Hello!

April Medinger

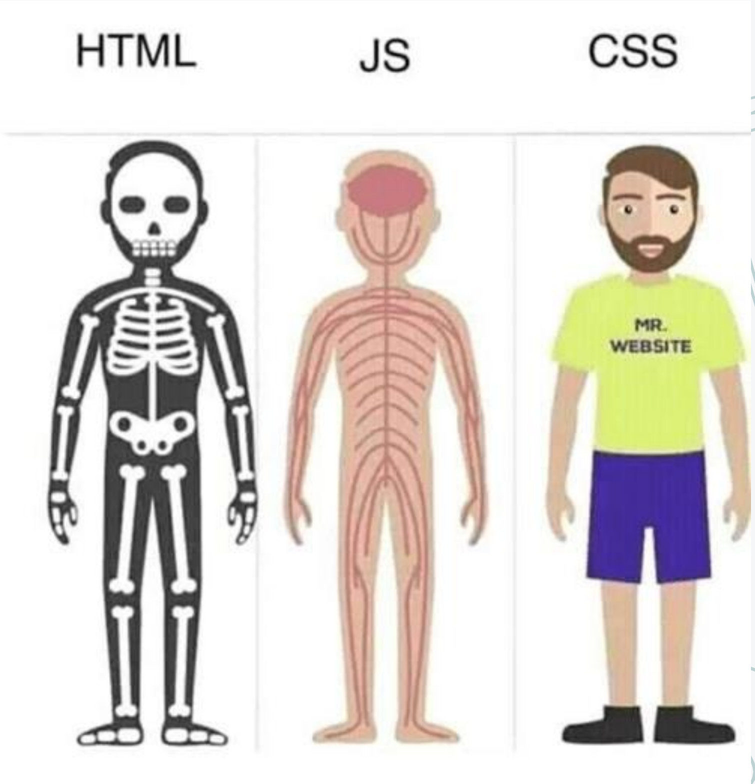
UI Designer / Developer - HarrisData

and I'm Houston!

Anatomy of Web Apps

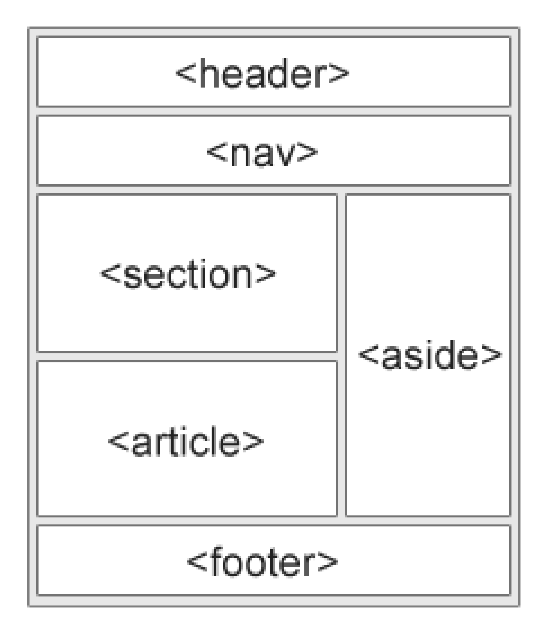
Anatomy overview

HTML	CSS	JavaScript
Bones	Tissue	Nervous System
Content	Design / Style	Interactivity

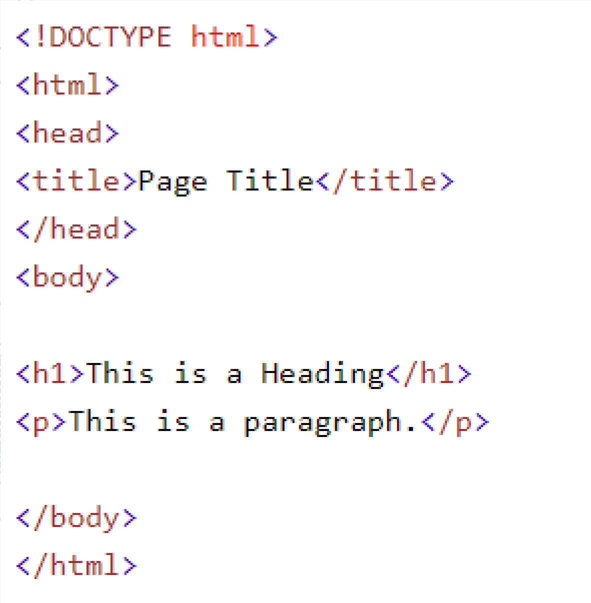


(HTML) HyperText Markup Language

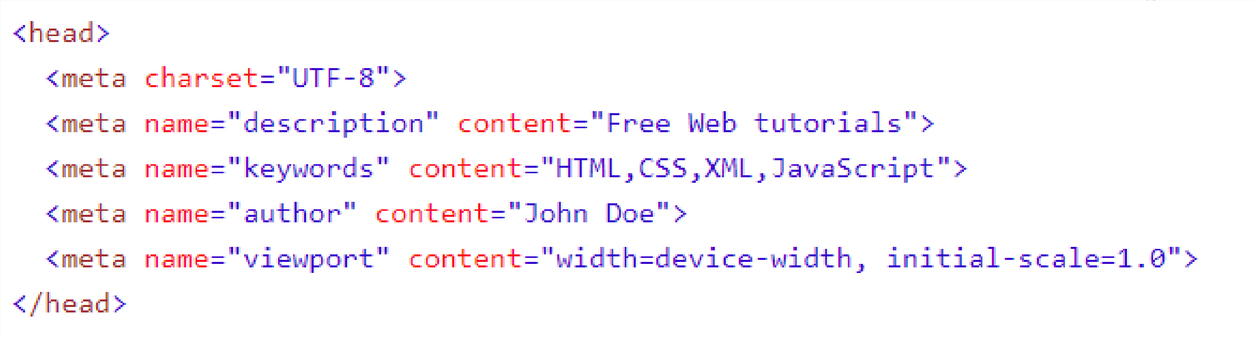
- ✓ Not a programming language
- ✓ Semantic meaningful markup (HTML5)
- ✓ Identifies structure of document
- ✓ Metadata about document
  - ○ Title, author, keywords.. Etc.
- ✓ More support for media (HTML5)
  - ○ Video and Audio
- ✓ Better integration with CSS and JS (HTML5)
  - ○ Eliminates need for Plugins!
  - ○ Flash and Applets (thank goodness)



(HTML) HyperText Markup Language



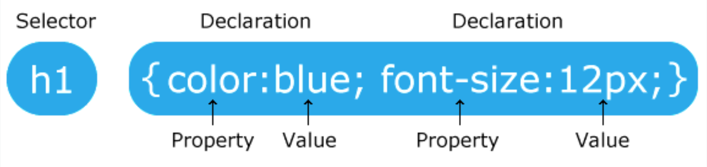
```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

(CSS) Cascading Style Sheets

- ✓ Specifies the document’s style
  - ○ Layout, colors, font, etc.

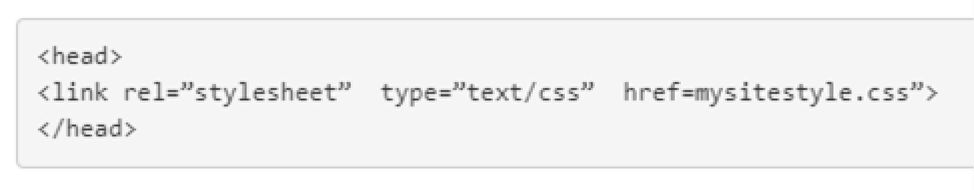


Selector Declaration Declaration

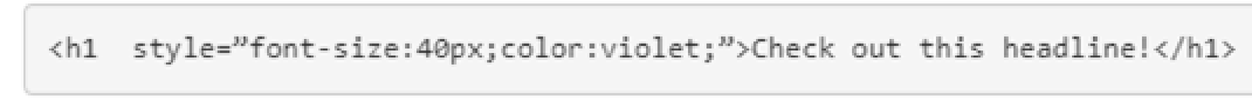
h1 { color:blue; font-size:12px; }

Property Value Property Value

- ✓ Best practices: Keep content separate from design
  - ○ External styles



