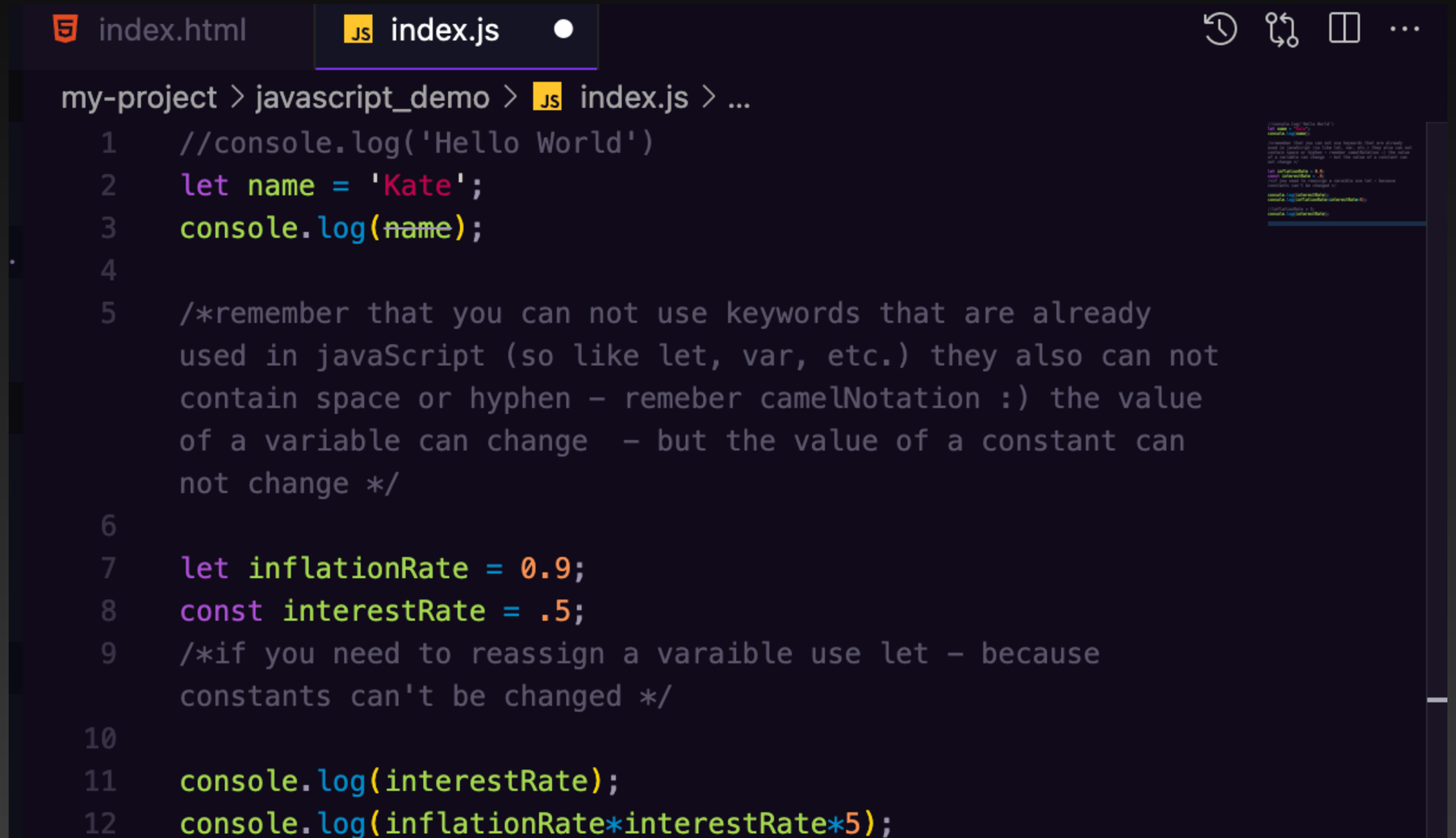
(though there are always new developments on getting the loading time down and parsed differently - so this is in flux)

- HTML basically would have things appear in the order you write them, CSS rarely requires you to think about order, in JavaScript, order really matters - the functions generally load from top to bottom.
- You have to be really careful that the objects (content) referenced in your code are named before you try to enact the things you want to have happen to them with JavaScript. (Think of it as a magic act - you can magically cut someone in 1/2 - but if you start cutting before you show the audience the someone is whole, they won't see the magic.)

camelCase



Single-line comments are marked with //
Block comments are enclosed between /* ... */.



JavaScript Style Guide
(Adopted from W3school)

1. Be consistent with coding conventions **for all your JavaScript projects.**

2. Variable Names
use **camelCase** for identifier names (variables and functions).

All names start with a **letter.**

```
firstName = "John";
lastName = "Doe";

fullPrice = price + (price * tax);
```

3. Spaces Around Operators
Always put spaces around operators (= + - * /), and after commas:
Examples:

```
let x = y + z;
const myArray = ["Volvo", "Saab", "Fiat"];
```

Statement Rules

- Always end a **simple statement** with a semicolon.

