

# Modern Web Development

with HTML5, CSS3, & JavaScript libraries  
(includes jQuery) (5 days)



# **Copyright Karmoxie, LLC 2018**

All Rights Reserved.

This document may not be copied or reproduced in any form without prior written consent of Karmoxie LLC.

All trademarks belong to their respective companies.

Course Author: Judy Lipinski, Karmoxie Consulting [judy@karmoxie.com](mailto:judy@karmoxie.com)

# Introduction



## Course Objectives and Overview

Course Logistics and Introductions

Development Environment Setup

Using Git to obtain Course Demos & Solutions

Basics of VSCode

# Course Objectives

- What is **Modern Web Development**?
  - Set up a local **Node** server for development and make **AJAX** calls
- What are **HTML5** semantic tags, forms, and features?
  - Work with **HTML5** APIS: Geolocation, Media, Canvas, and Web Storage
- How can **CSS3** separate design from content and address accessibility?
  - Add styling to pages and save money in print
- What is **Bootstrap**?
- What is Modern **JavaScript** (ES6)?
- How can **jQuery** enhance productivity and allow cross-browser compatibility?

# This class always has people with different experience

- Whether a beginner or more experienced - let's get you to the next level
- Hands-On exercises are essential
  - Much of what is needed during exercises is found **on slides**
- Beginners - many demos and solutions for exercises are provided
  - read and try to reproduce
- More experienced students: challenge yourself
  - Attempt bonuses
  - Compare solutions to your approach
  - Implement your imagination

# Course Outline

- Chapter 0      Introduction
- Chapter 1      Overview of Modern Web Dev and Node
- Chapter 2      Working with Browsers
- Chapter 3      HTML5 Elements
- Chapter 4      CSS3 Features
- Chapter 5      Responsive Design
- Chapter 6      Modern JavaScript
- Chapter 7      Client-Side APIs
- Chapter 8      Working with HTML5 Forms

# Course Outline

Chapter 9      Bonus: JavaScript Modules

Chapter 10     Bonus: Unit Testing JavaScript

Chapter 11     jQuery Overview and Selectors

Chapter 12     CSS Manipulation and more jQuery Selectors

Chapter 13     jQuery Event Model

Chapter 14     jQuery DOM Manipulation

Chapter 15     jQuery Animation and Effects

Chapter 16     jQuery AJAX

Chapter 17     Summary

# Introduction

Course Objectives and Overview



## **Course Logistics and Introductions**

Development Environment Setup

Using Git to obtain Course Demos & Solutions

Basics of VSCode

# Logistics

Start Time:	8:30 am (Central)
Mid morning break:	At least one 15 minute break
Lunch:	12 pm - 1 pm
Afternoon break:	At least one 15 minute break
End Time:	5:30 pm (Monday - ending 4:00pm)

Please be mindful of being on time!

# Student Introductions

- Name & City
- Current Position
- Programming experience?
- Expectations for this class?
- What do you like to do for fun?



# About Your Instructor



- Judy Lipinski
  - [judy@karmoxie.com](mailto:judy@karmoxie.com)      [www.karmoxie.com](http://www.karmoxie.com)
- Background:
  - Web & Mobile Development
  - Software Engineering, Consulting, Project Management
  - Trainer & Mentor
- Expectations:
  - That you learn some new web dev and challenge yourself

# About Your Instructor

- Chris Cioffi
  - [chris@karmoxie.com](mailto:chris@karmoxie.com)      [www.karmoxie.com](http://www.karmoxie.com)
- Background:
  - Penn State - Engineering College -- B.S. in Computer Science
  - Software Engineer and Integration Consultant in Financial Technology
  - Trainer
- Expectations:
  - Your active participation will lead to strong web dev foundation



# **GREAT resources for HTML, CSS, JS and more**

- w3schools.com
  - <https://www.w3schools.com/>
  - Brief introduction to starting material for HTML, CSS, JS and more
- Mozilla Developer Network
  - <https://developer.mozilla.org/en-US/>
  - More in-depth coverage of web technologies

# Introduction

Course Objectives and Overview

Course Logistics and Introductions



## **Development Environment Setup**

Using Git to obtain Course Demos & Solutions

Basics of VSCode

# Connecting to ProTech Remote Machines

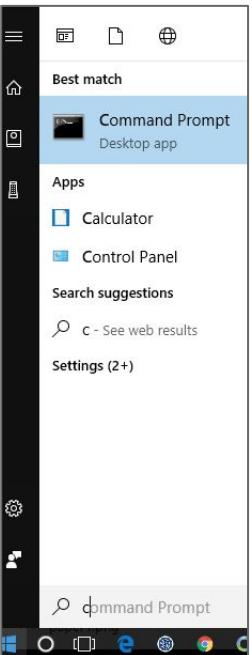
- To adhere to security policies, you will connect to:
  - a Protech remote machine **from** your Wells Fargo training PC
- Ensure your Well Fargo Training PC is maximized
  - Then use IE11 or Chrome to go to:
    - <http://labs.protechtraining.com>
- You will log in with a user and password supplied for you
  - FIRST put the browser in full screen
- Let's review this and do this together now

# Software used for class

- The class virtual machine uses Git, VSCode, Node, and npm

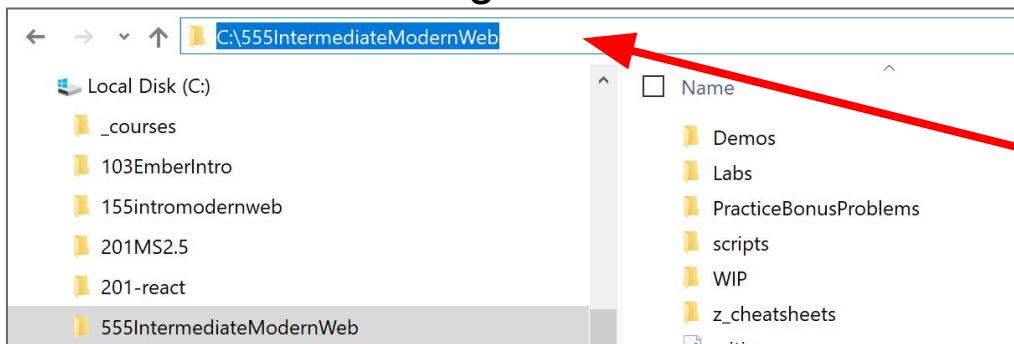


- **Git** will be used to obtain the coursefiles and demo projects
- **VScode** will be used to create and edit files
- **Node (8.9+)** will be used to execute JavaScript outside of the browser
- **npm (5+)** will be used to install additional packages and libraries
  - from the npm server: <https://www.npmjs.com/>



# 2 ways to reach Command Prompt in Windows

1. Can click start menu in bottom left and start typing **cmd**
  - then navigate using cd (change directory)
  
2. From Windows Explorer, while viewing the directory you want:
  - Can click in file path and type **cmd** and hit **return**
  - Or Shift-Right-Click and then choose from menu to open here



type **cmd** here

# Helpful commands at Windows Prompt

**dir** to list the files and directories

**cd directory\_name** for change directory

- You can hit Tab to autocomplete long names

**cd ..** goes up one level, to parent folder

# Reaching Terminal window on Mac (if needed)

- On a Mac, can hit command-space for Spotlight and type **terminal**
  - **ls** to list the files and directories
  - **cd** to change directory
- Or, can setup a shortcut in Finder
  - System Preferences: Keyboard > Shortcuts > Services
  - Click box for "New Terminal at Folder" in the settings
  - From then on, in Finder,
    - just right-click a folder and you're shown New Terminal at Folder



# Software versions can be confirmed from prompts

## Command Prompt/Powershell on Windows

```
C:\Windows\System32\cmd.exe  
C:\Users\judy\Documents\course-dev\555intermediatemodernweb>dir  
Volume in drive C is OS  
Volume Serial Number is 0668-6C66  
  
Directory of C:\Users\judy\Documents\course-dev\555intermediatemodernweb  
  
03/03/2018 10:03 AM <DIR> .  
03/03/2018 10:03 AM <DIR> ..  
03/03/2018 10:03 AM 932 .gitignore  
03/03/2018 10:03 AM <DIR> Demos  
03/03/2018 10:03 AM <DIR> Labs  
03/03/2018 10:03 AM <DIR> scripts  
      1 File(s)    932 bytes  
      5 Dir(s) 250,771,734,528 bytes free  
  
C:\Users\judy\Documents\course-dev\555intermediatemodernweb>git --version  
git version 2.11.1.windows.1  
  
C:\Users\judy\Documents\course-dev\555intermediatemodernweb>code --version  
1.19.2  
490ef761b76b3f3b3832eff7a588aac891e5fe80  
ia32  
  
C:\Users\judy\Documents\course-dev\555intermediatemodernweb>node -v  
v8.9.3  
  
C:\Users\judy\Documents\course-dev\555intermediatemodernweb>npm -v  
5.6.0
```

## Terminal on Mac

```
Judys-MBP:IntermediateModernWeb555 judylipinski$ ls -la  
total 8  
drwxr-xr-x  7 judylipinski  staff  238 Jan 23 09:22 .  
drwxr-xr-x@ 22 judylipinski  staff  748 Mar  3 09:45 ..  
drwxr-xr-x  13 judylipinski  staff  442 Mar  3 10:18 .git  
-rw-r--r--   1 judylipinski  staff  872 Jan 23 09:22 .gitignore  
drwxr-xr-x  15 judylipinski  staff  510 Jan 23 09:22 Demos  
drwxr-xr-x  14 judylipinski  staff  476 Jan 23 09:22 Labs  
drwxr-xr-x   3 judylipinski  staff  102 Jan 23 09:22 scripts  
Judys-MBP:IntermediateModernWeb555 judylipinski$ git --version  
git version 2.14.3 (Apple Git-98)  
Judys-MBP:IntermediateModernWeb555 judylipinski$ code --version  
1.20.1  
f88bbf9137d24d36d968ea6b2911786bfe103002  
x64  
Judys-MBP:IntermediateModernWeb555 judylipinski$ node -v  
v9.2.1  
Judys-MBP:IntermediateModernWeb555 judylipinski$ npm -v  
5.7.1
```

# DO NOW: Confirm installed software

5 min

1. Open a command prompt or terminal window
2. Confirm the version numbers on your machine for
  - a. Git: `git --version`
  - b. VScode: `code --version`
  - c. Node (at least 8): `node -v`
  - d. npm (at least 5): `npm -v`

# Introduction

Course Objectives and Overview

Course Logistics and Introductions

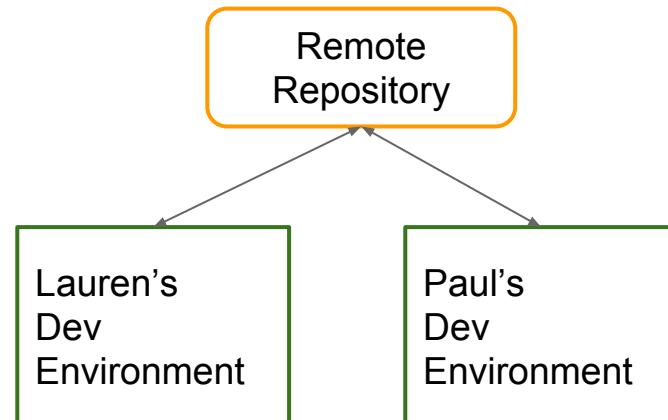
Development Environment Setup

## ➤ **Using Git to obtain Course Demos & Solutions**

Basics of VSCode

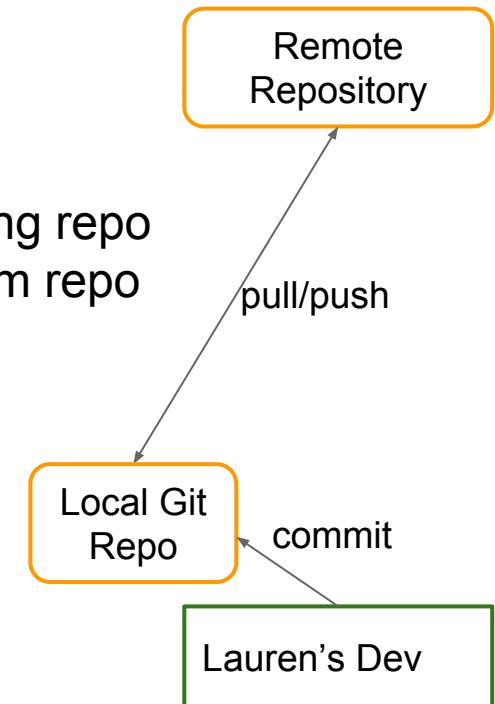
# Version control of software

- There are many options for version control of software
- Developers get current projects into their local dev environment
  - From a repository
- Then put that code back into the repo
- We will be using Git in this course