```
<head>
<link rel=”stylesheet”  type=”text/css”  href=mysitestyle.css”>
</head>
```

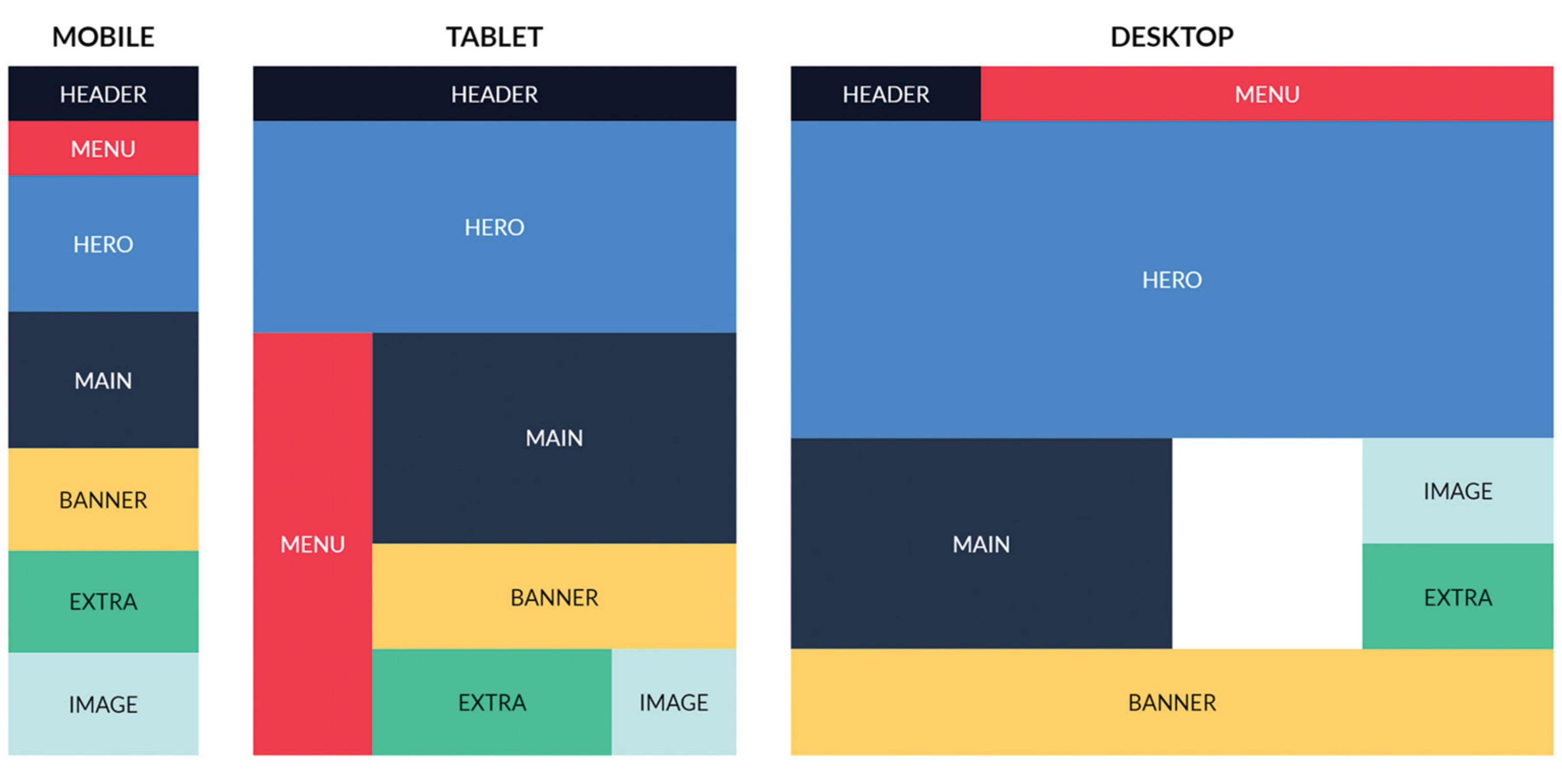


```
<h1  style=”font-size:40px;color:violet;”>Check out this headline!</h1>
```

Responsive Web Design

- ✓ Practice of building a website suitable to work on every device and every screen size, no matter how large or small, mobile or desktop
- ✓ CSS Grid Layout
- ✓ @media rules
  - ○ Detects width, height, orientation, resolution
- ✓ Mobile detection libraries (JS, PHP, etc..)

Responsive Web Design

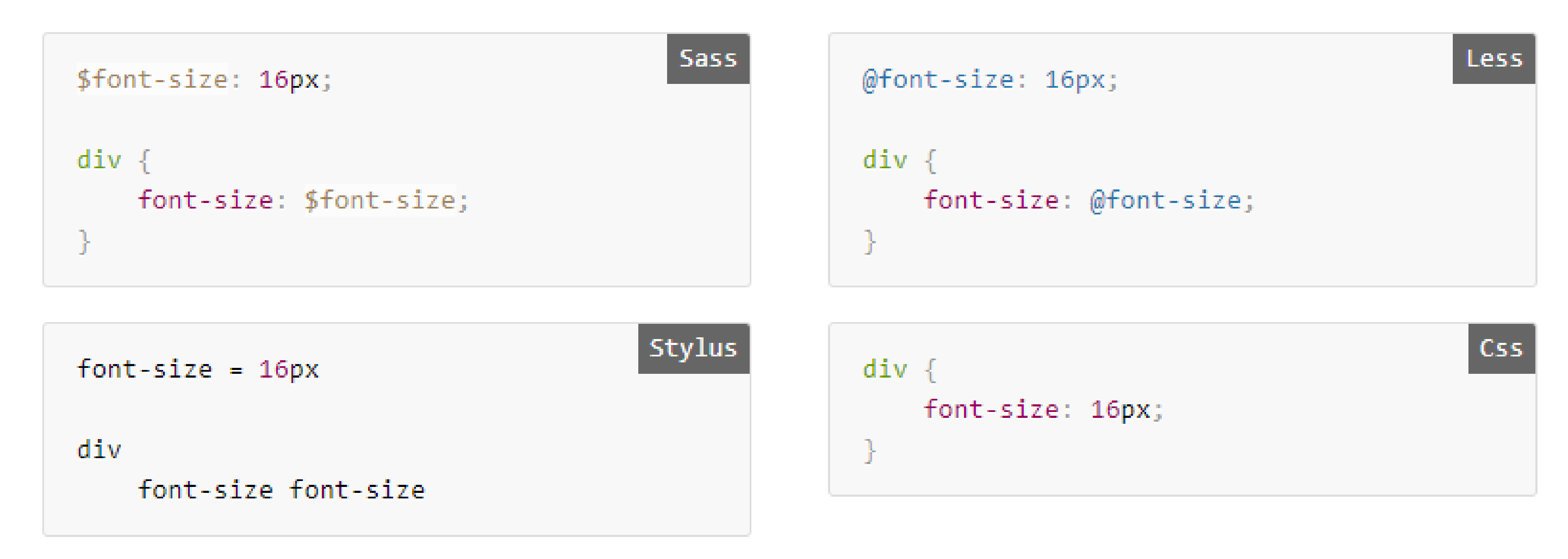


CSS Preprocessors

- ✓ Programs generate CSS from unique syntax
- ✓ Make the CSS structure more readable and easier to maintain
- ✓ Adds features not available in pure CSS
    - ○ Variables
    - ○ Functions and mixins
    - ○ Nesting
    - ○ Extends

{less}
Sass
stylus

CSS Preprocessors - Variables



Sass