Examples:

```
const cars = ["Volvo", "Saab", "Fiat"];

const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

General rules for complex (compound) statements:

- Put the opening bracket at the end of the first line.
- Use one space before the opening bracket.
- Put the closing bracket on a new line, without leading spaces.
- **Do not** end a complex statement with a semicolon.

Functions:

```
function toCelsius(fahrenheit) {
  return (5 / 9) * (fahrenheit - 32);
}
```

Loops:

```
for (let i = 0; i < 5; i++) {
  x += i;
}
```

Conditionals:

```
if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

p5* File Edit Sketch Help English Log in or Si

Auto-refresh Grove cent

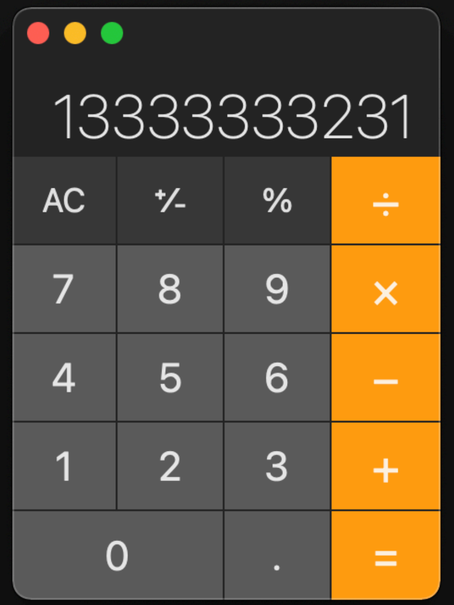
sketch.js

```
1  console.log (1+1);
2
3  let hugeNumber = 12345678910;
4
5  console.log (hugeNumber+hugeNumber);
6
7  let bigNumber = 987654321
8
9  console.log (hugeNumber+bigNumber)
```

Console Clear

```
2
24691357820
13333333231
```

Preview



A function statement *(how it come to be like a command)* consists of the function keyword, followed by:

- The name of the function.
- A list of parameters to the function, enclosed in parentheses and separated by commas.

Functions

- Everything in javascript works using functions
- Functions in JavaScript perform a task or calculate “values” when you “call them”
- Function names can contain letters, digits, underscores, and dollar signs

console.log(‘input’)

This is a built in function that even a web browser will recognize and which you can use to test code

Variables

A variable holds a value to be used in calculations, adding some kind of message, updating the appearance of a web-page element or creating a p5.js visual object, etc. Variable are major part of the syntax of JavaScript. They help us keep track of data in our code.

When declaring a variable, chose a meaningful variable name to help us remember what we are using it for in our code . You can not use dashes or spaces and certain javaScript specific language (like var, for, etc...)

```
var nameofvariable = 3212983759874357 + 4920932
```

Primitive Data Types:

- **Number:** Any number, including numbers with decimals
- **String:** Characters on your keyboard (letters, numbers, spaces, symbols, etc.) surrounded by single quotes: ‘ ‘ or double quotes “ “, (commonly single quotes.)
- **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.
- **Null:** This data type represents the intentional absence of a value, and is represented by the keyword null (without quotes).
- **Undefined: T**his 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. undefined means that a given value does not exist.
- **Symbol:** A newer feature to the language, symbols are unique identifiers, useful in more complex coding. No need to worry about these for now.
- **Object:** Collections of related data.

Using the p5.js console to learn how to declare a variable

A variable holds a value, the console can demonstrate this by showings the way the variable can be used to be assigned a value and then later using that variable's name to make a calculation. ('var' used to be more common - generally variables are now named using 'let')

```
let hugeNumber = 12345678910111213141 5
```

Data Types =

classifications we give to the different kinds of data that we use in programming.

Type	`typeof` return value	Object wrapper
Null	`"object"`	N/A
Undefined	`"undefined"`	N/A
Boolean	`"boolean"`	`Boolean`
Number	`"number"`	`Number`
BigInt	`"bigint"`	`BigInt`
String	`"string"`	`String`
Symbol	`"symbol"`	`Symbol`

Properties - Prototype based objects

- Similarities in properties help us recognize objects
- Property changing features inside an object are called “methods”
- Fill in the properties to their specifications
- Object can contain other objects - when an object is nested inside another object (and has its own methods and properties) - so an object is considered a method of the object that it is contained inside of.

Methods

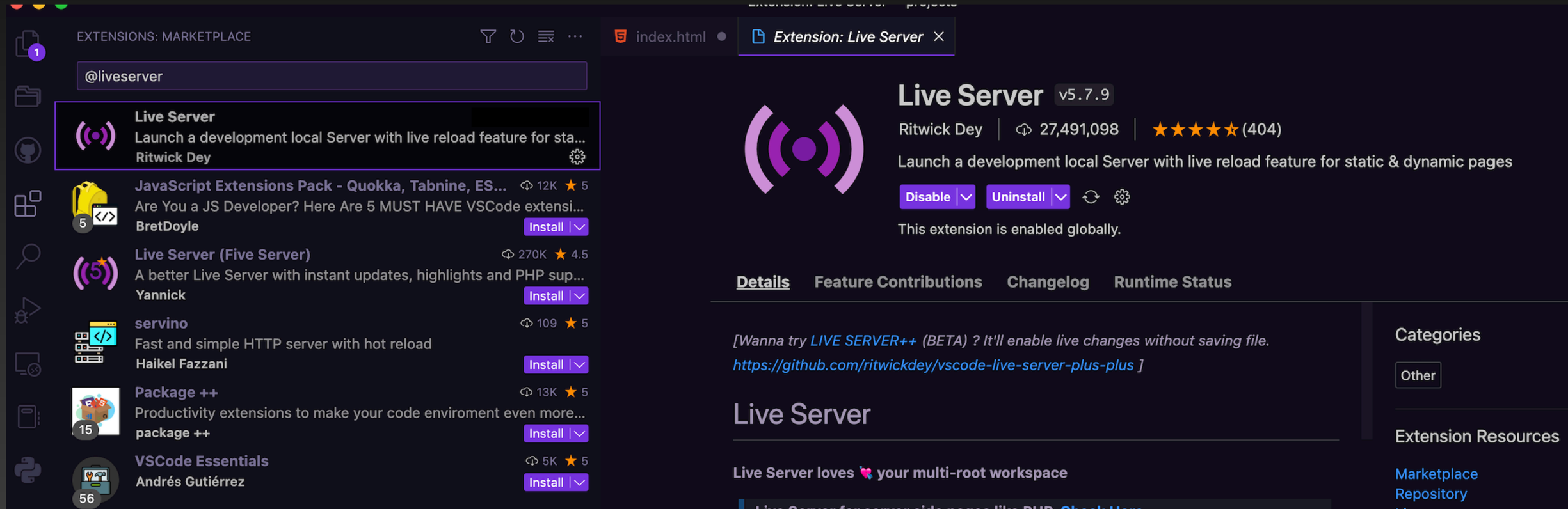
Methods are specific to different data types and allow us to perform actions specific to that data type. JavaScript provides a number of string methods.