# Basics of using Git

- Git repos allow developers to do a **pull** from an existing repo
  - updates their local environment with changes from repo
- While working, devs can **commit** changes locally
  - Can revert to previous code if need be
- Developers **push** their changes to the repo
  - git checks there is no conflict with other changes
  - if there is, devs choose how to **merge** their changes
- Developers can work on the same files at the same time



# Initializing a repository in your local environment

- When joining a project, you can **clone** the files project from remote repository
- This sets up
  - the **local** Git repository for your work to be tracked
  - the **remote** repository for you to **pull** additional changes
  - ...and to **push** changes if you have rights to the remote repo

# DO NOW: Get the coursefiles

5 min

- Choose a location for the files for your lab machine:
  - anywhere, but keep the path relatively short
  - recommend using c:\ for Windows
  - and \Users\your-user\Documents on Mac
- Instructor will share **full path** to repository and command via shared doc
- From a command prompt / terminal execute the entire command with git clone
  - **git clone link-to-course-repository**

# **Introduction**

Course Objectives and Overview

Course Logistics and Introductions

Development Environment Setup

Using Git to obtain Course Demos & Solutions



**Basics of VSCode**



# VSCODE is the editor used in class

- Developed by Microsoft, MIT license, [source on GitHub](#)
  - Written in JavaScript (TypeScript)
  - Understands HTML, CSS, JS, TypeScript
- Helpful learning tool
  - Code completion
  - Good source code colorization
- Very extensible, large community of users AND contributors

# Supports HTML, CSS and JavaScript out of the box

- Offers closing HTML tags
- Can clean-up / indent code properly with **SHIFT-ALT-F**
- Autocompletion of CSS and JavaScript options

A screenshot of a code editor interface. On the left, there is a vertical toolbar with icons for file operations like new, open, save, and close. The main area shows a line of code starting with '

# Starting VSCode from prompt

- If installed and on the path - can open from command prompt / terminal
- Navigate to the folder you wish to open, then use **code .**
  - (the dot means to launch with the current directory)