```scss
$font-size: 16px;

div {
    font-size: $font-size;
}
```

Less

```less
@font-size: 16px;

div {
    font-size: @font-size;
}
```

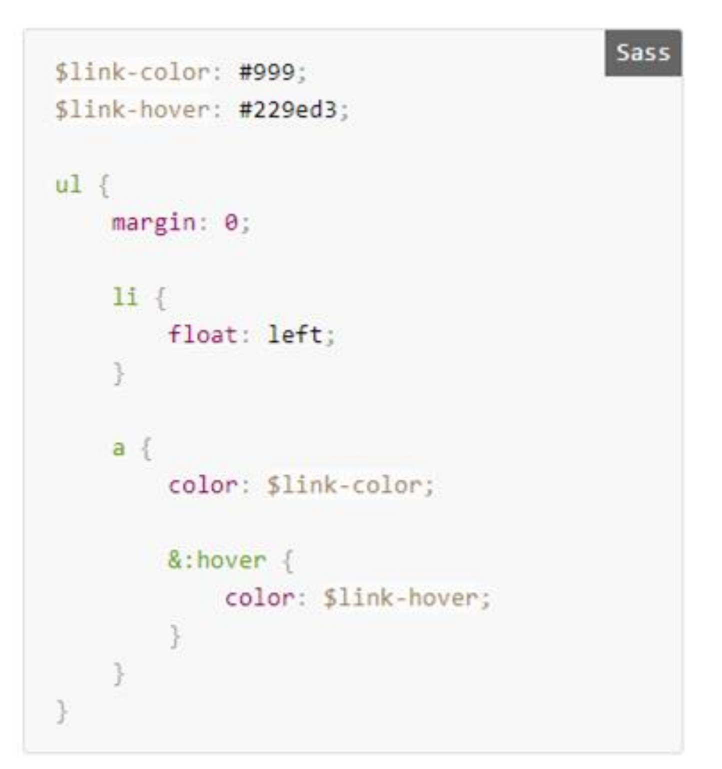
Stylus

```stylus
font-size = 16px

div
    font-size font-size
```

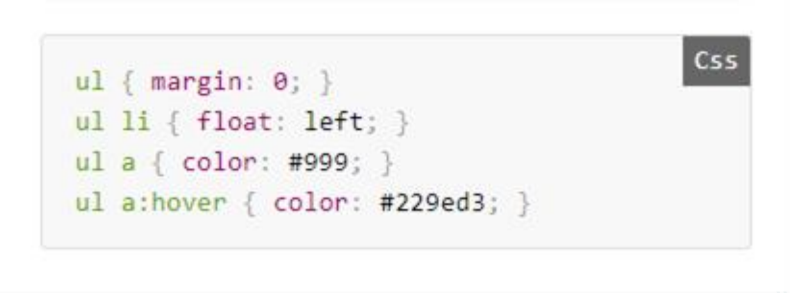
Css

```css
div {
    font-size: 16px;
}
```

CSS Preprocessors - Nesting

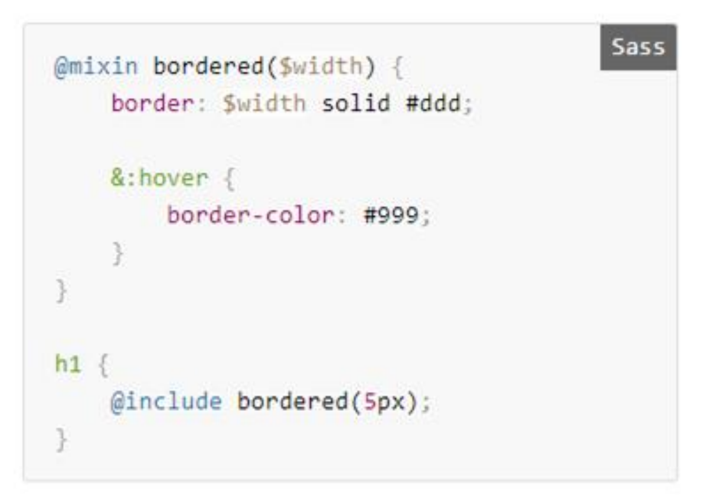


```
$link-color: #999;
$link-hover: #229ed3;

ul {
    margin: 0;

    li {
        float: left;
    }

    a {
        color: $link-color;

        &:hover {
            color: $link-hover;
        }
    }
}
```

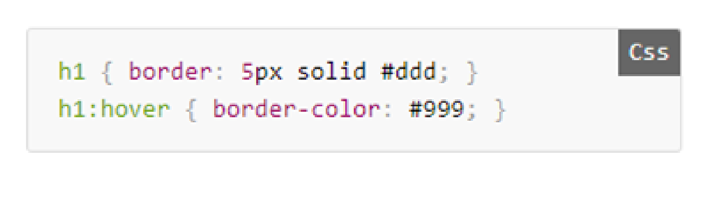


```
ul { margin: 0; }
ul li { float: left; }
ul a { color: #999; }
ul a:hover { color: #229ed3; }
```

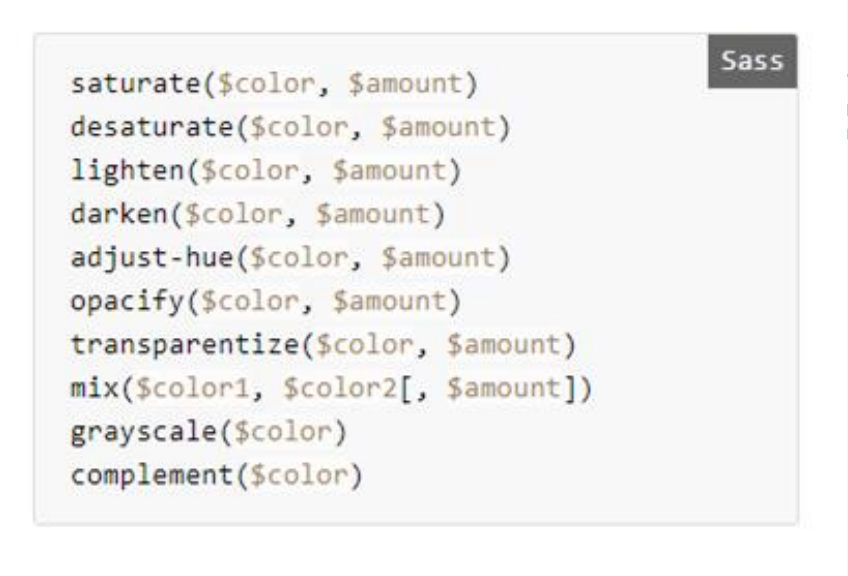
CSS Preprocessors - Mixins / Functions



```
@mixin bordered($width) {
    border: $width solid #ddd;