We *call*, or use, these methods by appending an instance with:

- a period (the dot operator)
- the name of the method
- opening and closing parentheses

In visual studio code add the extension live server on Visual Studio Code

Open extensions, and install and then make sure
that you restart visual studio code



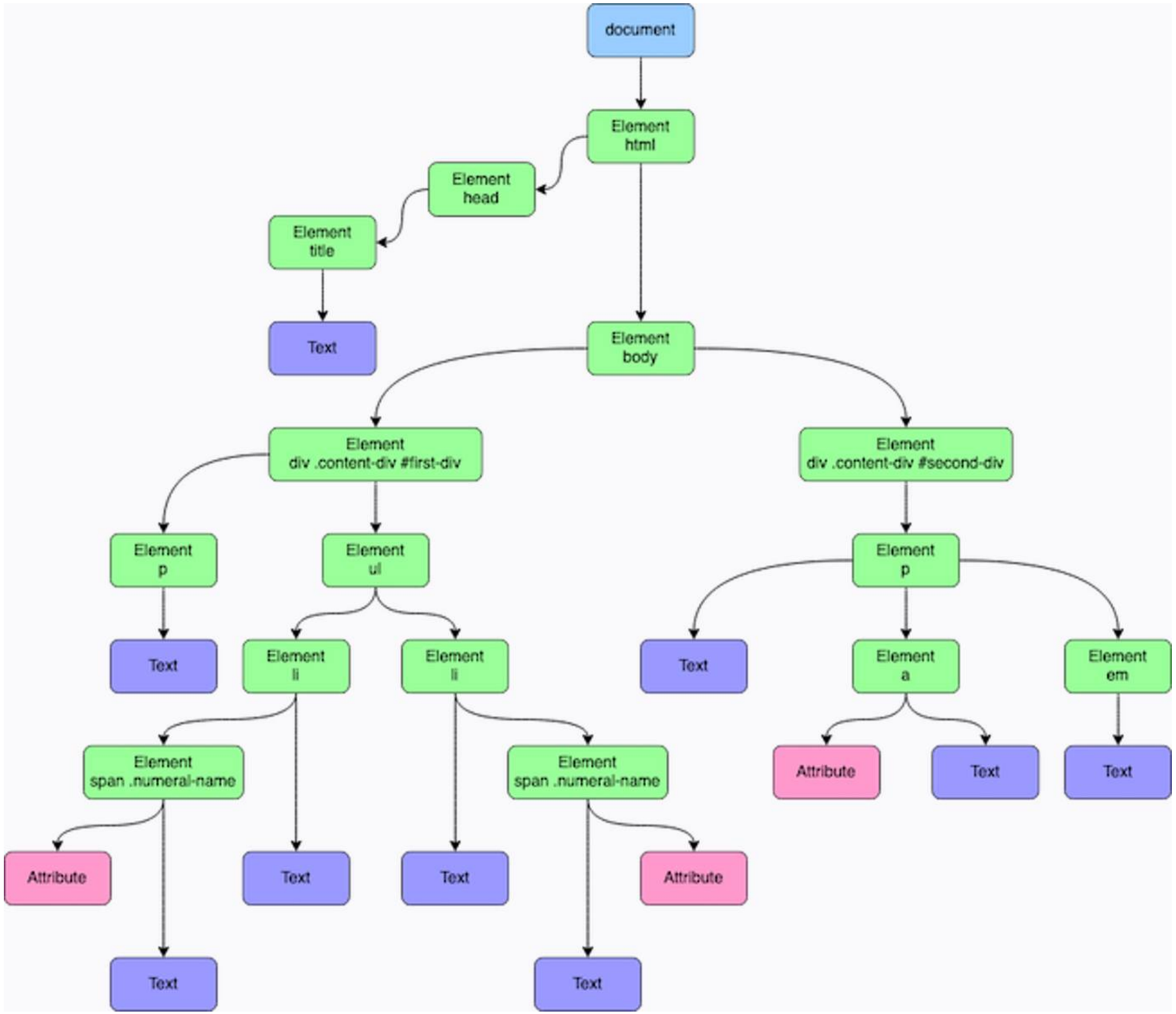
Research Presentations

Presentation Expectations:

Around 7-9 minutes - Upload presentation so that others can see slides/get links
Pick something to research that has a direct relationship conceptually or technically to your final project. Prepare a presentation that:

1. Explains how and why you researched the topic/code library/ element etc.
2. How it may be useful or relevant to others.
3. What you think you will do based off of this research.
4. Most importantly - try to contextualize the research in the history of the topics discussed in this class, modern web design and aesthetics generally.
5. The more clear and simple your explanation is, the more likely your peers will be able to benefit from your research. Consider this a moment to practice teaching something you are excited about .