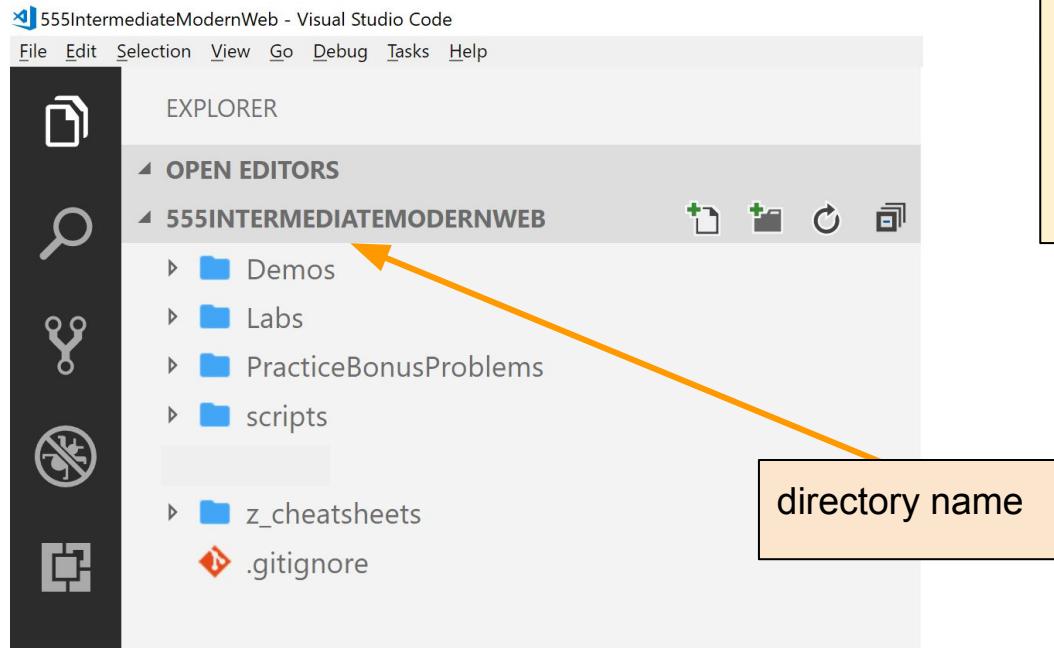
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\555IntermediateModernWeb>code .
```

Windows Setup Help  
<https://code.visualstudio.com/docs/setup/windows>

Mac Setup Help  
<https://code.visualstudio.com/docs/setup/mac>

# What the course directory should look like

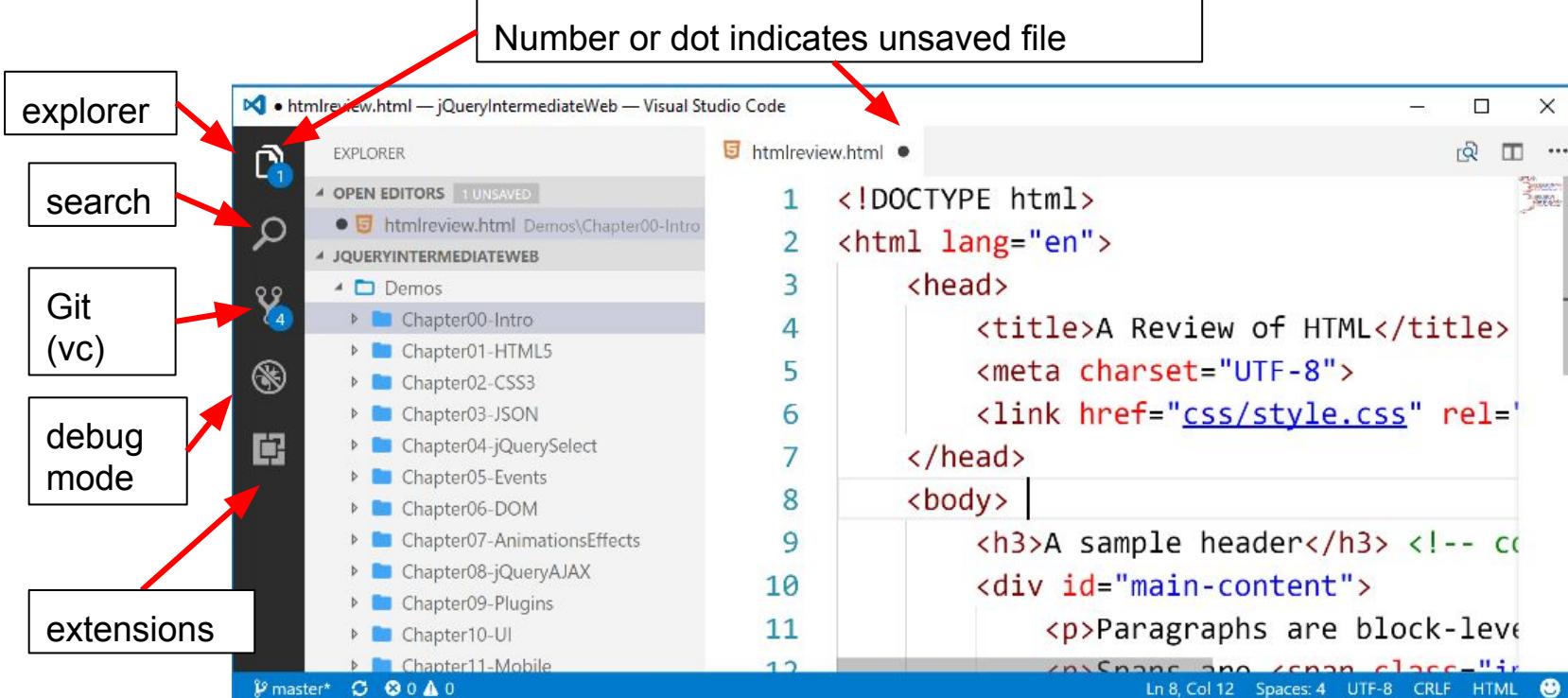


Use **ctrl+(plus sign)** to zoom in

and

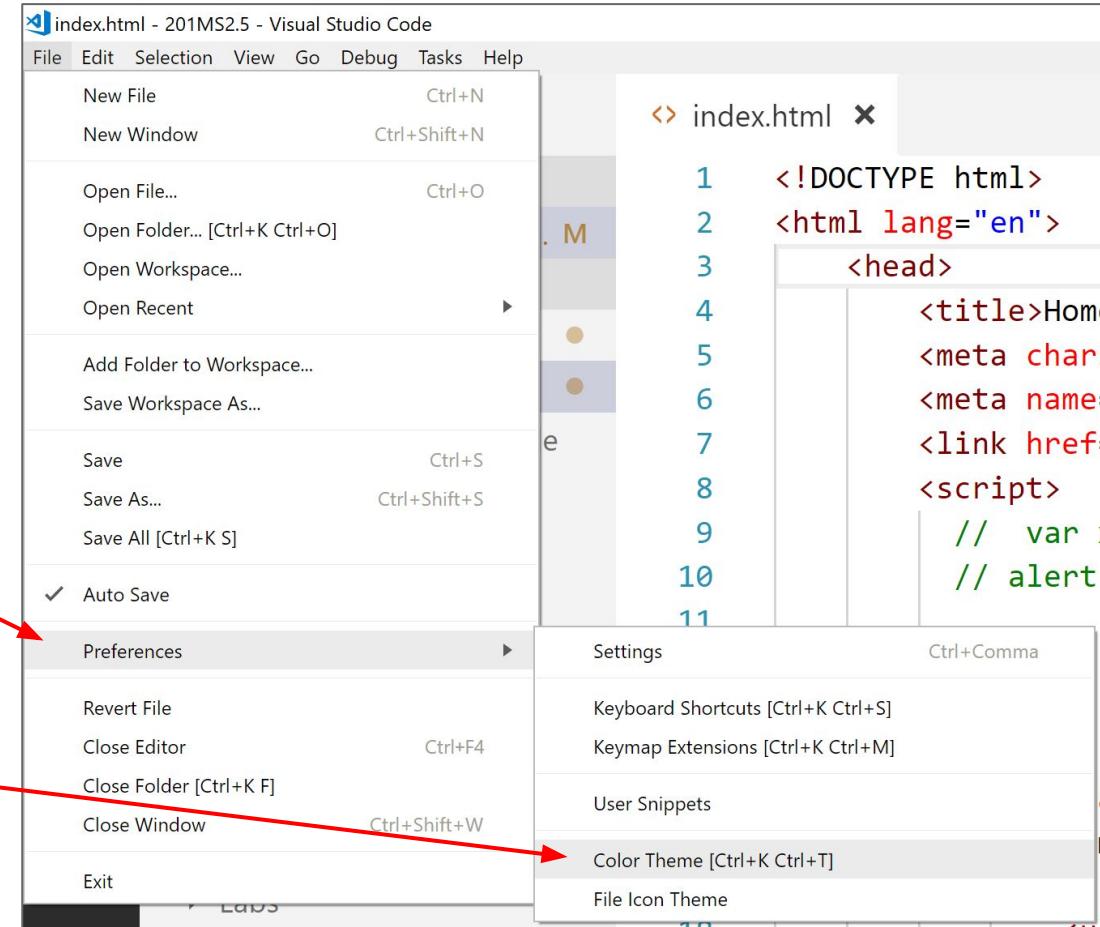
**ctrl+(minus sign)** to zoom out

# Basics of VS Code



# Adjusting Settings

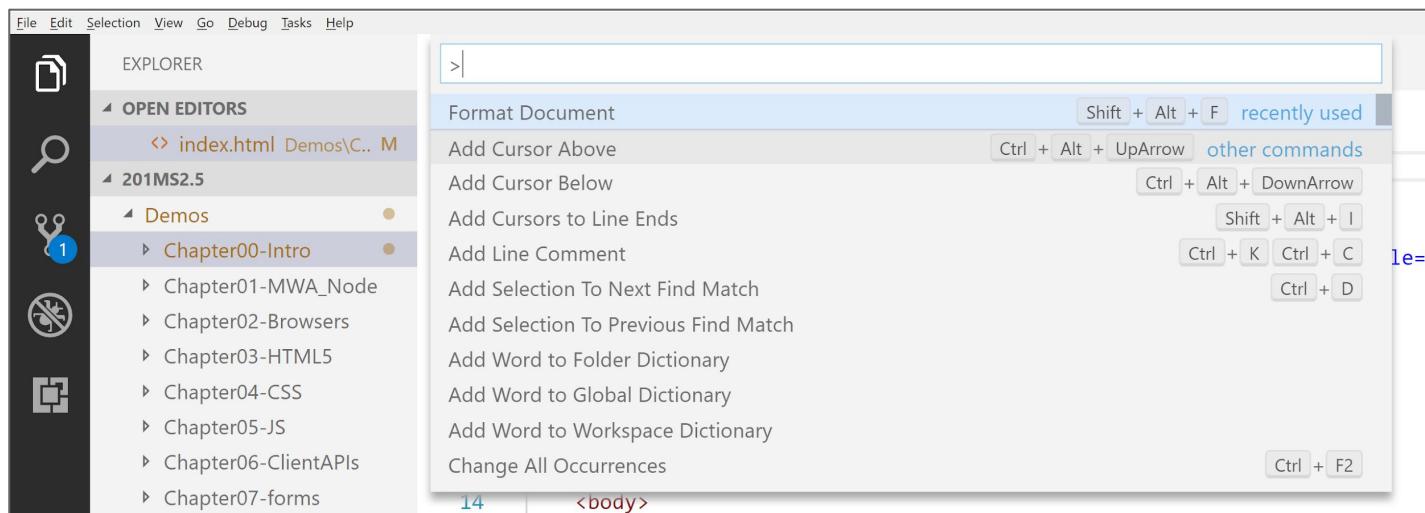
- File | Auto Save
- File | Preferences
  - Color Theme
  - Icons



# Command Palette

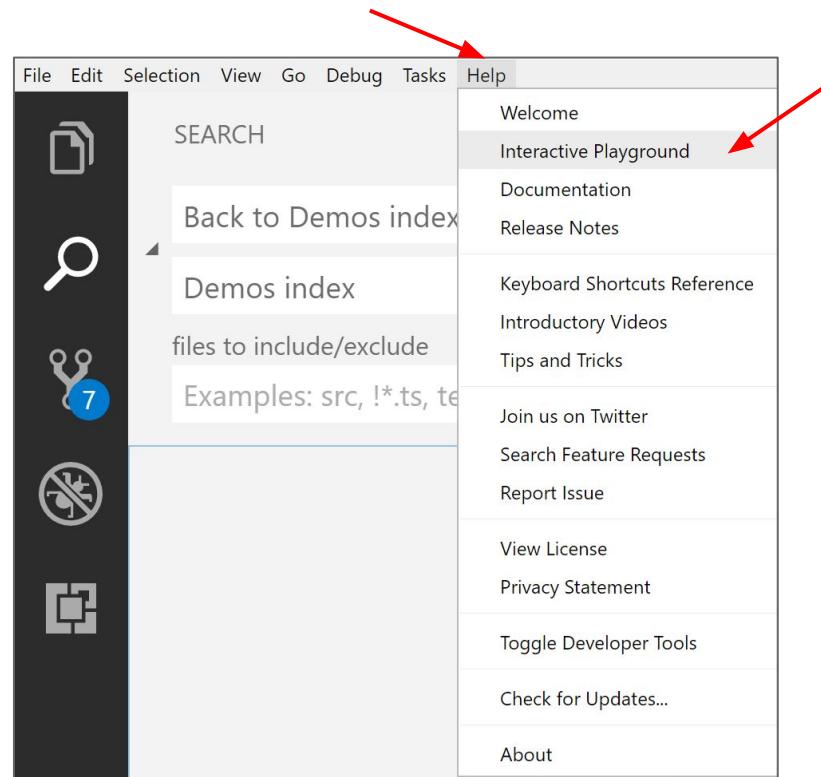
Write this down! If you remember no other VSCode shortcuts, remember this!

- Control-Shift-P | Command-Shift-P
- start typing what you remember... format, icons, color, etc.



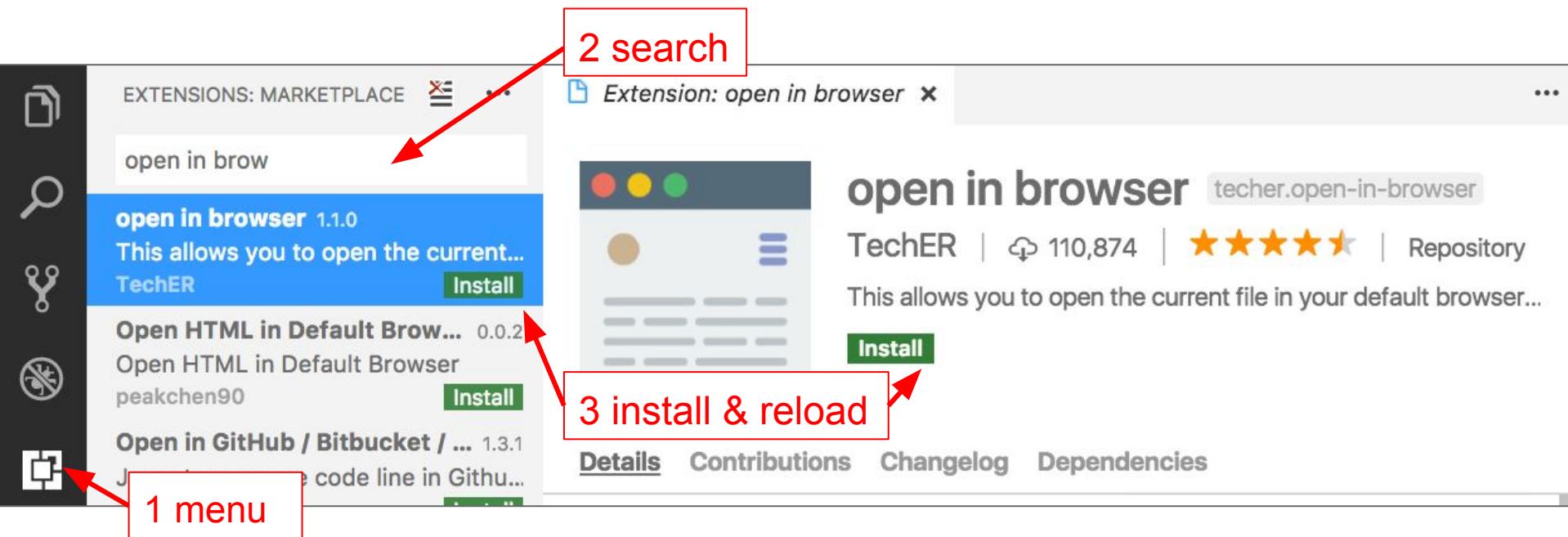
# VScode has many more built in features to offer

- Many more options/features exist
- We will highlight more during class
- If done with exercises before others, explore the Interactive Playground



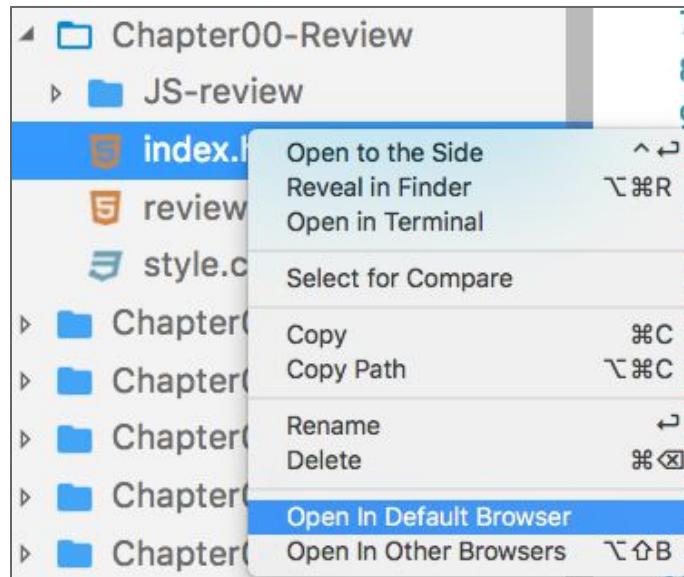
# VSCODE can be enhanced through extensions

- You can modify your environment with helpful extensions



# The Open in Browser extension gives shortcuts

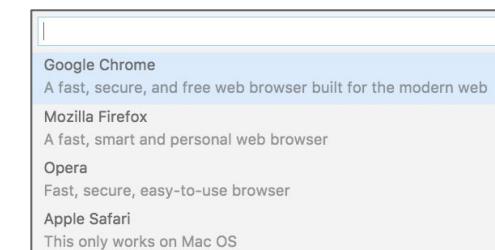
- VSCode extensions allow you to open into a browser directly



Right-click for context menu

Use Alt-B for default browser

Use Alt-Shift-B for list



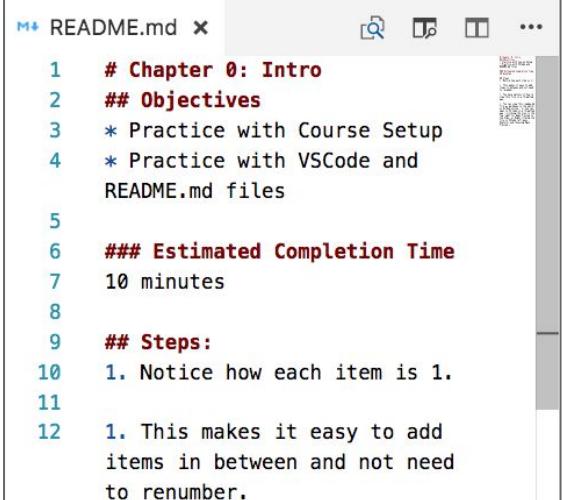
# DO NOW: Open HTML file in browser with short-cut

5 min

1. Confirm **Open in Browser** extension is installed in VSCode
  - a. If it isn't, follow the slides to install
2. Navigate to \Demos\Chapter00-Intro\simpleDemos\index.html
  - a. Double-click to open the file in the editor
3. Use Alt-B or right click the file to Open in Browser
  - a. If not already set, **choose Chrome** as the default browser

# Markdown Files are used in Modern Web

- Markdown files are designed to be converted
  - to HTML and many other formats
- Markdown is often used to:
  - format readme files
  - write messages in online discussion forums
  - create rich text using a plain text editor
- We will be using in class for the instructions to run demos and for exercises



A screenshot of the Visual Studio Code (VSCode) interface showing a file named "README.md". The code editor displays the following Markdown content:

```
1 # Chapter 0: Intro
2 ## Objectives
3 * Practice with Course Setup
4 * Practice with VSCode and
  README.md files
5
6 ### Estimated Completion Time
7 10 minutes
8
9 ## Steps:
10 1. Notice how each item is 1.
11
12 1. This makes it easy to add
  items in between and not need
  to renumber.
```

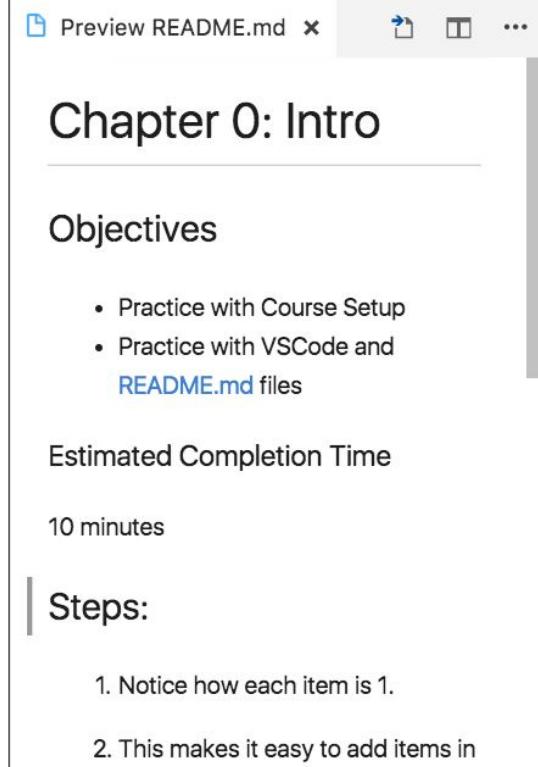
# Lab: Practice with Course Setup & README.md

10 min

- In VSCode, navigate to the folder \Labs\Ch00-Intro
- Open the file README.md file - follow ALL of the steps