    &:hover {
        border-color: #999;
    }
}

h1 {
    @include bordered(5px);
}
```

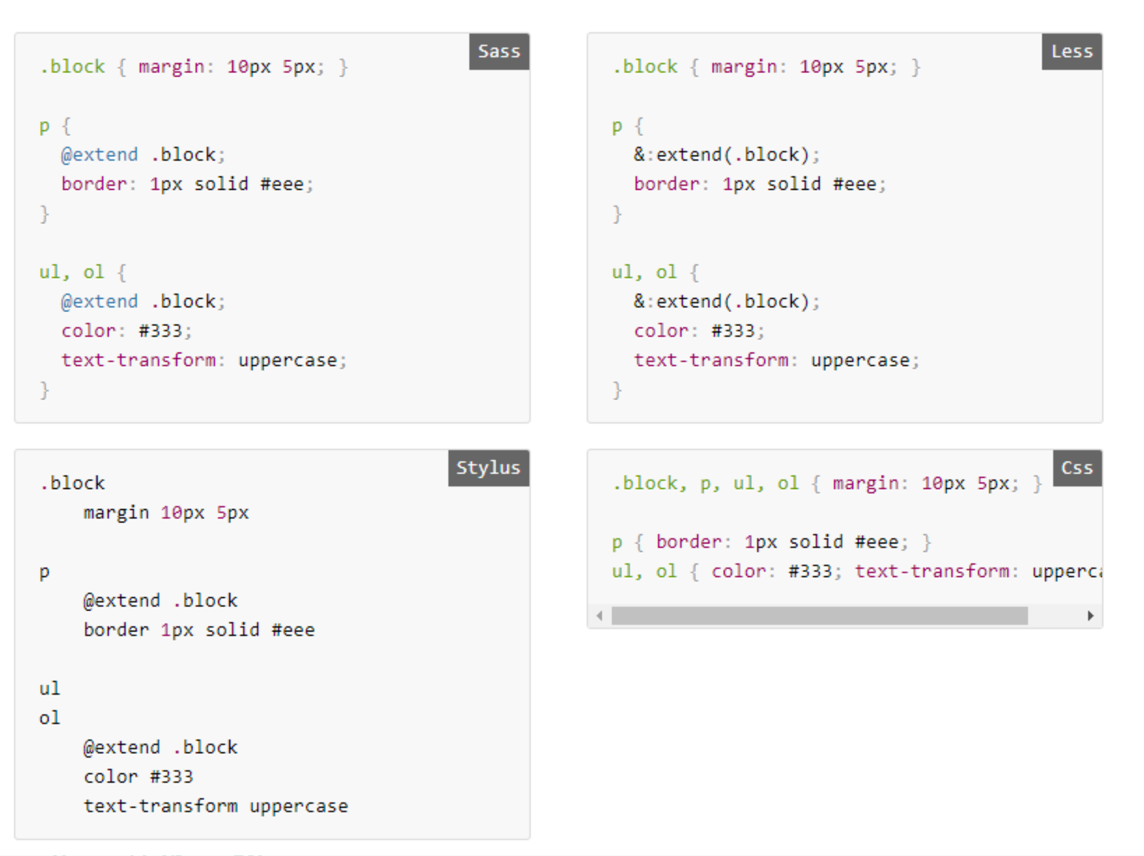


```
h1 { border: 5px solid #ddd; }
h1:hover { border-color: #999; }
```



```
saturate($color, $amount)
desaturate($color, $amount)
lighten($color, $amount)
darken($color, $amount)
adjust-hue($color, $amount)
opacify($color, $amount)
transparentize($color, $amount)
mix($color1, $color2[, $amount])
grayscale($color)
complement($color)
```

CSS Preprocessors - Extends



Sass

```
.block { margin: 10px 5px; }

p {
  @extend .block;
  border: 1px solid #eee;
}

ul, ol {
  @extend .block;
  color: #333;
  text-transform: uppercase;
}
```

Less

```
.block { margin: 10px 5px; }

p {
  &:extend(.block);
  border: 1px solid #eee;
}

ul, ol {
  &:extend(.block);
  color: #333;
  text-transform: uppercase;
}
```

Stylus