DUE November 14th


document
Element
html
Element
head
Element
title
Text
Element
body
Element
div .content-div #first-div
Element
div .content-div #second-div
Element
p
Element
ul
Element
p
Text
Element
li
Element
li
Text
Element
a
Element
em
Element
span .numeral-name
Element
span .numeral-name
Attribute
Text
Attribute
Text
Text
Text
Attribute
Text
Text
Text
Text

RESPONSIVE WEB CHALLENGES

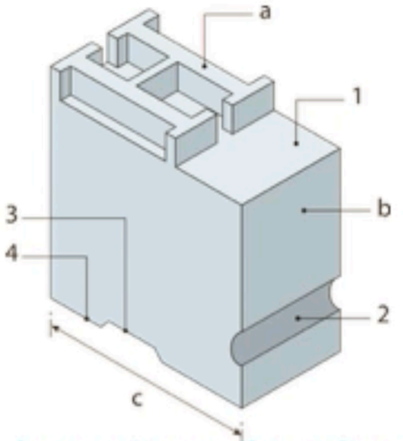
- Diversity of platforms
- Diversity of use cases
- Unknown screen sizes
- Unknown download speeds

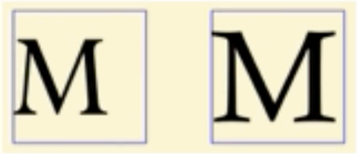
FLUID LAYOUTS: MAX-WIDTH AND MARGIN: AUTO

```
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
    width:500px;
    margin: auto;
    border: 3px solid #73AD21;
}

div.ex2 {
    max-width:500px;
    margin: auto;
    border: 3px solid #73AD21;
}
</style>
</head>
<body>

<div class="ex1">This div element has width: 500px;</div>
<br>

<div class="ex2">This div element has max-width: 500px;</div>

<p><strong>Tip:</strong> Drag the browser window to smaller than
500px wide, to see the difference between
the two divs!</p>

</body>
</html>
```

- Using max-width will limit the width of your page if it gets too big.
- Using min-width will keep it from getting too small.
- You can use pixels or percentages - percentages are the most flexible (but also be prepared for the results in extreme cases)
- Use margin: auto; to center items on the screen.

USING RELATIVE FONT SIZES W/ EMS





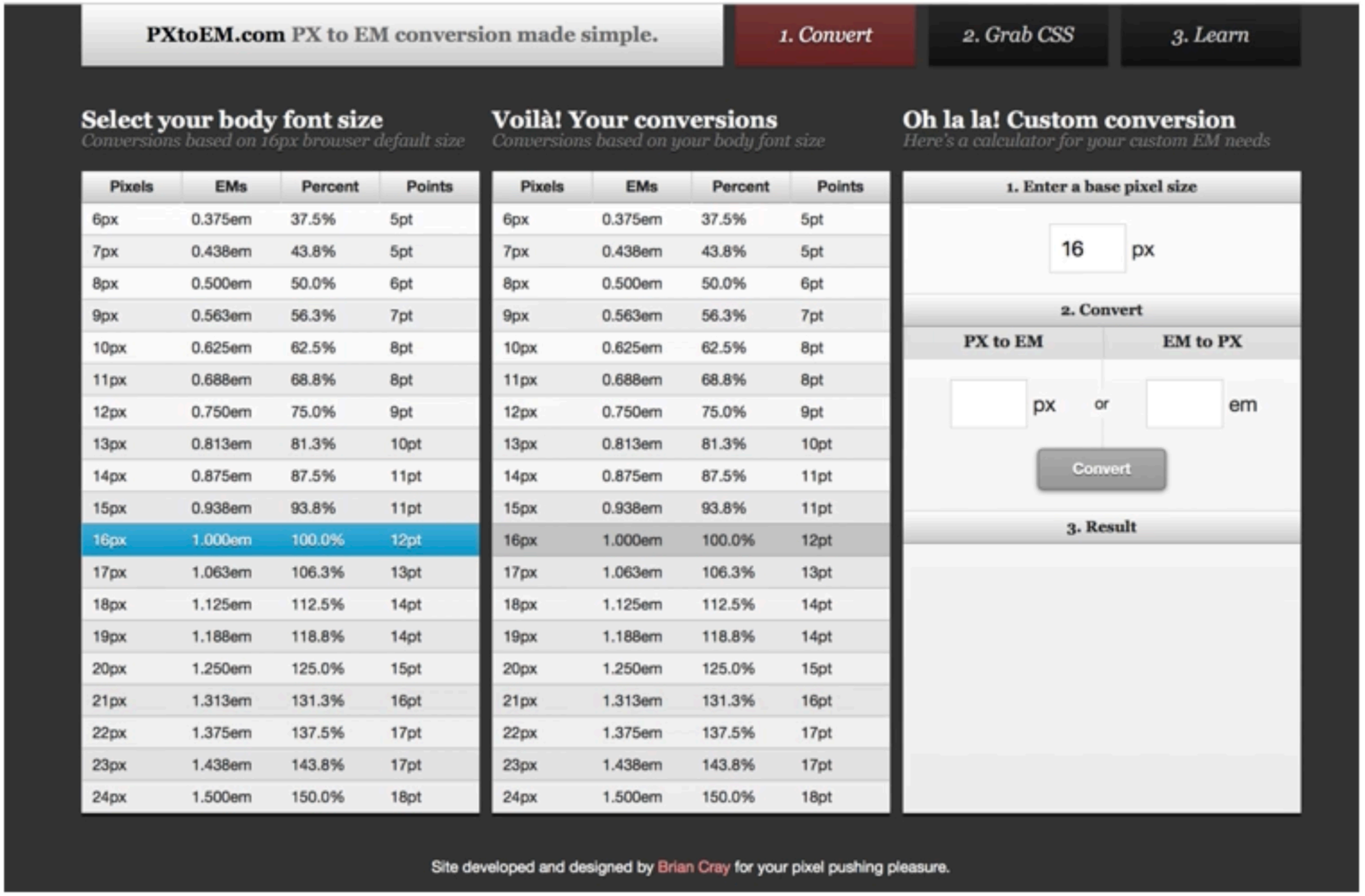
https://en.wikipedia.org/wiki/Em_(typography)

- Relative font sizes will scale based on our needs
- First, the base font. Typically we set this as 100%. Default on a browser is 16px. Let's make it 12px instead for our example by saying the base font is 75% of what the browser thinks it should be (default 16px). (Note: this is for illustration - 100% is a better idea for usability.)
- Next we can define the headers. For example, h1 could be 2x the base, or here it would be 24px.
- Perhaps we also set the line height relative to the font size
- NOTE: ems do not cascade - ems are relevant to the current element and percentages to the container, so % inside % will be smaller

```
body {
font-size: 75%;
}
h2 {
font-size: 2em;
}
p {
line-height: 1.5em;
}
```

remember CSS em <> HTML !