```
README.md x 🔎 ⚡ 📁 ...  
1 # Chapter 0: Intro  
2 ## Objectives  
3 * Practice with Course Setup  
4 * Practice with VSCode and  
    README.md files  
5  
6 ### Estimated Completion Time  
7 10 minutes  
8  
9 ## Steps:  
10 1. Notice how each item is 1.  
11  
12 1. This makes it easy to add  
    items in between and not need  
    to renumber.
```



Preview README.md x

## Chapter 0: Intro

---

### Objectives

- Practice with Course Setup
- Practice with VSCode and README.md files

### Estimated Completion Time

10 minutes

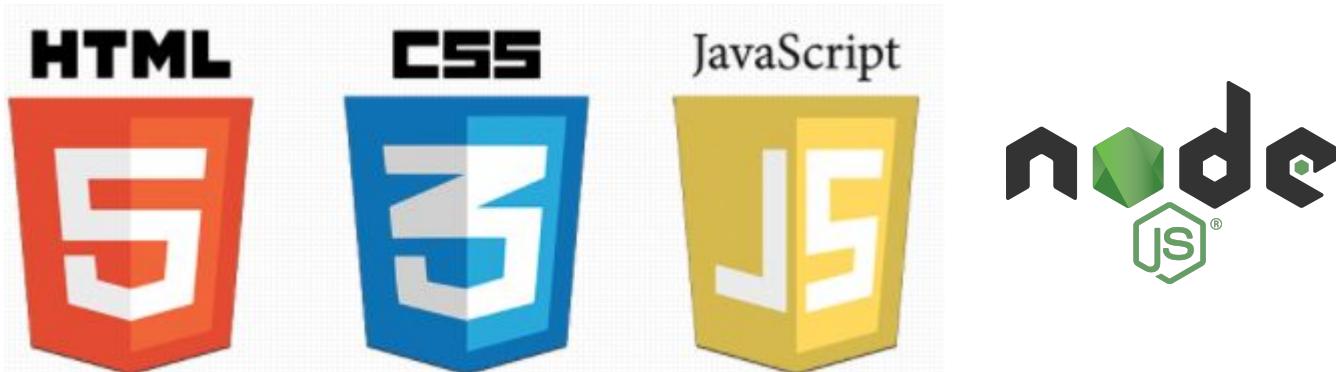
### Steps:

- Notice how each item is 1.
- This makes it easy to add items in

# Modern Web Development

## Chapter 1

Overview of Modern Web Apps and Node



# Chapter Objectives

In this chapter we will look at:

- Brief overview of HTML5, CSS3, and JavaScript
- Web Accessibility Guidelines
- Why do we need servers?
- An overview of AJAX and REST APIs
- What is Node? What is npm?
- Managing project dependencies and automating tasks in Modern Web Dev
- Popular libraries and frameworks for SPA's

This chapter is an OVERVIEW of these technologies. Do not expect to have deep skills after this chapter! We will continue using these technologies for the rest of the course.

# Overview of Modern Web Apps & Node



## 3 Pillars, Standards, & Guidelines of Web Dev

File vs HTTP protocol

Introducing Node

Starting / Stopping a Node server

AJAX

REST API

Using package.json and npm

Overview of Modern Web Apps

# Brief history of software architecture (web)

- Thick Client-server
  - install software updates on client machine, floppy, CD, sneaker-net
- Browser as a thin client (no software installs)
  - changes able to be applied when user refreshes
  - HTML, JavaScript (JS), and CSS
- Now popular is SPA, Single Page Applications (a **thick** browser client)
  - still HTML, CSS, JS but pages / views loaded like an application
  - shifting focus to components
  - Angular, React, Vue, Ember are used for SPA

# The 3 Pillars of Front-End Web development

**HTML5** is for content

```
<h1>This is a simple example</h1>
<p id="output">Date goes here</p>
```

**This is a simple example**

Date goes here

# The 3 Pillars of Front-End Web development

**HTML5 is for content**

```
<h1>This is a simple example</h1>
<p id="output">Date goes here</p>
```

**This is a simple example**

Date goes here

**CSS3 is for styling**

```
<style>
  h1 {color: green}
  p#output {color: blue}
</style>
```

**This is a simple example**

Date goes here

# The 3 Pillars of Front-End Web development

**HTML5** is for content

**CSS3** is for styling

**JavaScript** is for behavior

```
<h1>This is a simple example</h1>
<p id="output">Date goes here</p>
```

```
<style>
  h1 {color: green}
  p#output {color: blue}
</style>
```

```
<script>
  document.getElementById("output").innerText=new Date();
</script>
```

This is a simple example

Date goes here

This is a simple example

Date goes here

This is a simple example

Mon Apr 23 2018 10:56:26 GMT-0400 (Eastern Daylight Time)

These 3 technologies make up the basis for Web Standards

# Brief History of W3C and Web Standards

- Longer history: [https://www.w3.org/wiki/The\\_history\\_of\\_the\\_Web](https://www.w3.org/wiki/The_history_of_the_Web)
- 90s: The browser wars - IE & Netscape
- 94: Tim Berners-Lee formed the World Wide Web Consortium (W3C)
  - vision to standardize **protocols** & **technologies** used to build the web
  - HTML, CSS, PNG files
- HTML history: HTML4, DHTML, XHTML...finally HTML5 (2007)

# The WHATWG and the HTML Living Standard

- The W3C based the HTML5 spec on work started by the WHATWG
- The WHATWG continues to make modifications to the HTML standard
  - the “HTML Living Standard”



The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web.

<https://www.w3.org/>



WHATWG

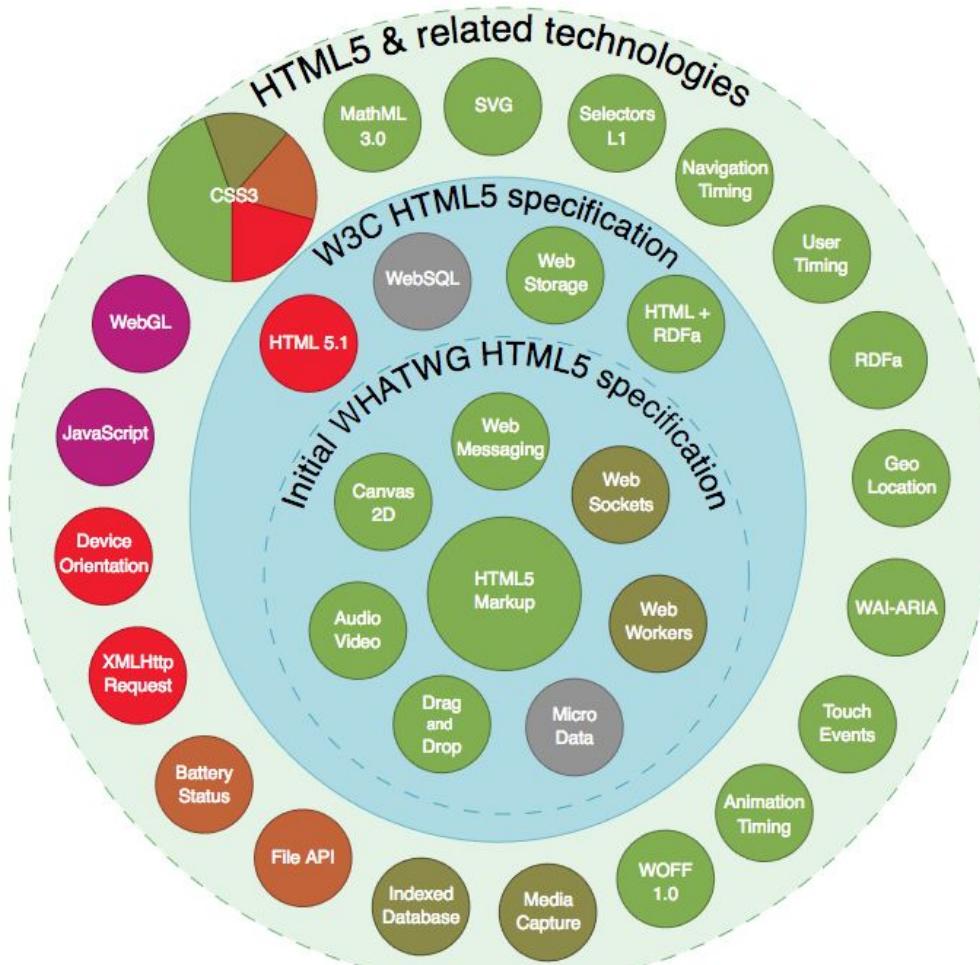
The WHATWG (Web Hypertext Application Technology Working Group) is an organization that maintains and develops HTML and APIs for Web applications

<https://whatwg.org>

# HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



By Mercury999 - Own work, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=36352535>

# Web Accessibility Guidelines

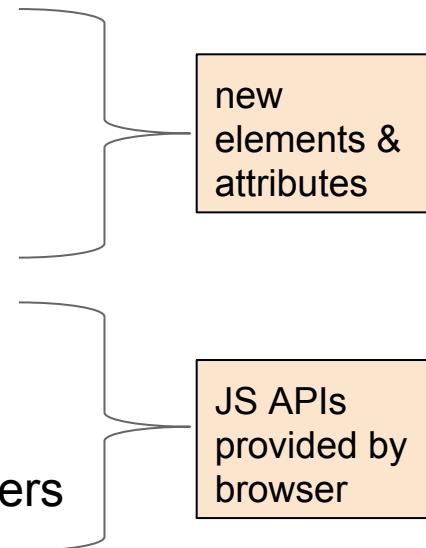
- Section 508: United States federal standard for website accessibility
  - [Set of standards](#) to be aware of when coding sites
  - Refers to the Web Content Accessibility Guidelines (WCAG)
    - published from W3C's Web Accessibility Initiative (WAI)
- Following these ensures users can visit websites with different:
  - abilities and devices
  - not everyone can see or has a keyboard

# Brief Highlights of Section 508

- A text equivalent for every non-text element shall be provided
  - Forms inputs should have descriptive labels
  - Every image has good alt text
- Don't rely on color coding alone to convey meaning
- Sites should be readable without use of CSS stylesheets
- Tables should have header rows or columns, and only be used for data

# HTML5 - New Features

- Better Semantics - more descriptive *elements* and *attributes*
- Form Validation - *without additional coding*
- Multimedia support - *audio and video*
- Drawing on the canvas - *data graphs, custom images*
- Offline Storage - *session and persistent storage*
- Geolocation, Websockets, Server communication, many others



# What is HTML (Hypertext Markup Language)?



What is Hypertext Markup Language?

Links! Pages of content are able to  
<a href="page2.html">Link</a> to one another



What are some examples of content?

Text, paragraphs, headings, tables, images, lists, links, forms

# Structure of HTML

- A hierarchy is formed with **html** as the root element, **body** is a child
- **Elements** are indicated with start tags and end tags
- **Attributes** provide additional information

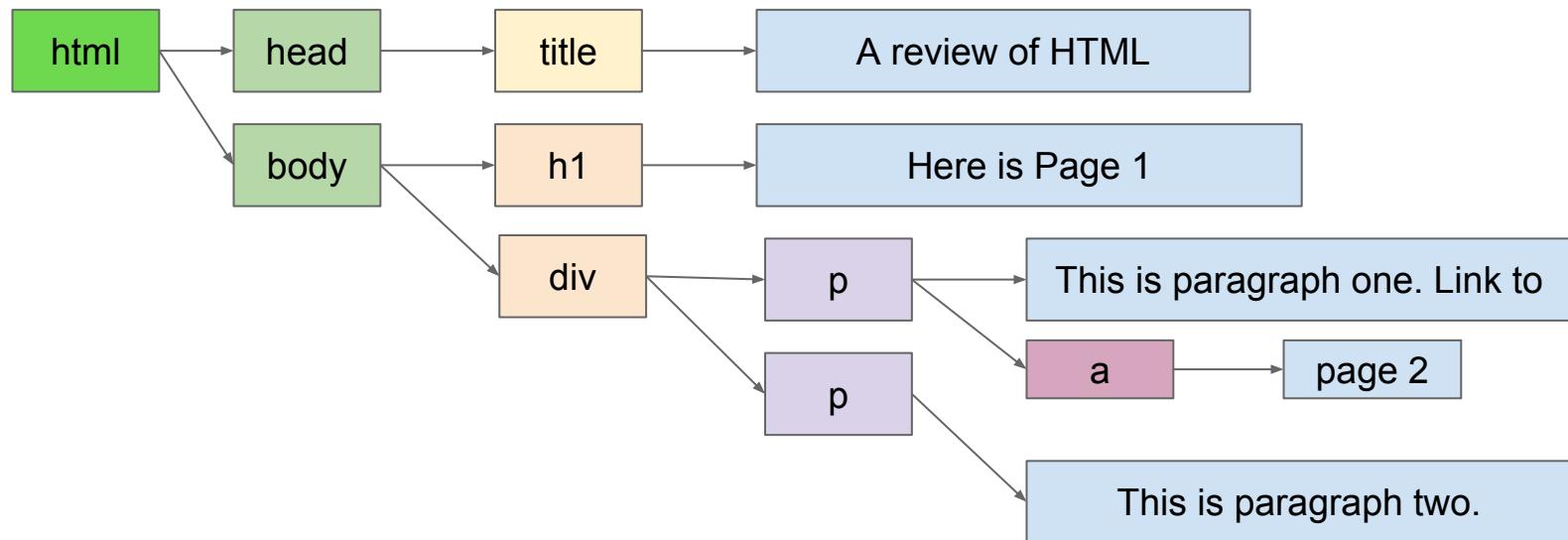


# Browser reads HTML into the DOM in memory



What is the DOM?

The Document Object Model, loaded into **memory** of the browser



# Overview of CSS: rules target the DOM

- **CSS Rulesets**