```
.block
    margin 10px 5px

p
    @extend .block
    border 1px solid #eee

ul
ol
    @extend .block
    color #333
    text-transform uppercase
```

Css

```
.block, p, ul, ol { margin: 10px 5px; }

p { border: 1px solid #eee; }
ul, ol { color: #333; text-transform: upperca
```

(JS) JavaScript

JavaScript

- ✓ Been around for almost 25 years
- ✓ Dominant programming language in web development
- ✓ Typically used as Client-side code
- ✓ Provides animation to static websites
- ✓ Creates dynamically updating content

BACKBONE.JS

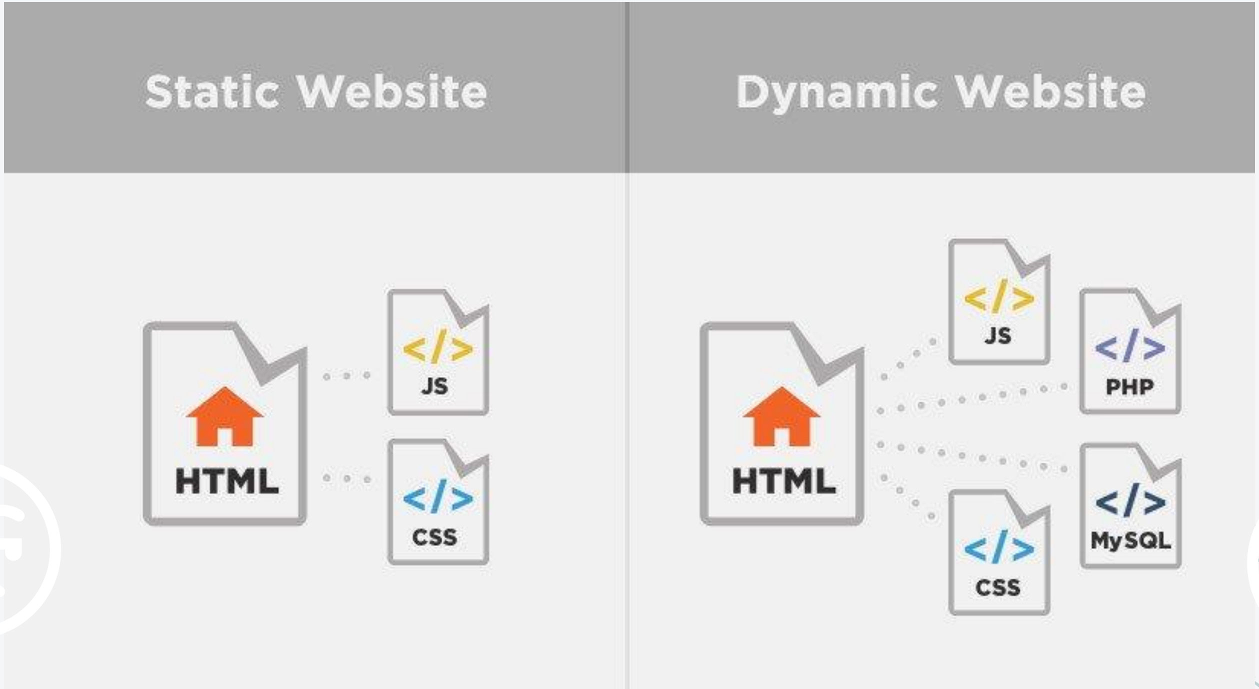
jQuery

ANGULARJS

React

(JS) JavaScript

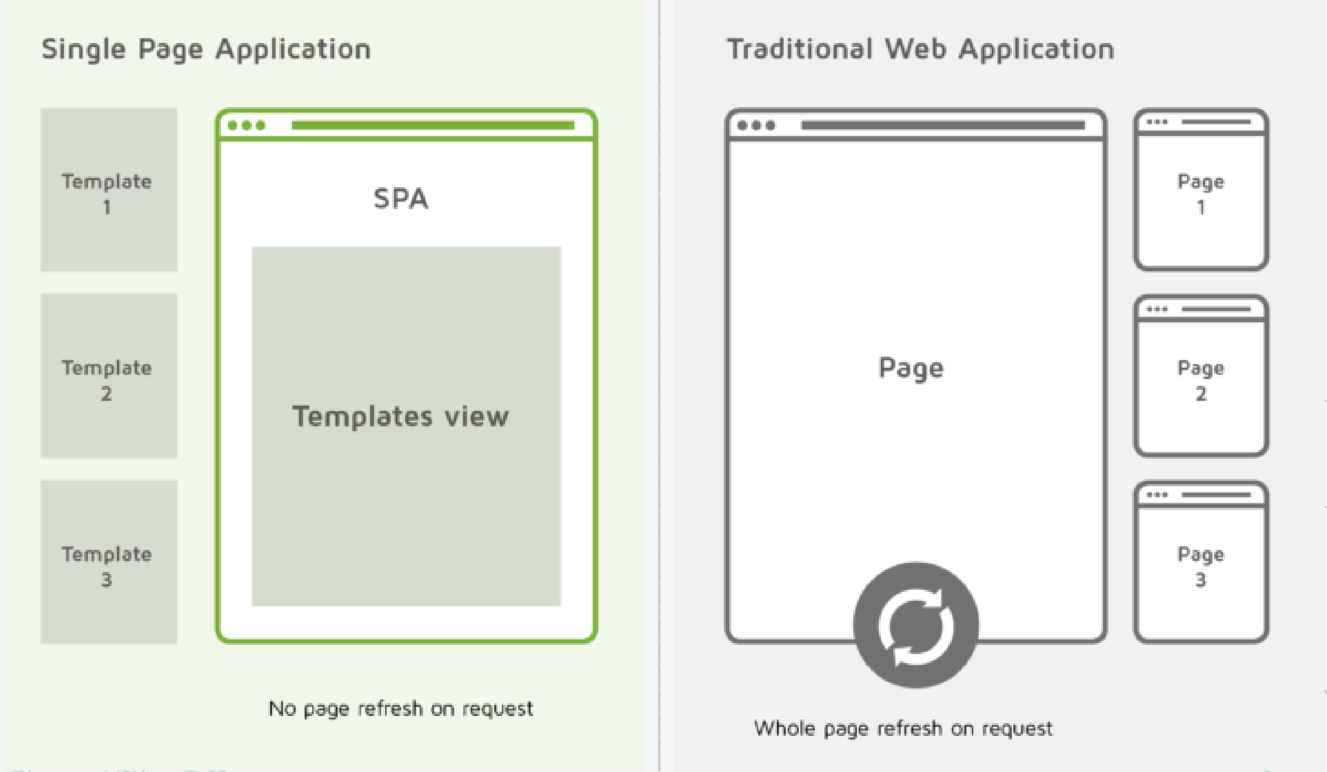
A website becomes dynamic when content changes



SPA …Not this kind.

(SPA) Single Page App

JavaScript



Single Page Application

Template 1

Template 2

Template 3

SPA

Templates view

No page refresh on request

Traditional Web Application

Page

Page 1

Page 2

Page 3

Whole page refresh on request

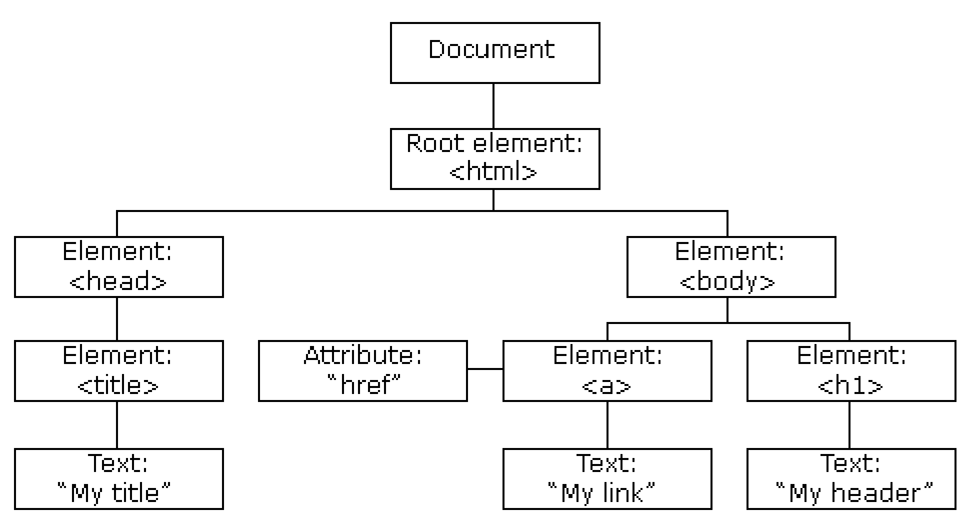
(JS) JavaScript

JavaScript is nothing without an (API) Application Programming Interface

Content Manipulation	Data
DOM (Document Object Model)	AJAX (Asynchronous JavaScript and XML)
Web APIs	Third Party APIs

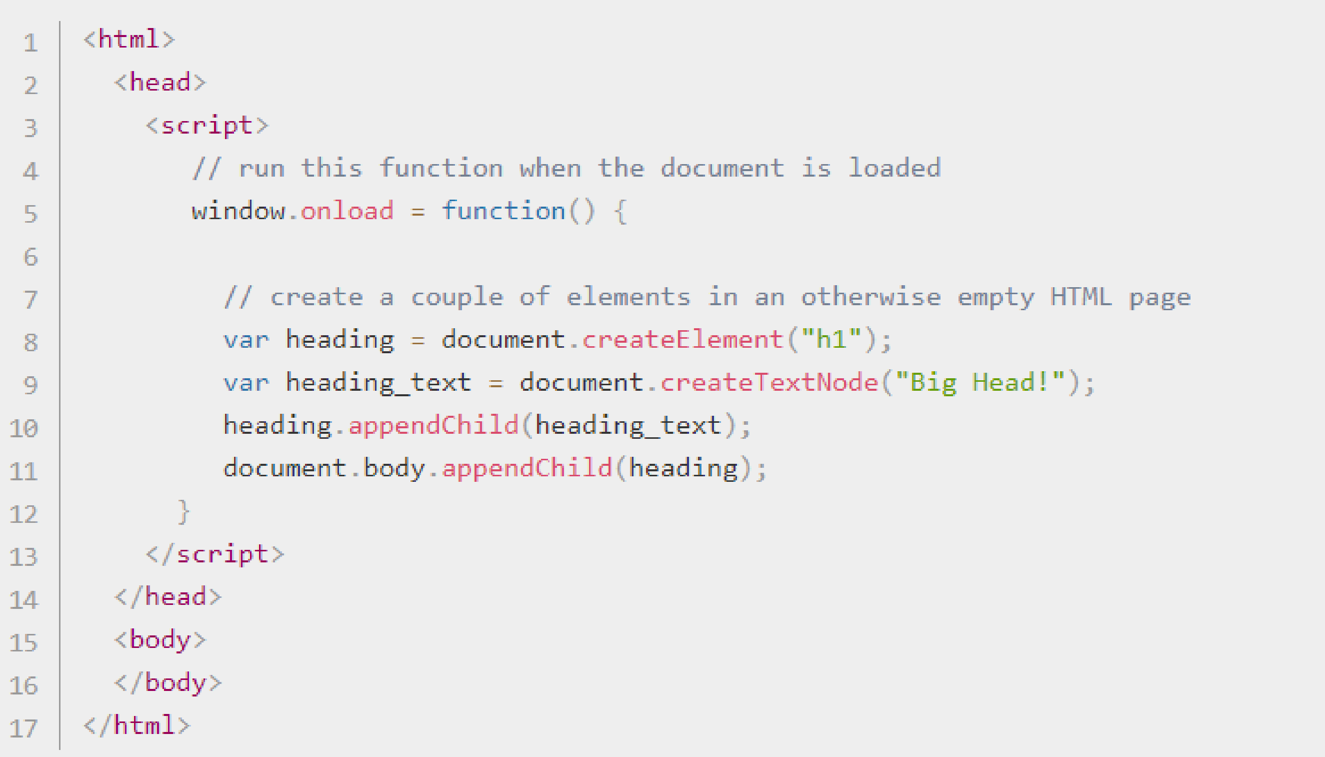
(DOM) Document Object Model

- ✓ Programming interface for HTML and XML documents
- ✓ Not a programming language
- ✓ Is not JavaScript!
- ✓ Represents the document in memory as objects
- ✓ Allowing programs to change structure, style and content



(DOM) Document Object Model

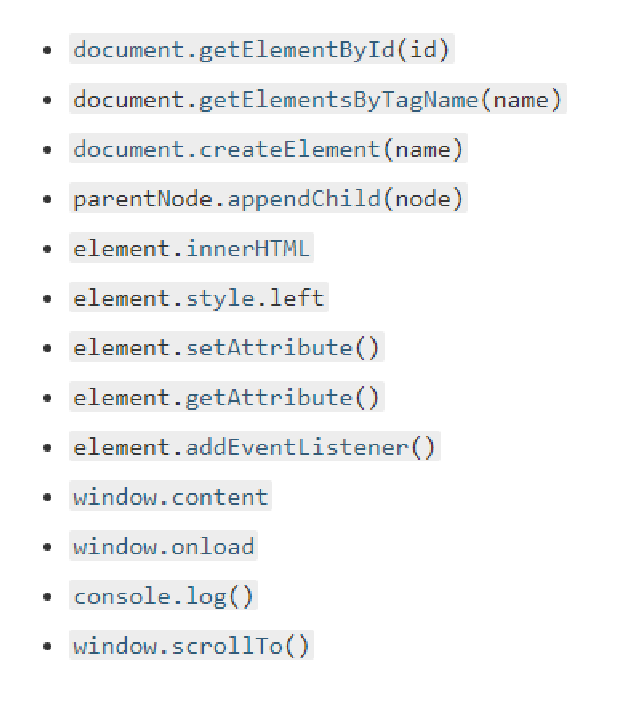
- ✓ Example of JavaScript utilizing the DOM API



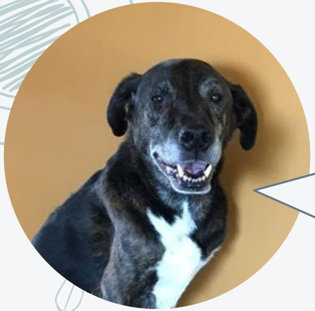
```
<html>
  <head>
    <script>
       // run this function when the document is loaded
       window.onload = function() {