CALCULATING PIXELS TO EMS



PXtoEM.com PX to EM conversion made simple.

1. Convert | 2. Grab CSS | 3. Learn

Select your body font size
Conversions based on 16px browser default size

Pixels	EMs	Percent	Points
6px	0.375em	37.5%	5pt
7px	0.438em	43.8%	5pt
8px	0.500em	50.0%	6pt
9px	0.563em	56.3%	7pt
10px	0.625em	62.5%	8pt
11px	0.688em	68.8%	8pt
12px	0.750em	75.0%	9pt
13px	0.813em	81.3%	10pt
14px	0.875em	87.5%	11pt
15px	0.938em	93.8%	11pt
16px	1.000em	100.0%	12pt
17px	1.063em	106.3%	13pt
18px	1.125em	112.5%	14pt
19px	1.188em	118.8%	14pt
20px	1.250em	125.0%	15pt
21px	1.313em	131.3%	16pt
22px	1.375em	137.5%	17pt
23px	1.438em	143.8%	17pt
24px	1.500em	150.0%	18pt

Voilà! Your conversions
Conversions based on your body font size

Pixels	EMs	Percent	Points
6px	0.375em	37.5%	5pt
7px	0.438em	43.8%	5pt
8px	0.500em	50.0%	6pt
9px	0.563em	56.3%	7pt
10px	0.625em	62.5%	8pt
11px	0.688em	68.8%	8pt
12px	0.750em	75.0%	9pt
13px	0.813em	81.3%	10pt
14px	0.875em	87.5%	11pt
15px	0.938em	93.8%	11pt
16px	1.000em	100.0%	12pt
17px	1.063em	106.3%	13pt
18px	1.125em	112.5%	14pt
19px	1.188em	118.8%	14pt
20px	1.250em	125.0%	15pt
21px	1.313em	131.3%	16pt
22px	1.375em	137.5%	17pt
23px	1.438em	143.8%	17pt
24px	1.500em	150.0%	18pt

Oh la la! Custom conversion
Here's a calculator for your custom EM needs

1. Enter a base pixel size

16 px

2. Convert

PX to EM | EM to PX

px or em

Convert

3. Result

Site developed and designed by Brian Cray for your pixel pushing pleasure.

http://pxtoem.com

HOW TO CHOOSE WHAT TO OPTIMIZE OR ALTER

- How will the site be used?
  - Are the anticipated website versus mobile usage patterns different? (lookup versus data entry; targeted activity versus exploration)
  - Does any media rely on really fast connection speeds? Note that even fancy phones can get hung up w/o speedy wifi
  - Javascript or plug-in dependencies? (May not work on mobile)

INTRODUCING THE VIEWPORT

The viewport is the user's visible area of a web page.

The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

- New attribute for your meta tags - applies on devices; match width of site to width of screen - always add this

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Without the viewport meta tag

With the viewport meta tag

1. Do NOT use large fixed width elements - For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.

2. Do NOT let the content rely on a particular viewport width to render well - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

3. Use CSS media queries to apply different styling for small and large screens - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

https://www.w3schools.com/css/css_rwd_viewport.asp

EXAMPLE VIEWPORT SIZES

"We noticed that 360x640 is the most popular viewport size worldwide followed by 375x667. "