```
selector { property: value }
```

```
<h1>Here is Page 1</h1>
<div>
  <p class="one">This is
    <a href="page2.htm"
  </p>
  <p class="two">This is
</div>
```

```
6   <style>
7     h1 { color: red }
8
9     p {
10       width: 300px;
11       border: 3px solid green;
12       padding: 5px;
13       height: 40px;
14     }
15
16     p.two { border-color: purple; }
17   </style>
```

CSS Rulesets

Here is Page 1

This is paragraph one. Click for page 2

This is paragraph two

Rendered Page

# Including CSS (Cascading Style Sheets)

- Include external CSS files using link, - OR -
- Include CSS inside HTML files between <style> tags

```
3 <head>
4     <title>CSS inclusion example</title>
5     <link rel="stylesheet" href="main.css" />
6     <style> h1 {color: red} </style>
7 </head>
```

**rel** indicates to apply as a stylesheet

**href** is the location of the file



```
main.css x
1 p {width: 300px;
2     border: 3px solid green;
3     padding:5px;
4     height:40px;}
5
6 p.two {
7     border-color: purple;}
```

# Basics of CSS (Cascading Style Sheets)

- **CSS Rulesets**

```
selector { property: value }
```

- selector targets areas of the DOM
- rules allow us to apply styling/formatting

## Here is Page 1

This is paragraph one. Click for [page 2](#)

This is paragraph two

- We can choose to target:

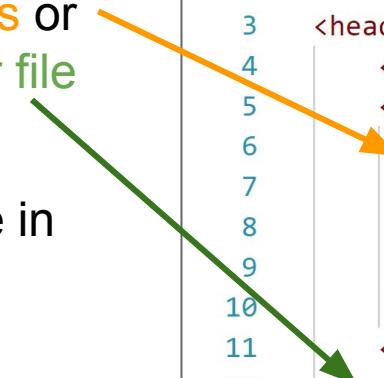
- html elements (like p, h1, etc)
- classes (like class="two")
- id values (like id="main")



```
main.css x
1 p {width: 300px;
2   border: 3px solid green;
3   padding:5px;
4   height:40px;}
5
6 p.two {
7   border-color: purple;}
```

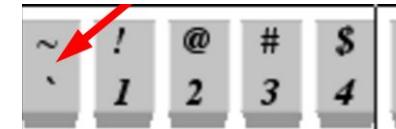
# Overview of JavaScript

- JS inside `<script>` tags or included from another file
- Lines of code execute in order
- In browser can write code to respond to browser events
- Can use document object to work with the DOM



```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Simple JS Example</title>
5      <script>
6          console.log('This is printed to the console');
7          function init() {
8              document.getElementById("output").innerHTML
9                  = new Date();
10         }
11     </script>
12     <script src="main.js"></script>
13 </head>
14 <body onload="init()">
15     <h1>Here is a JS Example Page</h1>
16
17     <div id="output"></div>
18 </body>
19 </html>
```

# JavaScript variables



- Variables can hold values
  - strings can be in double, single, or backtick quotes
  - arrays can hold multiple types of elements

JS variables.js X

```
1  let musicianName = 'Buckethead';
2  let instrument = 'guitar';
3  let bands = [ 'Praxis',
4                "Guns 'N Roses",
5                `Colonel Claypool's "Bucket of Bernie Brains`];
```

# JavaScript objects

- Reference variables (like **aMusician**) can point to an object
- Objects can be defined with curly braces
  - they can have multiple properties that can hold data

JS objects.js

```
1 let aMusician = {  
2   name: 'Les Claypool',  
3   instrument: 'Bass',  
4   bands: ['Primus', 'Oysterhead']  
5 }  
6  
7 console.log(aMusician);  
8  
9 console.log(aMusician.name + ' plays ' +  
10  |   |   aMusician.instrument + ' in ' + aMusician.bands[1]);
```

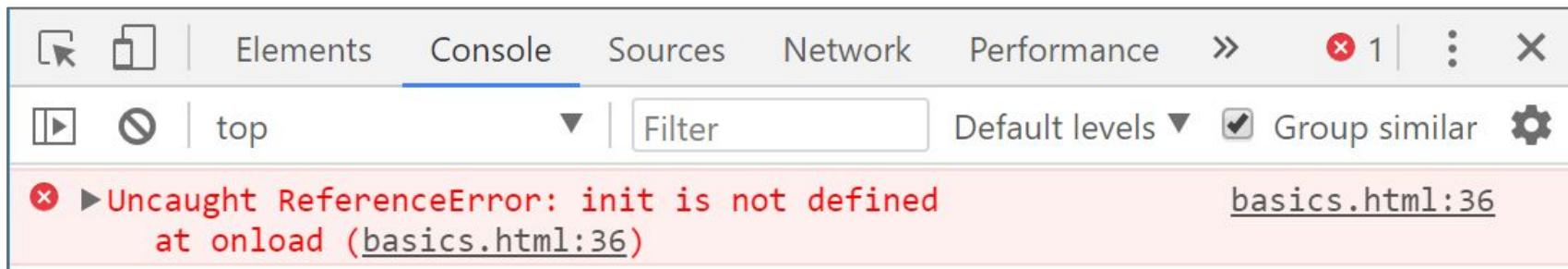
```
{ name: 'Les Claypool',  
  instrument: 'Bass',  
  bands: [ 'Primus', 'Oysterhead' ] }  
Les Claypool plays Bass in Oysterhead
```

# Lab: Create A Basic Web page

20 min

1. Navigate to /Labs/Ch01-MWA\_Node/basics
2. Open the README.md file and follow the steps
  - a. you will be creating a basic webpage
  - b. you will apply some simple styling
  - c. you will create a basic JavaScript function

If your webpage is not working as expected, right click the browser window and choose Inspect. This brings up the dev tools and may point to possible errors



# **Overview of Modern Web Apps & Node**

3 Pillars, Standards, & Guidelines of Web Dev



## **File vs HTTP protocol**

Introducing Node

Starting / Stopping a Node server

AJAX

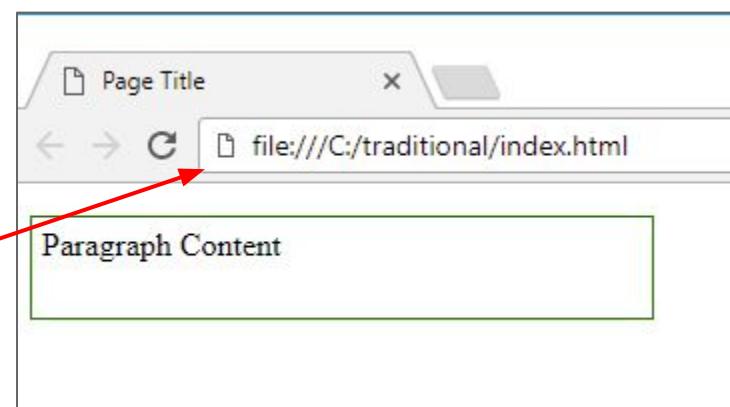
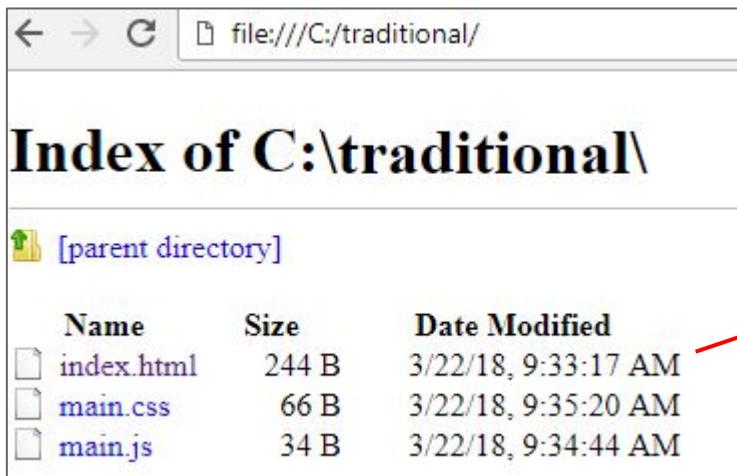
REST API

Using package.json and npm

Overview of Modern Web Apps

# Loading pages directly in the browser

- HTML that uses CSS and JS can be loaded directly into the browser
  - Directly loaded from file system - uses **file://** protocol
  - Paths to directories show contents, need to specify index.html etc



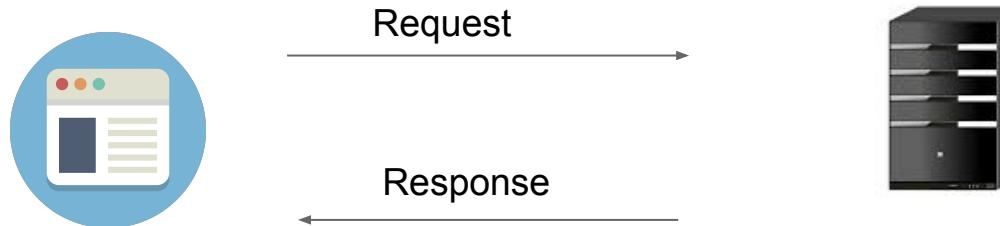
# Certain work requires a server to be running

- Example: You may want to save data to a database
  - Can't do this using **file:// protocol**
- When application is done, we deploy code to a remote, production server
  - It needs to be tested on a local server
- You can run a server locally for development & testing
  - it is often referred to as a **localhost**
  - In most OS <http://localhost> or <http://127.0.0.1> can be used

What is HTTP?

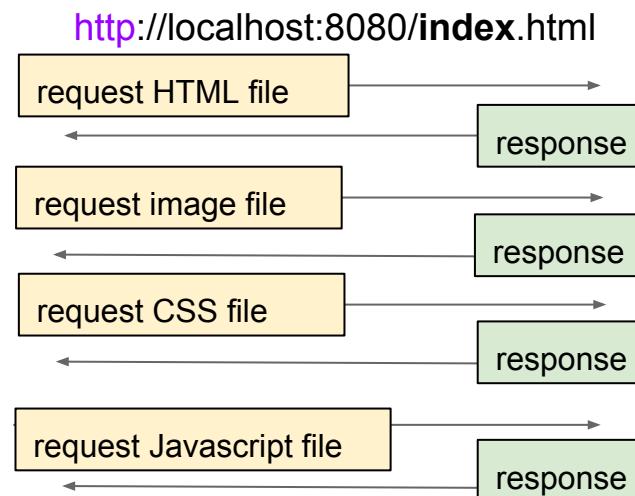
# The HTTP protocol (Hypertext Transfer Protocol)

- A simple transport protocol which runs over TCP/IP
- Client makes an HTTP Request and the server sends an HTTP Response
  - Request and response have a set of headers and an optional body



# Request & Response cycles: get resources from server

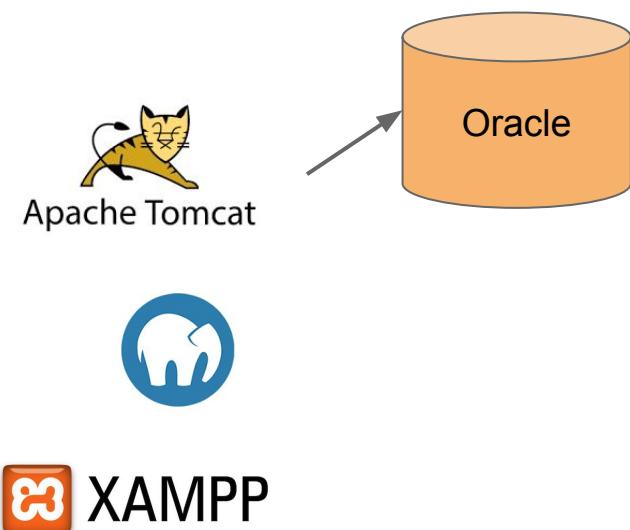
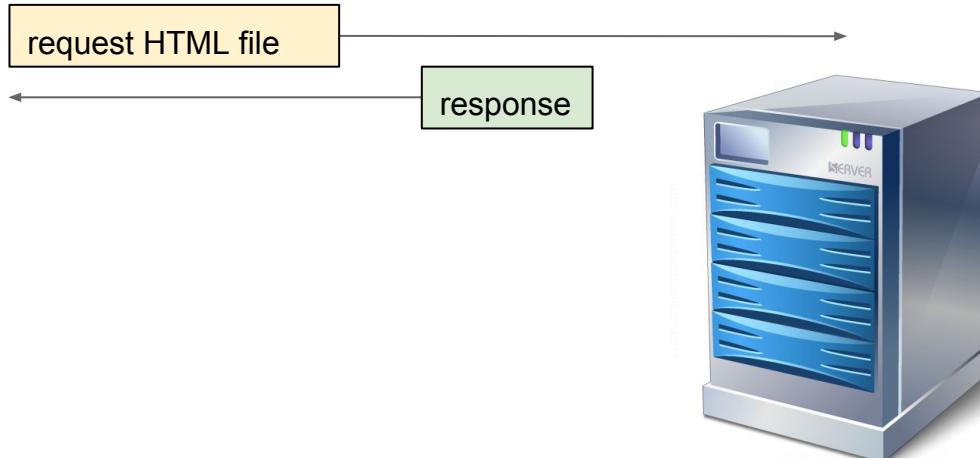
- Browser sends request to server for HTML and receives response
- Parses HTML file, then requests any additional resources
  - Large requests, for many scripts, etc, can mean large responses



# Server-Side Code

- HTML files could be static files simply returned
- HTML could be generated
  - Using Java, .NET, PHP, et al.

<http://localhost:8080/index.html>



# Response Codes and meanings

- HTTP responses have a response **code** and **content** in the response body
- Example response codes:

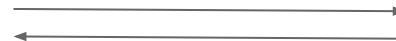
<b>200 OK</b>	Operation successful and content in the body
<b>201 Created</b>	New object created and returned in body
<b>301 Moved Permanently</b>	New URI in Location header
<b>401 Unauthorized</b>	Username and password required
<b>404 Not Found</b>	No content found for URI
<b>5xx</b>	Server unable to respond to request

# Each link/load is an HTTP GET request

- Loading up a page or linking to pages is a GET request



`http://thesite.com/index.html`



`http://thesite.com/resources.html`



`http://thesite.com/contact.html`



# HTTP Methods used with forms

Forms are processed on a server, and then the server, in some way returns a response - **We need a server!**

- **GET** requests place parameters on the URI
  - <http://hostname/resource?key1=value1&key2=value2>
- **POST** pass the parameters in the request body

Search

Search Archive Records?  Yes  No

Minimum Age

/search?archive=y&minage=18

Registration

Username

Password

Email

/register *with data hidden in header*

# **Overview of Modern Web Apps & Node**

3 Pillars, Standards, & Guidelines of Web Dev

File vs HTTP protocol



## **Introducing Node**

Starting / Stopping a Node server

AJAX

REST API

Using package.json and npm

Overview of Modern Web Apps

# Introducing Node



- What is Node?
  - JavaScript outside of the browser
  - Based on Chrome's V8 JavaScript engine
- We will be using Node to:
  - setup a local server
  - practice with modern web development tools

# Where does Node run?

- All of the popular operating systems
  - Windows, Linux, Mac OS X
- Can download the source to build it on other platforms
- Website includes installers for Windows and Mac OS X systems
- Also available on GitHub

# Installation of Node

<http://nodejs.org>

- Download LTS or latest
- wizard, default settings
- Installs Node and npm

The screenshot shows the official Node.js website at <http://nodejs.org>. The header features the Node.js logo and navigation links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. A green call-to-action button labeled "March 2018 Security Releases" is prominently displayed. Below it, two large green buttons offer "Download for Windows (x64)" for versions 8.10.0 LTS (Recommended For Most Users) and 9.9.0 Current (Latest Features). At the bottom, links for "Other Downloads | Changelog | API Docs" are visible on both sides.

# Executing JavaScript using Node from command line

- **node nameOfProgram.js** (with or without the .js extension)
  - simple programs return to command prompt

hello.js

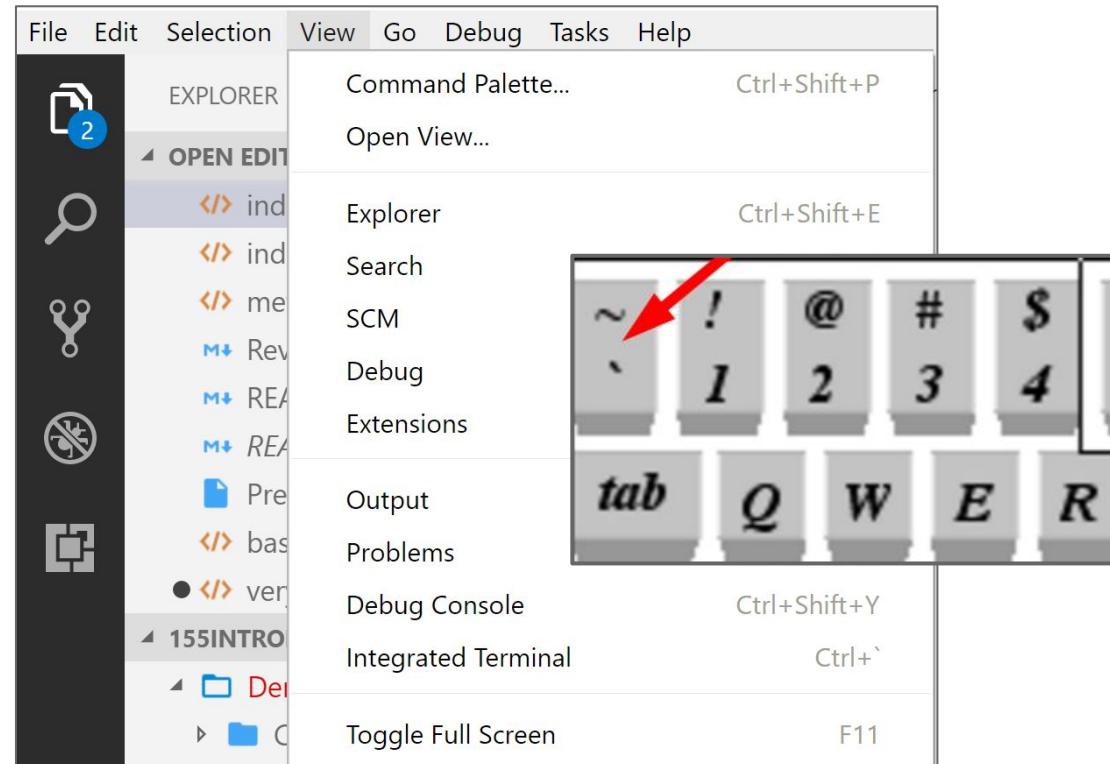