         // create a couple of elements in an otherwise empty HTML page
         var heading = document.createElement("h1");
         var heading_text = document.createTextNode("Big Head!");
         heading.appendChild(heading_text);
         document.body.appendChild(heading);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Web APIs

- ✓ DOM is a Web API
- ✓ See Interfaces
  https://developer.mozilla.org/en-US/docs/Web/API#Interfaces
- ✓ See Events
  https://developer.mozilla.org/en-US/docs/Web/Events



- `document.getElementById(id)`
- `document.getElementsByTagName(name)`
- `document.createElement(name)`
- `parentNode.appendChild(node)`
- `element.innerHTML`
- `element.style.left`
- `element.setAttribute()`
- `element.getAttribute()`
- `element.addEventListener()`
- `window.content`
- `window.onload`
- `console.log()`
- `window.scrollTo()`

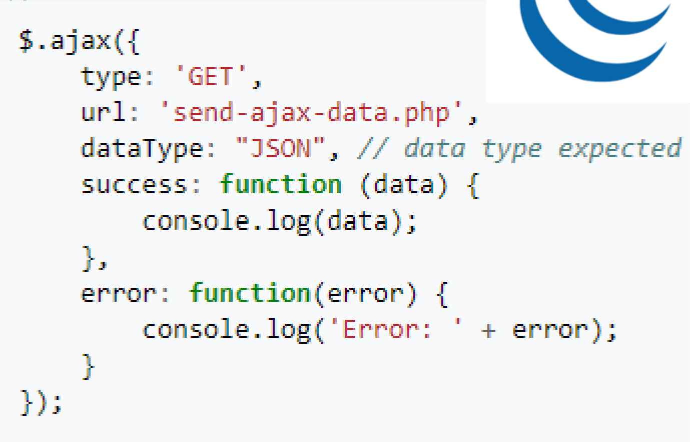


So is the DOM really the nervous system? And JavaScript is just the brain sending the signals?

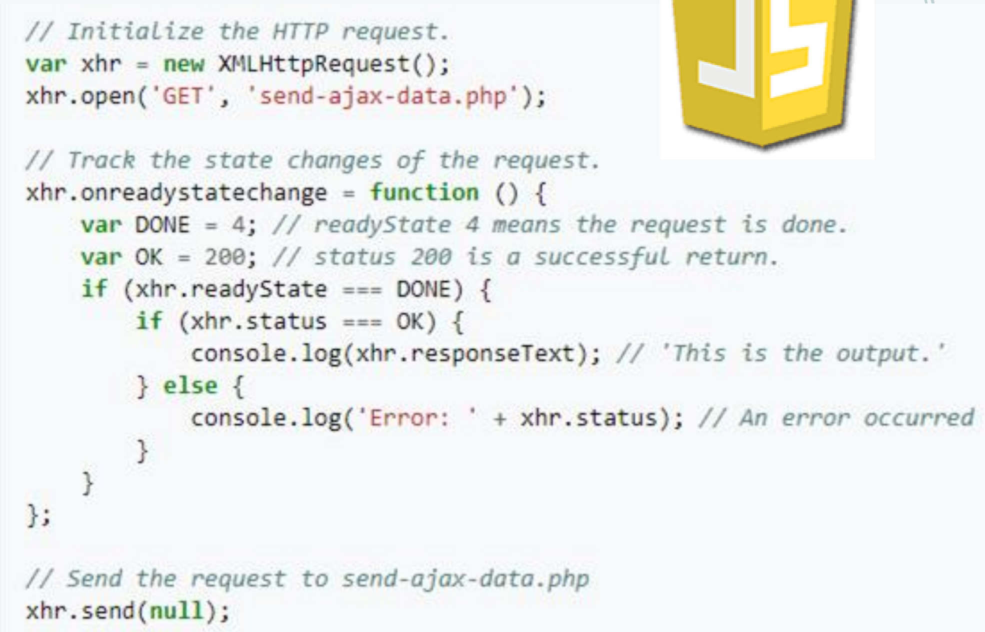
(AJAX) Asynchronous JavaScript and XML

- ✓ Not a programming language
- ✓ Set of web techniques
- ✓ Asynchronous = in the background
- ✓ Decouples the data interchange layer from the presentation layer
- ✓ Allows for dynamic content between client and server without ‘reloading’ the page (SPA)
- ✓ Uses XMLHttpRequest object