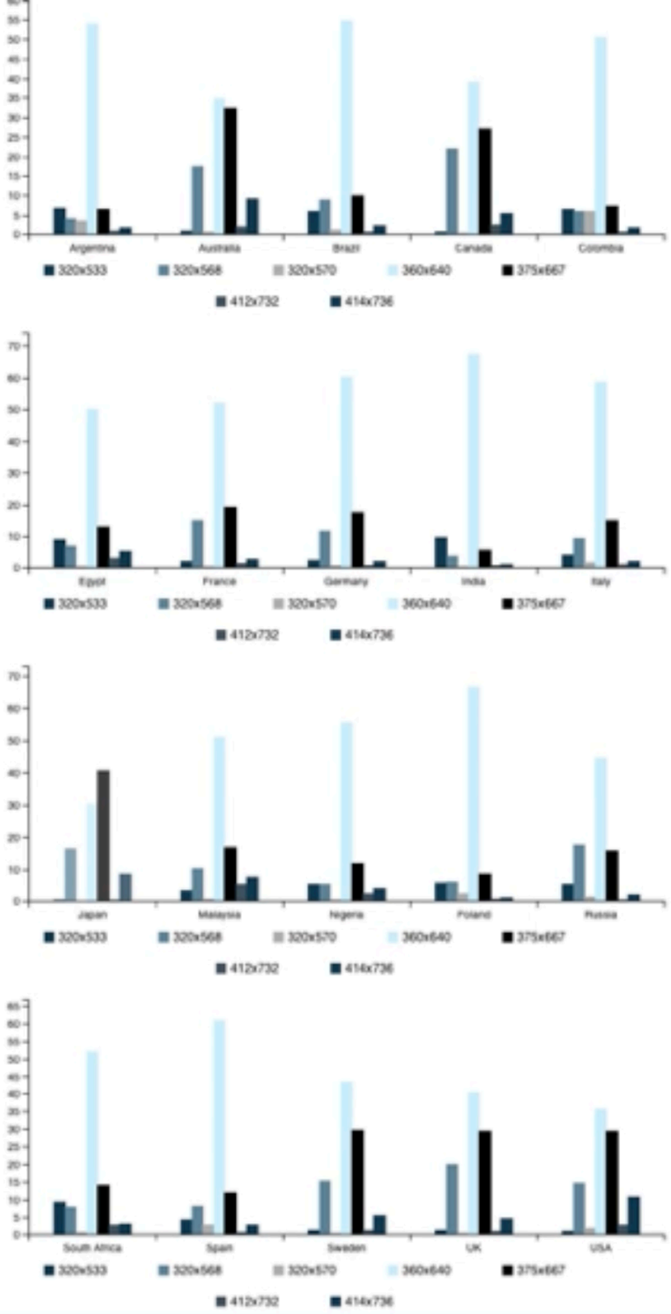
VIEWPORT SIZES

Filter

Device Name	Platform	OS Version	Portrait Width	Landscape Width	Release Date
Acer Iconia Tab A1-810	Android	4.2.2	768	1024	2013-05
Acer Iconia Tab A100	Android	4.0.3	800	1280	2011-04
Acer Iconia Tab A101	Android	3.2.1	600	1024	2011-05
Acer Iconia Tab A200	Android	4.0.3	800	1280	2012-01
Acer Iconia Tab A500	Android	4.0.3	648	1280	2011-04
Acer Iconia Tab A501	Android	3.2	800	1280	2011-04
ACER Liquid E2	Android	4.2.1	360	640	2013-05
Ainol Novo 7 Elf 2	Android	4.0.3	496	1024	2012-06
Alcatel (Vodafone) Smart Mini 875	Android	4.1.1	320	480	2013-07

http://viewportsizes.com

https://www.smashingmagazine.com/2012/08/towards-retina-web/



United States

	Screen resolution	Share
1.	750x1334	30.08%
2.	1080x1920	27.72%
3.	1440x2960	10.60%
4.	1440x2560	9.13%
5.	720x1280	7.08%

https://deviceatlas.com/blog/most-used-smartphone-screen-resolutions#us

https://deviceatlas.com/blog/mobile-viewport-size-statistics-2017

MEDIA QUERIES

- Look for device capabilities
  - width and height of the viewport
  - width and height of the device
  - orientation (is the tablet/phone in landscape or portrait mode?)
  - resolution
- True or False results
- Conditional CSS OR
- Designate a whole style sheet based on the results

```
@media not|only mediatype and (expressions) {

    CSS-Code;

}
```

```
@media screen and (min-width:
600px) {

  body {

    font-size: 1em;

  }

}
```

```
<head>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="2column-styles.css" media="screen and
    (min-width:780px)">
</head>
```

MEDIA QUERIES

Table 18-1. Media features you can evaluate with media queries

Feature	Description
`width`	The width of the display area (viewport).
`height`	The height of the display area (viewport).
`device-width`	The width of the devices rendering surface (the whole screen).
`device-height`	The height of the devices rendering surface (the whole screen).
`orientation`	Whether the device is in portrait or landscape orientation. (Does not accept min-/max- prefixes.)
`aspect-ratio`	Ratio of the viewport's width divided by height (width/height).
`device-aspect-ratio`	Ratio of the whole screen's (rendering surface) width to height.
`color`	The bit depth of the display; for example, `color: 8` tests for whether the device has at least 8-bit color.
`color-index`	The number of colors in the color lookup table.
`monochrome`	The number of bits per pixel in a monochrome device.
`resolution`	The density of pixels in the device. This is increasingly relevant for detecting high-resolution displays.
`scan`	Whether a tv media type uses progressive or interlace scanning. (Does not accept min-/max- prefixes.)
`grid`	Whether the device uses a grid-based display, such as a fixed-width font. (Does not accept min-/max- prefixes.)

https://www.smashingmagazine.com/2010/07/how-to-use-css3-media-queries-to-create-a-mobile-version-of-your-website/

http://mediaqueriestest.com

MEDIA QUERIES

Table 18-1. Media features you can evaluate with media queries

Feature	Description
`width`	The width of the display area (viewport).
`height`	The height of the display area (viewport).
`device-width`	The width of the devices rendering surface (the whole screen).
`device-height`	The height of the devices rendering surface (the whole screen).
`orientation`	Whether the device is in portrait or landscape orientation. (Does not accept min-/max- prefixes.)
`aspect-ratio`	Ratio of the viewport's width divided by height (width/height).
`device-aspect-ratio`	Ratio of the whole screen's (rendering surface) width to height.
`color`	The bit depth of the display; for example, `color: 8` tests for whether the device has at least 8-bit color.
`color-index`	The number of colors in the color lookup table.
`monochrome`	The number of bits per pixel in a monochrome device.
`resolution`	The density of pixels in the device. This is increasingly relevant for detecting high-resolution displays.
`scan`	Whether a tv media type uses progressive or interlace scanning. (Does not accept min-/max- prefixes.)
`grid`	Whether the device uses a grid-based display, such as a fixed-width font. (Does not accept min-/max- prefixes.)

https://www.smashingmagazine.com/2010/07/how-to-use-css3-media-queries-to-create-a-mobile-version-of-your-website/

http://mediaqueriestest.com

UNIT 5: NEXT STEPS - RESPONSIVE WEB

MEDIA QUERY EXAMPLE: CONDITIONAL CSS

Run »

>= 480px

Result Size: 573 x 842

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<style>
.wrapper {overflow: auto;}

#main {margin-left: 4px;}

#leftsidebar {
    float: none;
    width: auto;
}

#menulist {
    margin: 0;
    padding: 0;
}

.menuitem {
    background: #CDF0F6;
    border: 1px solid #d4d4d4;
    border-radius: 4px;
    list-style-type: none;
    margin: 4px;
    padding: 2px;
}