```
1 //open the directory from a command window or terminal
2 //type node hello.js
3 console.log('Hello from Node.');
```

```
Judys-MBP:Node judylipinski$ node hello.js
Hello from Node.
Judys-MBP:Node judylipinski$ node hello
Hello from Node.
```

# VSCODE has an Integrated Terminal

- You do not need to open an external command prompt or terminal
- Can open a terminal window from File | View menu, or use Ctrl + backtick (tilda sans shift)

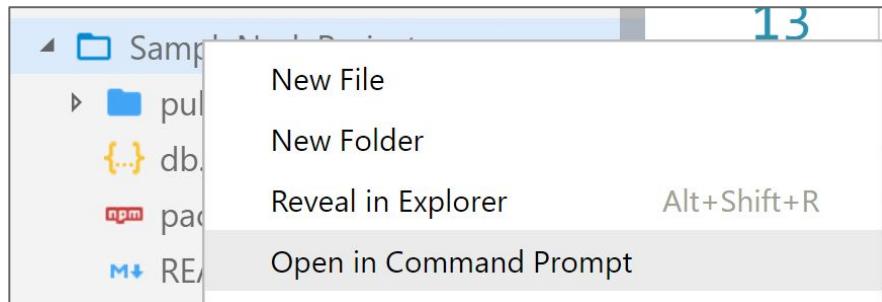


# HINT! Opening Terminal at correct location

TIP: You **must** be in the correct location for commands to find files

VSCode makes this easy

- right-click desired folder or file in that folder
- choose **open in command prompt** for the exact directory location



# DO NOW: Execute JavaScript outside the browser

5 min

1. Navigate to /Demos/Chapter01-MWA\_Node/HelloNode
2. Follow the steps in the README.md
3. You will practice:
  - a. Opening terminal/prompt at this directory level
  - b. Execute a simple JavaScript program

# **Overview of Modern Web Apps & Node**

3 Pillars, Standards, & Guidelines of Web Dev

File vs HTTP protocol

Introducing Node

## **Starting / Stopping a Node server**

AJAX

REST API

Using package.json and npm

Overview of Modern Web Apps

# Node has a built in http package

- Servers listens on a single port number
  - Only 1 server can listen to a specific port at one time

The screenshot shows a terminal window titled "serverExample.js" with the following code:

```
1 const http = require('http');
2
3 const hostname = '127.0.0.1';
4 const port = 3000;
5
6 const server = http.createServer((req, res) => {
7   res.statusCode = 200;
8   res.setHeader('Content-Type', 'text/plain');
9   res.end('Hello World\n');
10 });
11
12 server.listen(port, hostname, () => {
13   console.log(`Server running at http://\$\${hostname}:\$\${port}/`);
14 });
```

To the right of the code, the terminal output is shown in a green-bordered box:

Chapter01-MWA\_Node\NodeServer> **node** .\serverExample.js  
Server running at <http://127.0.0.1:3000/>

A yellow callout box on the right side of the slide contains the text:

...because it is listening,  
the process doesn't  
return to prompt

# Common Error: EADDRINUSE

- If you try to start a server on a **port already being used** you will get an error

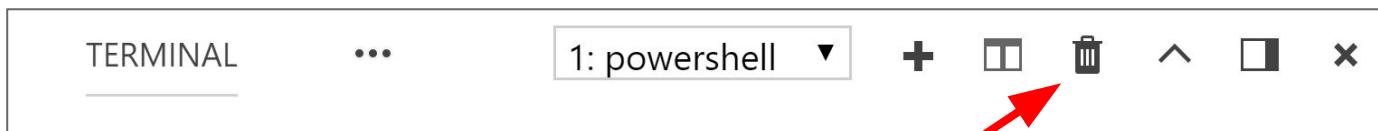
```
Chapter01-MWA_Node\NodeServer> node .\serverExample.js
events.js:183rap_node.js:608:3
    throw er; // Unhandled 'error' event
          ^
Error: listen EADDRINUSE 127.0.0.1:3000
    at Object._errnoException (util.js:1024:11)
    at _exceptionWithHostPort (util.js:1046:20)
    at Server.setupListenHandle [as _listen2] (net.js:1351:14)
    at listenInCluster (net.js:1392:12)
    at doListen (net.js:1501:7)
    at _combinedTickCallback (internal/process/next_tick.js:14
```

# Stopping a server

- Kill process with CTRL-C - and it will return you to the prompt

```
Chapter01-MWA_Node\NodeServer> █
```

- Hitting the trash can icon in VSCode also can kill the process
  - But also closes the integrated terminal window
  - (Hitting X just hides the window, but does not kill the process)



# **DO NOW: Start and stop a server**

**5 min**

1. Navigate to /Demos/Chapter01-MWA\_Node/NodeServer
2. Follow the steps in the README.md
3. You will practice:
  - a. Starting and stopping a server

# **Overview of Modern Web Apps & Node**

3 Pillars, Standards, & Guidelines of Web Dev

File vs HTTP protocol

Introducing Node

Starting / Stopping a Node server



**AJAX**

REST API

Using package.json and npm

Overview of Modern Web Apps

# Inefficient form validation

- This form has been validated by submitting to the server, getting a response, and updating the screen



Why is this a bad practice?



What could be done instead?

Registration

Username	<input type="text" value="vivymepa"/> <i>That username is already being used</i>
Password	<input type="password"/> <i>That password does not meet criteria</i>
Email	<input type="text" value="gytideo@gmail.com"/> <i>That email is already being used.</i>

Username and email can be checked using AJAX

Password can be validated client-side using JavaScript - *no trip to server*

# Asynchronous JavaScript and XML (AJAX)

- Request & Response happen without a page refresh
  - Users can still move cursor, interact with screen
  - JS code executes asynchronously behind the scenes
  - Servers recognize the incoming requests and return a response
- Data historically was returned using XML
  - Today a variety of formats are supported



What data format is more prominent these days?

JSON

# JavaScript Object Notation (JSON)

- Recall, objects have properties that can hold data

JS objects.js x