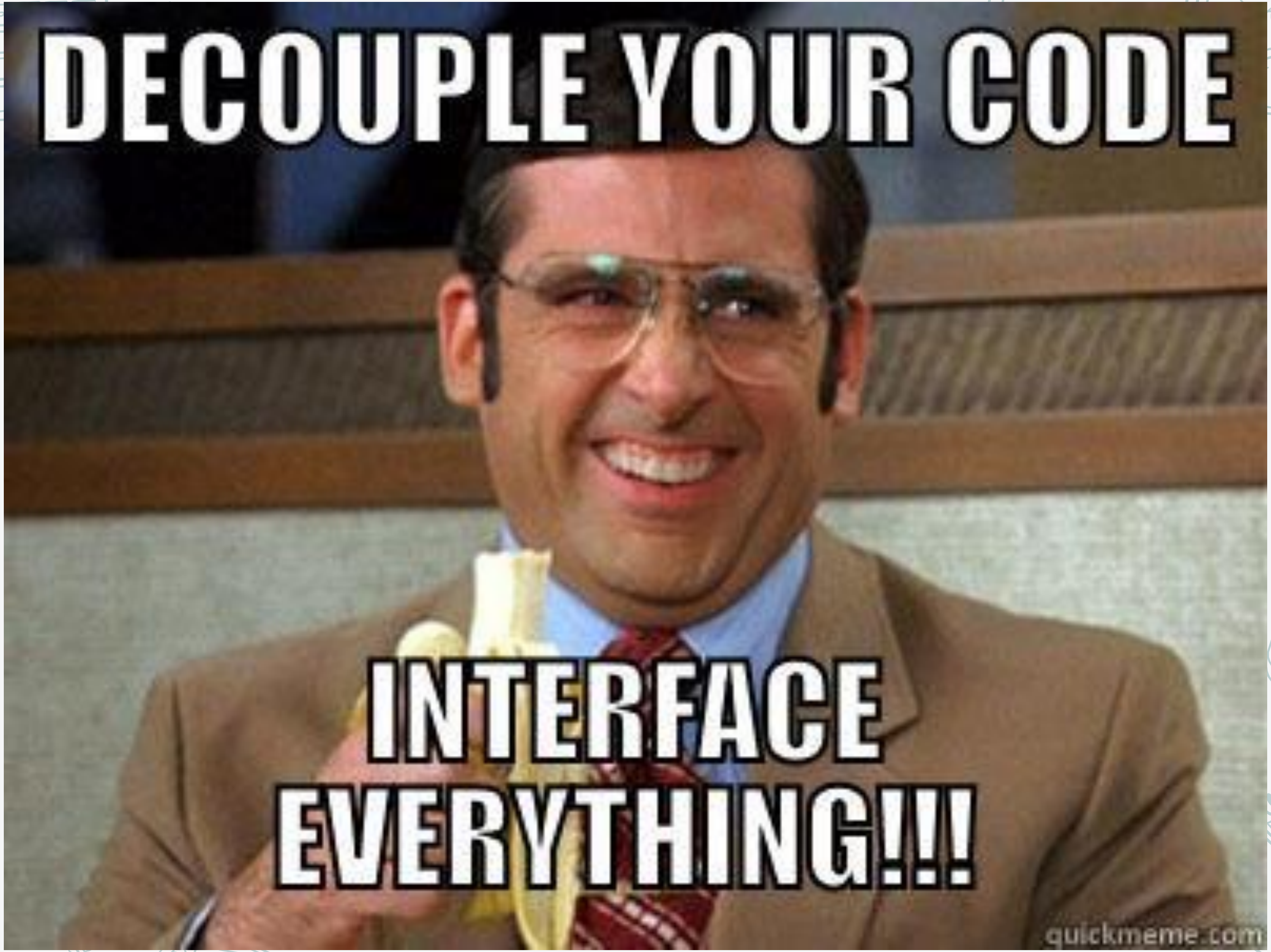
(AJAX) Asynchronous JavaScript and XML



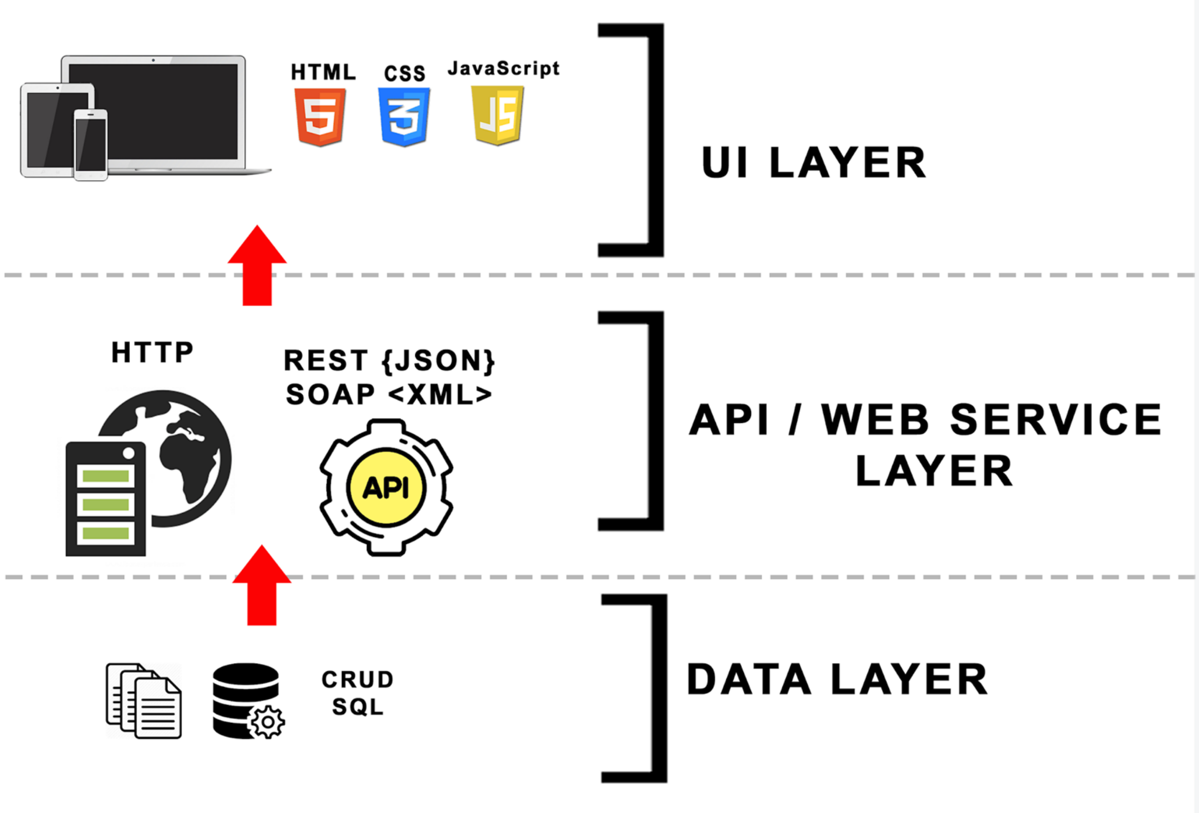
```
$.ajax({
    type: 'GET',
    url: 'send-ajax-data.php',
    dataType: "JSON", // data type expected
    success: function (data) {
        console.log(data);
    },
    error: function(error) {
        console.log('Error: ' + error);
    }
});
```



```
// Initialize the HTTP request.
var xhr = new XMLHttpRequest();
xhr.open('GET', 'send-ajax-data.php');

// Track the state changes of the request.
xhr.onreadystatechange = function () {
    var DONE = 4; // readyState 4 means the request is done.
    var OK = 200; // status 200 is a successful return.
    if (xhr.readyState === DONE) {
        if (xhr.status === OK) {
            console.log(xhr.responseText); // 'This is the output.'
        } else {
            console.log('Error: ' + xhr.status); // An error occurred
        }
    }
};

// Send the request to send-ajax-data.php
xhr.send(null);
```


DECOUPLE YOUR CODE
INTERFACE EVERYTHING!!!
quickmeme.com

Decoupling allows for modernizing



API Web Service Layer

- ✓ UI doesn’t care where the data is coming from!
- ✓ Create a web service api on top of legacy software
- ✓ Use new UI tools with your existing software
- ✓ Open source options available

IT FRESCHE SOLUTIONS

apigility by zend framework

node JS

Seriously what is Node, Angular and TypeScript?

NODE AND (NPM) NODE PACKAGE MANAGER

- ✓ Open source server environment
- ✓ Node runs JavaScript on the server

- ✓ Online repository for publishing open-source Node.js projects
- ✓ Command-line utility for interacting with repository
  - ○ Package installation
  - ○ Version management
  - ○ Dependency management

Angular and TypeScript

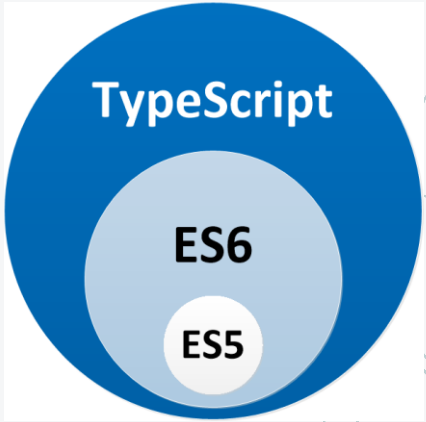
- ✓ Two major versions
  - ○ Angular JS = JavaScript framework
  - ○ Angular 2+ = TypeScript framework
- ✓ Open source project led by Angular team at Google

- ✓ Superset of JavaScript
- ✓ Compiles to JavaScript
- ✓ Static and Strongly Typed
- ✓ Object Oriented Programming language
- ✓ Developed and maintained by Microsoft