@media screen and (min-width: 480px) {
    #leftsidebar {width: 200px; float: left;}
    #main {margin-left: 216px;}
}
</style>
</head>
```

default status - resizes to fit screen on top

if true, then... else default status

- Menu-item 1
- Menu-item 2
- Menu-item 3
- Menu-item 4
- Menu-item 5

Resize the browser window to see the effect!

This example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider. If the viewport is less than 480 pixels, the menu will be on top of the content.

below 480px

- Menu-item 1
- Menu-item 2
- Menu-item 3
- Menu-item 4
- Menu-item 5

Resize the browser window to see the effect!

This example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider. If the viewport is less than 480 pixels, the menu will be on top of the content.

UNIT 5: NEXT STEPS - RESPONSIVE WEB

MEDIA QUERY EXAMPLE - COLOR CHANGES

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<style>
body {
    background-color: lightgreen;
}

@media only screen and (max-width: 500px) {
    body {
        background-color: lightblue;
    }
}
</style>
</head>
<body>

<p>Resize the browser window. When the width of this
less than 500 pixels, the background-color is "lightb
otherwise it is "lightgreen".</p>

</body>
</html>
```

Resize the browser window. When the width of this document is less than 500 pixels, the background-color is "lightblue", otherwise it is "lightgreen".

Dynamic changes!

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-
width, initial-scale=1.0">
<style>
body {
    background-color: lightgreen;
}

@media only screen and (max-width: 500px) {
    body {
        background-color: lightblue;
    }
}
</style>
</head>
<body>

<p>Resize the browser window. When the width of
this document is less than 500 pixels, the
background-color is "lightblue", otherwise it is
"lightgreen".</p>

</body>
</html>
```

Resize the browser window. When the width of this document is less than 500 pixels, the background-color is "lightblue", otherwise it is "lightgreen".

https://www.w3schools.com/css/tryit.asp?filename=tryresponsive_mediaquery

UNIT 5: NEXT STEPS - RESPONSIVE WEB

MOBILE FIRST DESIGN APPROACH

- Build up from mobile foundation unless your site is really desktop-focused - this is dependent upon the purpose of the site
- Think about the baseline design first, and then the flourishes based on screen real estate- use things like min-width to decide what to show

Always Design for Mobile First

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices).

This means that we must make some changes in our CSS.

Instead of changing styles when the width gets *smaller* than 768px, we should change the design when the width gets *larger* than 768px. This will make our design Mobile First:

Example

```
/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}
@media only screen and (min-width: 768px) {
    /* For desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    .col-5 {width: 41.66%;}
    .col-6 {width: 50%;}
    .col-7 {width: 58.33%;}
    .col-8 {width: 66.66%;}
    .col-9 {width: 75%;}
    .col-10 {width: 83.33%;}
    .col-11 {width: 91.66%;}
    .col-12 {width: 100%;}
}
```

Try it Yourself »

https://www.w3schools.com/css/css_rwd_mediaqueries.asp

UNIT 5: NEXT STEPS - RESPONSIVE WEB

MEDIA QUERIES AND IMAGE SELECTION

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<style>
/* For width smaller than 400px: */
body {
    background-repeat: no-repeat;
    background-image: url('img_smallflower.jpg');
}

/* For width 400px and larger: */
@media only screen and (min-width: 400px) {
    body {
        background-image: url('img_flowers.jpg');
    }
}
</style>
</head>
<body>

<p style="margin-top:360px;">Resize the browser widt
background image will change at 400px.</p>

</body>
</html>
```

conditional is minimum size to display larger image

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport"
content="width=device-width, initial-
scale=1.0">
<style>
/* For width smaller than 400px: */
body {
    background-repeat: no-repeat;
    background-image:
url('img_smallflower.jpg');
}

/* For width 400px and larger: */
@media only screen and (min-width:
400px) {
    body {
        background-image:
url('img_flowers.jpg');
    }
}
</style>
</head>
<body>

<p style="margin-top:360px;">Resize the
browser width and the background image
will change at 400px.</p>

</body>
```

default

mobile first designs?

Resize the browser width and the background image will change at 400px.

Note:

```
background-size: contain;
background-size: cover;
background-size: 100% 100%;
```

UNIT 5: NEXT STEPS - RESPONSIVE WEB

IMAGE SIZING

WIDTH VS. MAX-WIDTH AND MIN-WIDTH

```
img {
    width: 100%;
    height: auto;
}
```

```
img {
    max-width: 100%;
    height: auto;
}
```

min-width overrides max-width overrides width

The `max-width` property in CSS is used to set the **maximum width** of a specified element. The `max-width` property overrides the `width` property, but `min-width` will always override `max-width` whether followed before or after `width` in your declaration. Authors may use any of the length values as long as they are a positive value.

CSS

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<style>
img {
    max-width: 100%;
    height: auto;
}
</style>
</head>
<body>

<img src="img_chania.jpg" width="460" height="345">
<p>Resize the browser window to see how the image
wil scale when the width is less than 460px.</p>

</body>
</html>
```

Resize the browser window to see how the image wil scale when the width is less than 460px.

max-width, min-width **not** inherited

use these to control their behavior within, and effect on, other elements

min-width - good so something doesn't get too tiny

```
<div style="100%">

    <div style="width:30%;min-width:
50px;"></div>

</div>
```

https://www.w3schools.com/css/css_rwd_images.asp

https://css-tricks.com/almanac/properties/m/max-width/

ADDING VIDEO

CSS

```
video {

    width: 100%;

    height: auto;

}
```

resizes to fit 100%; could also use max-width here

HTML

```
<div class="col-6 col-s-9">
  <h1>The City</h1>
  <p>Chania is the capital of the Chania region on the island
of Crete. The city can be divided in two parts, the old town
and the modern city.</p>
  <video width="400" controls>
    <source src="mov_bbb.mp4" type="video/mp4">
    <source src="mov_bbb.ogg" type="video/ogg">
    Your browser does not support HTML5 video.
  </video>
</div>
```

ABOUT THOSE IDS...