Why TypeScript

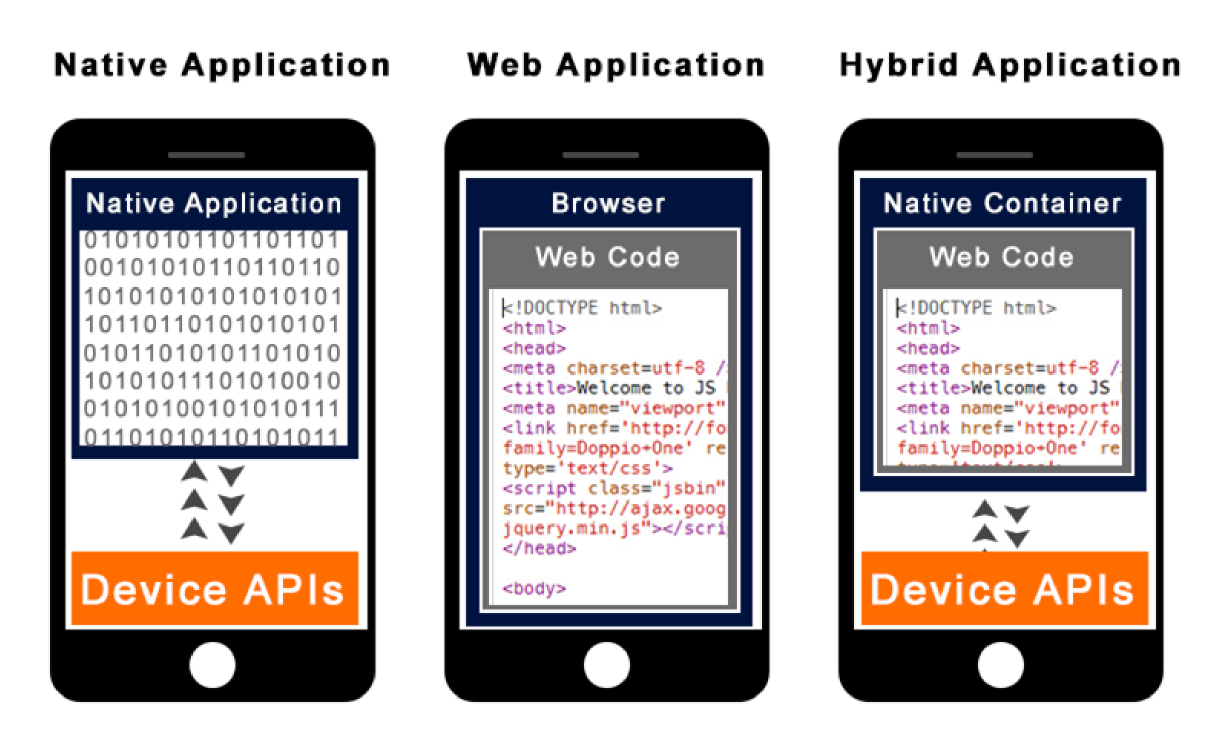
- ✓ Simplifies JavaScript making it easier to read and debug
- ✓ Compile time static type checking
- ✓ Compile to JavaScript different ES versions

TypeScript



Types of Web/Mobile Apps

Types of Apps



```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 /
<title>Welcome to JS
<meta name="viewport"
<link href='http://fo
family=Doppio+One' re
type='text/css'>
<script class="jsbin"
src="http://ajax.goog
jquery.min.js"></scri
</head>

<body>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 /
<title>Welcome to JS
<meta name="viewport"
<link href='http://fo
family=Doppio+One' re
```

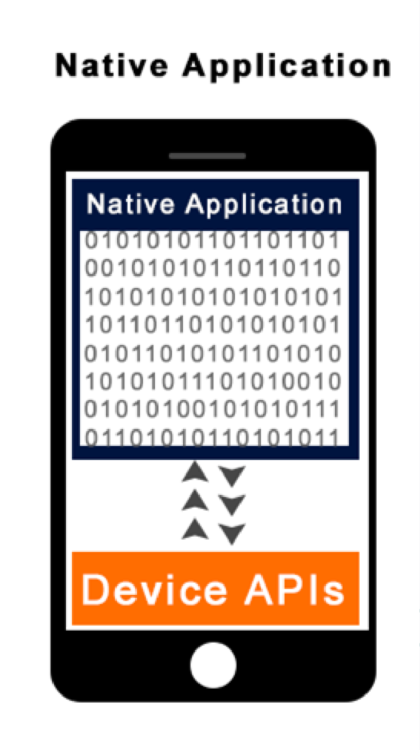
Types of Apps

Pros

- ✓ Direct access to Device
- ✓ Superior user experience and interactivity
- ✓ Best performance

Cons

- × More costly upfront
- × Requires experienced developers
- × Separate code for each platform



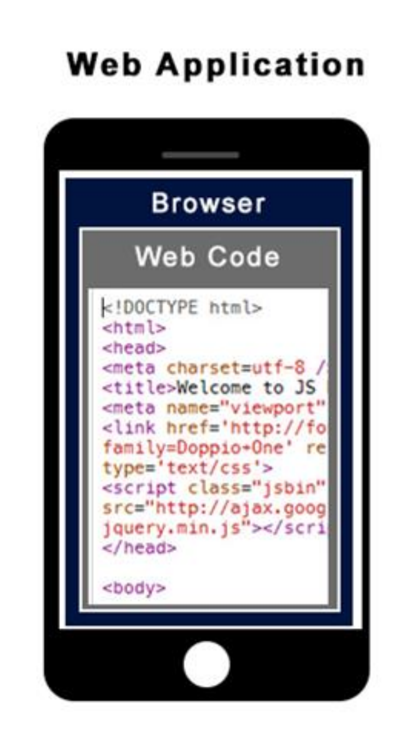
Types of Apps

Pros

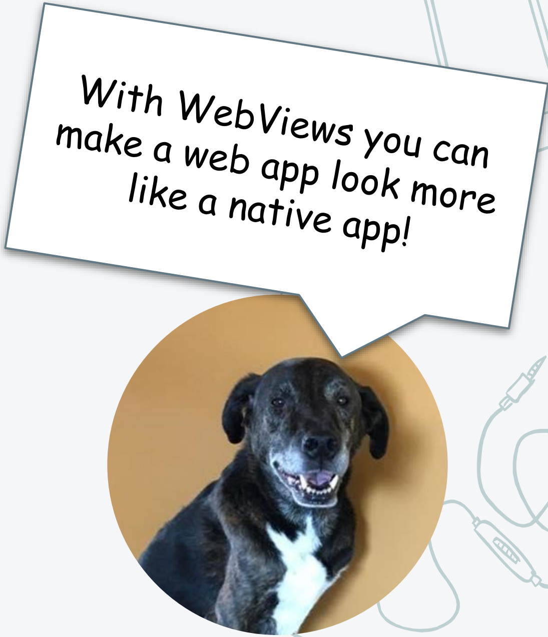
- ✓ Runs in browser on device
- ✓ Doesn’t take up storage
- ✓ Wide range of users

Cons

- × Limited access to device features / hardware
- × Slower and less interactive than native apps



```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 /
<title>Welcome to JS
<meta name="viewport"
<link href='http://fo
family=Doppio+One' re
type='text/css'>
<script class="jsbin"
src="http://ajax.goog
jquery.min.js"></scri
</head>

<body>
```



With WebViews you can make a web app look more like a native app!

Types of Apps

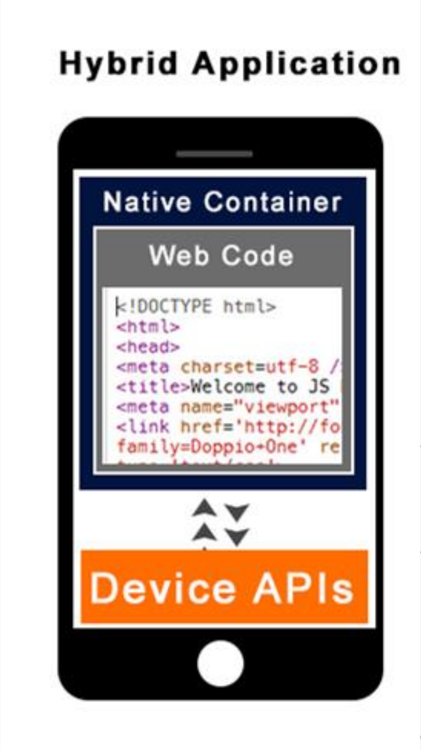
Pros

- ✓ Access to device’s internal API and hardware
- ✓ Doesn’t use browser

Cons

- × Dependent on a third-party platform to deploy the app’s wrapper

JavaScript Frameworks like React Native and Ionic allow developers to build authentic native iOS and Android apps with one codebase!



Hybrid Application

Native Container

Web Code

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 /
<title>Welcome to JS
<meta name="viewport"
<link href='http://fo
family=Doppio+One' re
```

Device APIs

THANK YOU!

Any questions?

I can take questions too!