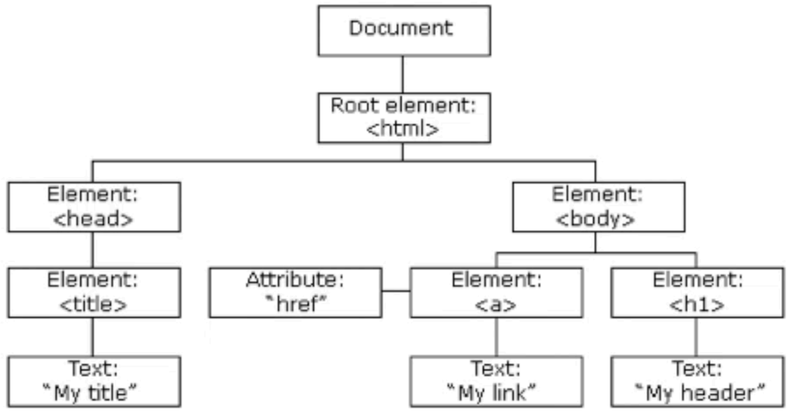
- in CSS CLASS and ID seemed almost inter-changeable (except you could use ID only once, and they outranked CSS for styling)
- in JavaScript you can identify elements in your page by ID and then *do things to them*

```
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

USING THE DOM MODEL TO TARGET PARTS OF THE PAGE

The HTML DOM document object is the owner of all other objects in your web page.

- document is the top level of the HTML page
- next item is the method
- third item is the parameter to "search" on - ID, name of the tag, class…



Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Declare all classes in a separate file to make sure the classes are declared before you create a constant. You can make as many object after that. You then state the import at the top of the script file.

Class gives you more functions.

CodeAcademy Review:

- Data is printed, or logged, to the console, a panel that displays messages, with console.log().
- We can write single-line comments with // and multi-line comments between /* and */.
- There are 7 fundamental data types in JavaScript: strings, numbers, booleans, null, undefined, symbol, and object.
- Numbers are any number without quotes: 23.8879
- Strings are characters wrapped in single or double quotes: 'Sample String'
- The built-in arithmetic operators include +, -, *, /, and %.
- Objects, including instances of data types, can have properties, stored information. The properties are denoted with a . after the name of the object, for example: 'Hello'.length.
- Objects, including instances of data types, can have methods which perform actions. Methods are called by appending the object or instance with a period, the method name, and parentheses. For example: ‘hello’.toUpperCase().
- We can access properties and methods by using the ., dot operator.
- Built-in objects, including Math, are collections of methods and properties that JavaScript provides

General rules for naming Variables:

Variable names cannot start with numbers.
Variable names are case sensitive, so myName and myname would be different variables.
It is bad practice to create two variables that have the same name using different cases.
Variable names cannot be the same as keywords.

Variables:

We can declare a variable without assigning the variable a value. In such a case, the variable will be automatically initialized with a value of undefined:

You can store any value in a *const* variable. The way you declare a *const* variable and assign a value to it follows the same structure as *let* and *var*. However, a *const* variable cannot be reassigned because it is constant. If you try to reassign a const variable, you'll get a TypeError.

You can store any value in a *const* variable. The way you declare a *const* variable and assign a value to it follows the same structure as *let* and *var*. However, a *const* variable cannot be reassigned because it is constant. If you try to reassign a *const* variable, you'll get a TypeError. Also Constant variables must be assigned a value when declared. If you try to declare a *const* variable without a value, you'll get a SyntaxError.

Mathematical Assignment Operators

```
let x = 20;
x -= 5; // Can be written as x = x - 5
console.log(x); // Output: 15

let y = 50;
y *= 2; // Can be written as y = y * 2
console.log(y); // Output: 100

let z = 8;
z /= 2; // Can be written as z = z / 2
console.log(z); // Output: 4
```

Also!

var++; = (var + 1)

var - - ; = (var - 1)

String Interpolation

In ES6 vs. if JavaScript, we can insert, or interpolate, variables into strings using template literals. Below a template literal is used to log strings together:

```
let myName= 'Kate';
console.log(`People call me ${myName}.`);
// Output: People call me Kate.
```

Template literals are wrapped by backticks ` (located on the top of your keyboard, left of the 1 key). Inside the template literal, you'll see a placeholder, ${myPet}. The value of myPet is inserted into the template literal.

One of the biggest benefits to using template literals is the readability of the code. Using template literals, you can more easily tell what the new string will be. You also don't have to worry about escaping double quotes or single quotes.

Variable Review:

- Variables hold reusable data in a program and associate it with a name.
- Variables are stored in memory.
- Variables are stored in memory.
- *let* is the preferred way to declare a variable when it can be reassigned, and *const* is the preferred way to declare a variable with a constant value.
- Variables that have not been initialized store the primitive data type undefined.
- The built-in arithmetic operators include +, -, *, /, and %.
- Mathematical assignment operators make it easy to calculate a new value and assign it to the same variable. +=, -=, *=, /=
- The + operator is used to concatenate strings including string values held in variables.
- In ES6, template literals use backticks ` and ${} to interpolate values into a string.
- The typeof keyword returns the data type (as a string) of a value

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: ===
Is not equal to: !==

identity operator (===)

Conditionals Review:

- An if statement checks a condition and will execute a task if that condition evaluates to true.
- if...else statements make binary decisions and execute different code blocks based on a provided condition.
- We can add more conditions using else if statements.
- *Comparison operators, including <, >, <=, >=, ===, and !== can compare two value*
- The logical and operator, &&, or "and", checks if both provided expressions are truthy.
- The logical operator ||, or "or", checks if either provided expression is truthy.
- The bang operator, !, switches the truthiness and falsiness of a value.
- The ternary operator is shorthand to simplify concise if...else statements.
- A switch statement can be used to simplify the process of writing multiple else if statements. The break keyword stops the remaining cases from being checked and executed in a switch statement.