```
1 let aMusician = {  
2     name: 'Les Claypool',  
3     instrument: 'Bass',  
4     bands: ['Primus', 'Oysterhead']  
5 }  
6  
7 console.log(aMusician.name + ' plays '  
8             aMusician.instrument + ' in ' + bands[0]);
```

JavaScript Object Literal

JSON package.json x

```
1 {  
2     "name": "server-form-validation",  
3     "version": "1.0.0",  
4     "description": "Demo showing server form validation",  
5     "main": "index.js",  
6     "scripts": {  
7         "test": "echo \"Error: no test specified\" && exit 1"  
8     },  
9     "author": "Judy Lipinski",
```

JSON

- JSON notation is similar but requires quotes around properties
  - Used when sending data to/from server, config files, et al.

# No longer need to wait for page to submit

- Username and email can be checked **asynchronously**  
- *helps user stay in context*

What may trigger the request to the server?

An event such as  
onblur or on keyup

—Registration—

Username	<input type="text" value="rukoxet"/>
<i>That username is already being used</i>	
Password	<input type="password"/>
Email	<input type="text"/>
<input type="button" value="Submit"/>	

isUserNameUnique?

{ "isUnique": false }



# The Beginning of Modern JavaScript

- AJAX caused a resurgence in interest in JavaScript
  - Can be done in pure JavaScript
  - Many tools/libraries emerged to make it “easier” to do AJAX



# **Overview of Modern Web Apps & Node**

3 Pillars, Standards, & Guidelines of Web Dev

File vs HTTP protocol

Introducing Node

Starting / Stopping a Node server

AJAX



## **REST API**

Using package.json and npm

Overview of Modern Web Apps

# AJAX calls are typically requests to do something

- GET data
  - such as using form data to search for records
  - is username or email unique?
  - get next 100 rows of data
  - refresh data every X seconds
- POST data
  - such as taking form input and add new data to server database

# **Servers respond to the incoming requests**

- Servers are listening for incoming GET and POST requests
- Various technologies can be used for the servers:
  - Java
  - .Net
  - PHP
  - Node

# The REST-based API

- Using HTTP Methods, work is performed on the server (CRUD)
    - GET & POST as before
    - PUT or PATCH update content on the server
    - DELETE the content on the server
  - Along with a meaningful URL (**endpoint**)
    - not an html file, but a route the server is set up to recognize
- create read update delete
- ```
DELETE /band/1 200 4.260 ms - 2
GET /band 200 1.180 ms - -
GET /band/3 200 1.084 ms - 179
PUT /band/3 200 32.896 ms - 187
GET /band 200 0.673 ms - -
POST /band 201 3.493 ms - 135
GET /band 200 0.679 ms - -
```

# **Overview of Modern Web Apps & Node**

3 Pillars, Standards, & Guidelines of Web Dev

File vs HTTP protocol

Introducing Node

Starting / Stopping a Node server

AJAX

REST API



**Using package.json and npm**

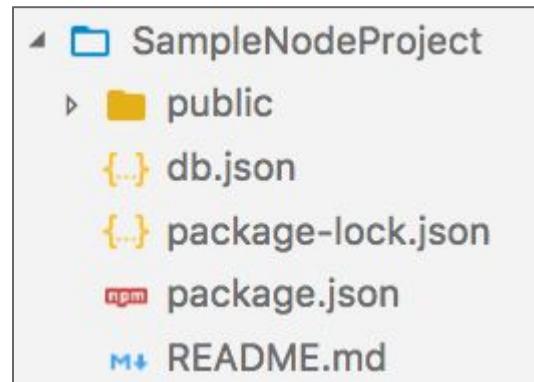
Overview of Modern Web Apps

# Using packages

- In Modern Web Dev, we leverage libraries / tools / software to:
  - start, stop & restart local servers for development
  - automatically update the browser
  - pre-process SASS/LESS
  - perform automated testing
  - bundle code
- In Modern development the term we use for these tools is  
**package**

# Project organization

- Projects contain files to help organize and manage dependencies
  - README.md (*helpful info on the project*)
  - package.json (*manages dependencies and scripts*)
  - package-lock.json (*track changes to package.json & conflicts*)



# The package.json is in the root of our project directory

- It gives metadata about the project
  - which tools we are depending on
  - script to execute processes: build, test, etc.
- Benefits of package.json
  - Easy to see the project's dependencies
  - Specifies versions of a package using semantic versioning rules
  - Makes build reproducible and easier to share with other developers

Don't want to say to others, "to run my program, go download these 10 packages..."

# Dev and production dependencies can be specified

- **"dependencies"**: required by your application in production as well as dev
- **"devDependencies"**: only needed for development and testing, not production

```
npm package.json ×  
1  {  
2    "name": "node-demo",  
3    "version": "1.0.0",  
4    "description": "Demonstration of working with Node",  
5    "main": "index.js",  
6    "scripts": {  
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"  
8    },  
9    "author": "",  
10   "license": "ISC",  
11   "dependencies": {  
12     "jquery": "^3.3.1"  
13   },  
14   "devDependencies": {  
15     "jasmine": "^3.1.0",  
16     "karma": "^2.0.0"  
17   }  
18 }
```

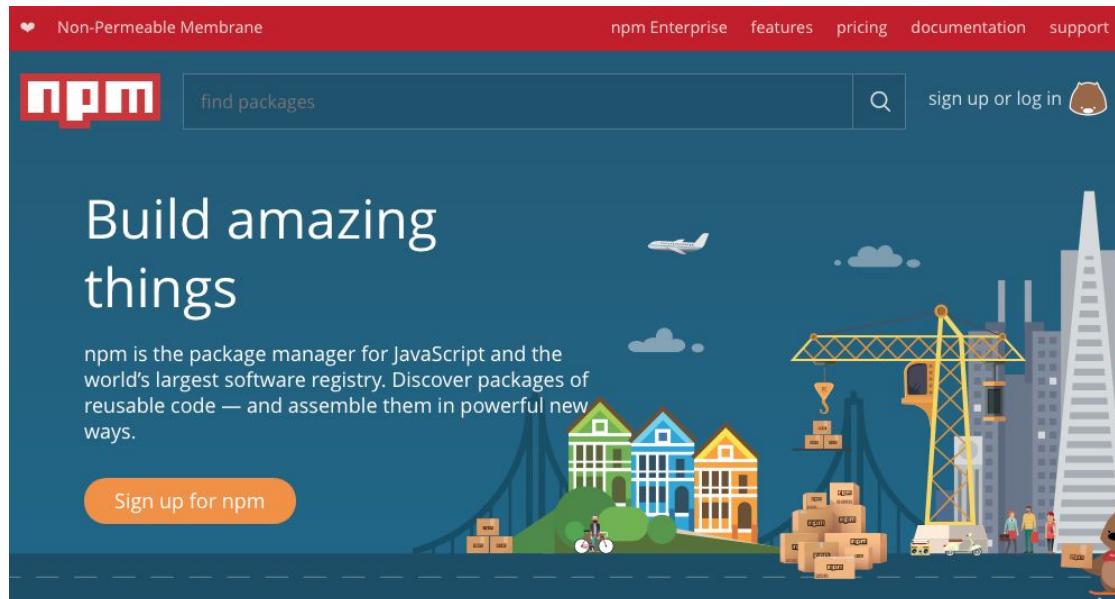
~ : patch release - 1.0.x  
^ : minor release - 1.x.0  
\* : major release - x.0.0

# Basics of Semantic Versioning (Semver)

- If a project is going to be shared it should start at 1 . 0 . 0
- Change the correct number to indicate the type of change
- Bug fixes / minor changes / patch release
  - increment the last number, e.g. 1.0.**1**
- New features that do not break existing
  - Minor release, increment the middle number, e.g. 1.**1**.0
- Changes which break backwards compatibility
  - Major release, increment the first number, e.g. **2**.0.0

# The npm registry is used to store dependencies

- The npm commands can be used to download and install dependencies
  - Corporations can configure local repositories and use proxies



# Working with Node projects and dependencies

- Node project dependencies are not pushed into the repo

```
 .gitignore ✘  
1 | # Dependency directories  
2 | node_modules/
```

- When you git a project out of a repo, you still need the dependencies
  - Get the dependencies with this command:  
**npm install**
- Reads package.json and downloads dependencies into /node\_modules

# Executing Scripts from package.json

- package.json can also contain scripts
  - to build the front-end, launch servers in dev mode, and more...
- Execute the scripts using:  
`npm run script-name`
- Here it would be  
`npm run start`
- start is common, so can use:  
`npm start`



A screenshot of a code editor showing a package.json file. The file contains the following JSON code:

```
1 | {  
2 |   "name": "samplenodeproject",  
3 |   "version": "1.0.0",  
4 |   "description": "Demonstrates working with node",  
5 |   "main": "index.js",  
6 |   "scripts": {  
7 |     "start": "json-server -p 3008 -w db.json"  
8 |   },  
9 |   "author": "Judy Lipinski",  
10 |  "license": "ISC",  
11 |  "dependencies": {  
12 |    "json-server": "^0.12.1"  
13 |  }  
14 | }
```

# DO NOW: Execute a Node project

15 min

1. Navigate to /Demos/Chapter01-MWA\_Node/SampleNodeProject
2. Open the README.md file and follow the steps
  - a. you will be obtaining dependencies
  - b. you will start a project with a client-side and back-end server

**Caution: Always be mindful where you are running your commands from!**

# **Overview of Modern Web Apps & Node**

3 Pillars, Standards, & Guidelines of Web Dev

File vs HTTP protocol

Introducing Node

Starting / Stopping a Node server

AJAX

REST API

Using package.json and npm



## **Overview of Modern Web Apps**

# Traditional web applications

- Every page/view is a separate request / response cycle
  - Simple pages can be fast
  - Each page takes time to load. Large requests, large responses
- HTML Code may be generated using Java, .NET, PHP, etc.



# Most of today's websites are inefficient

- Unnecessary network requests:
  - validations
  - separate requests for each view
- Every call to the server takes time
  - possible generation of HTML using server processes
- Users want applications to feel more responsive

# Single Page Applications

- Web frameworks and libraries that are very popular now, focus on Single Page Applications (SPA)
- Mini-applications running in the browser, with CRUD performed using AJAX
- Only 1 index.html, with views/templates all loaded into browser to start



# Making JavaScript Better

- Older, plain JavaScript is just plain messy.
  - Tools emerged to make it easier

TypeScript

 CoffeeScript

- In recent years the JS Standard has been moved forward

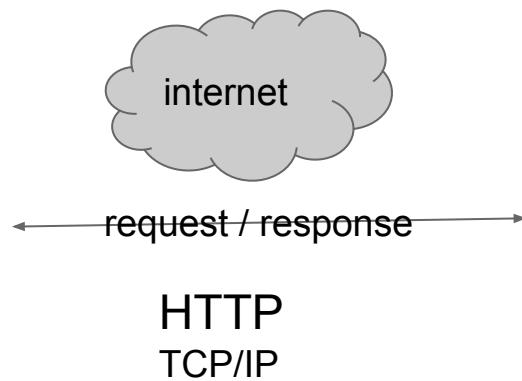


# Front-End / Back-End / Full-Stack



## Front End technologies

- HTML, CSS, JavaScript
- Bootstrap, LESS, Sass
- jQuery
- React, Ember, Angular



## Back End technologies

- PHP, Java (Spring Boot)
- .NET
- Node (JS)
- MongoDB(JS), Oracle, MySQL

# Chapter Summary

In this chapter we have looked at:

- Brief overview of HTML5, CSS3, and JavaScript
- Web Accessibility Guidelines
- Why do we need servers?
- An overview of AJAX and REST APIs
- What is Node?
- Managing project dependencies and automating tasks in Modern Web Dev
- Popular libraries and frameworks for SPA applications

# Modern Web Development

## Chapter 2

Working with Browsers



# Chapter Objectives

In this chapter we will look at:

- Accessing browser developer tools
- DOM inspection and manipulation
- JavaScript debugging
- Feature detection in browsers
- Using shims, shivs, polyfills

# Working with Browsers



## Browser Developer Tools

- JavaScript debugging

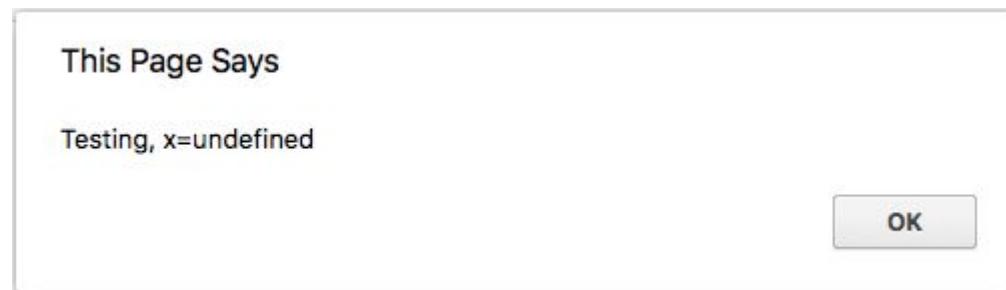
- Feature detection in browsers

- Using shims, shivs, polyfills

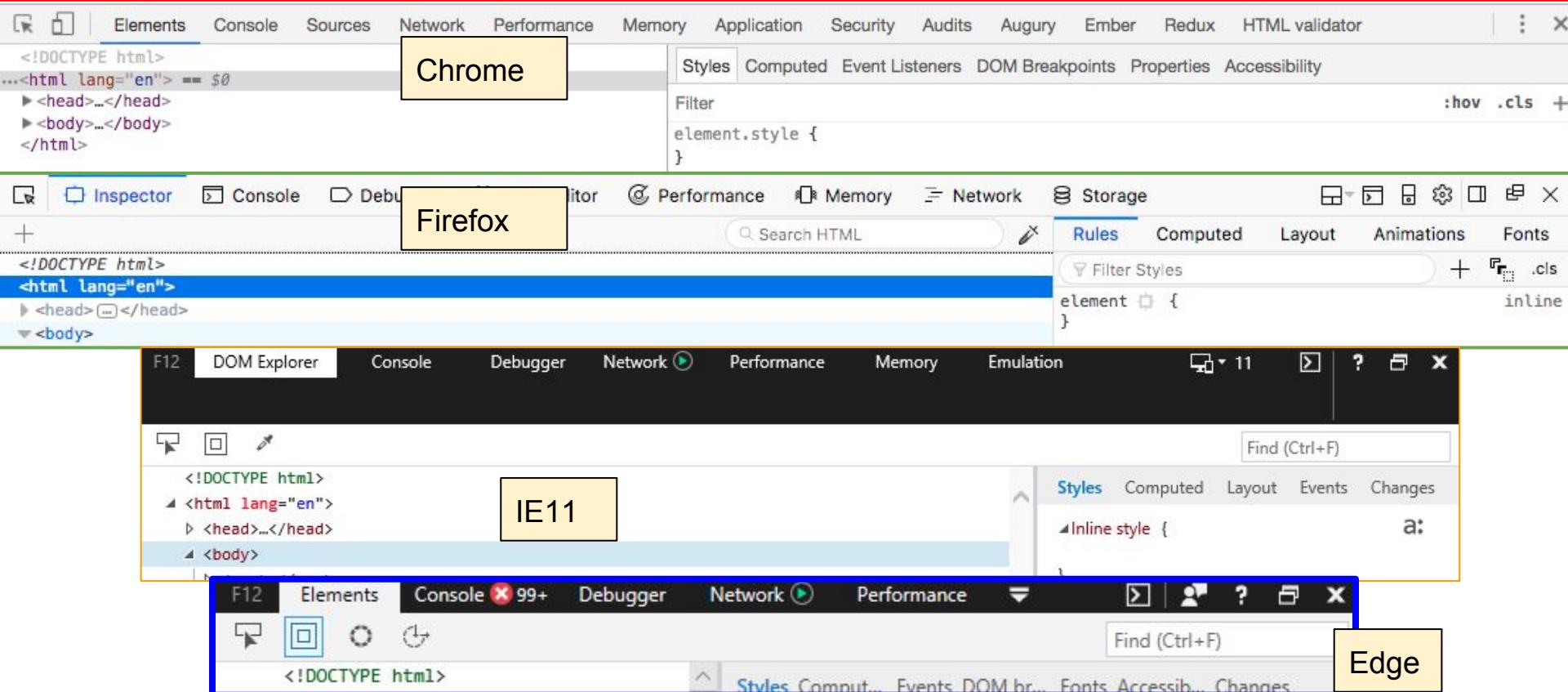
# Historical debugging prior to browser developer tools

- `window.alert()` was used for accessing state of variables

```
<script>
|   alert('Testing, x=' + x);
</script>
```

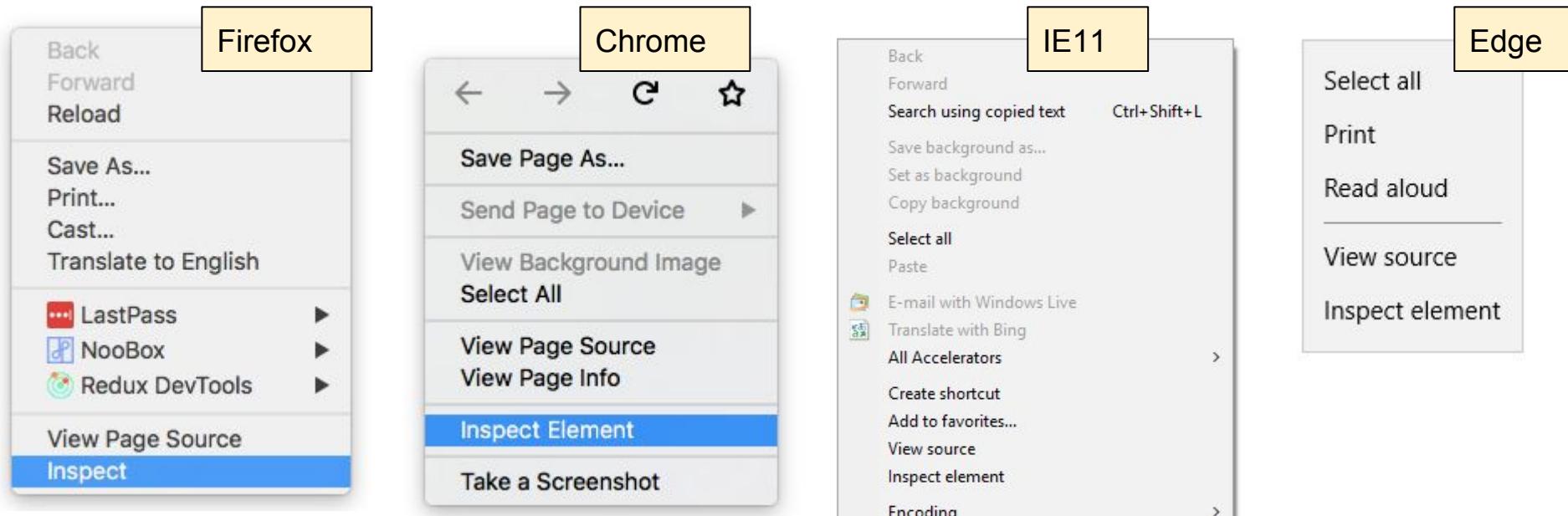


# Firebug was first plugin, now all browsers include devtools



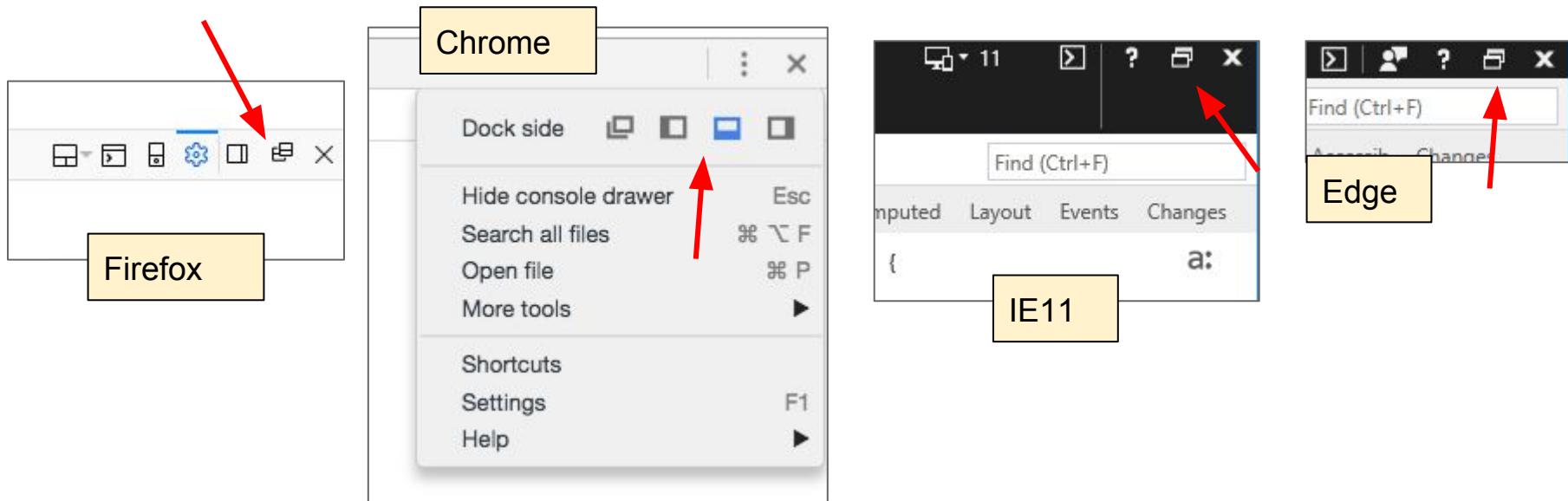
# Opening Developer Tools

- Can invoke using Right-click context menu or other shortcuts
  - F12, Ctrl-Shift-I, Option-Command-I



# Display of Developer Tools

- Can use right-side of console menus to arrange for display of tools
  - As separate window, right, left or bottom



# Elements / Inspector for DOM (HTML) and CSS

- Displays DOM and applied CSS
- Can click the inspector icon and then navigate via display or DOM
  - Applied CSS is then shown

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The main area displays the DOM tree:

```
<!DOCTYPE html>
<html lang="en">
  <head>...
  <body>
    <nav>...
    <main>
      <h1>Welcome to class</h1> == $0
      <p class="important">An important paragraph
        </p>
    </main>
  </body>
</html>
```

Three numbered arrows point to specific features:

- Red arrow pointing to the 'Elements' tab in the toolbar.
- Orange arrow pointing to the 'h1' element in the DOM tree.
- Purple arrow pointing to the 'Styles' panel on the right, which shows the applied CSS for the 'h1' element.

The 'Styles' panel content is as follows:

Styles	Computed	Event Listeners	DOM Breakpoints	Properties	Accessibility
--------	----------	-----------------	-----------------	------------	---------------

Filter: element.style { } h1 { color: purple; text-decoration-line: underline overline; }

# With Inspect you can see and modify the code

- You are able to view the HTML, and double-click to modify it

```
<input type="password" class="inputtext" name="pass" id="pass" tabindex="2" data-testid="royal_pass"
autocomplete="off" style="background-image: url("data:image/png;
base64, iVBORw0KGgoAAAANSUhEUgAAABAAAAASCAYAAABS015qAAAAAXNSR0IArs4c6QAAAxtJREFUOBGVUj2LwkAU
nIjBICIJK
e8nWKRKv2Ru/YYMn1N1Yw/73an1R/
```



```
<input type="text" class="inputtext" name="pass" id="pass" tabindex="2" data-testid="royal_pass"
autocomplete="off" style="background-image: url("data:image/png;
base64, iVBORw0KGgoAAAANSUhEUgAAABAAAAASCAYAAABS015qAAAAAXNSR0IArs4c6QAAAxtJREFUOBGVUj2LwkAU
nIjBICIJK
e8nWKRKv2Ru/YYMn1N1Yw/73an1R/
```

- By changing input type to **text**, the browser displays the “hidden” password...



Email or Phone      Password

winegoddess@gmail.com      notmypassword

Forgot account?

# You can add and modify CSS as well

- For selected HTML element, you can add to **element.style**
- If a selected rule is already being applied, you can:
  - uncheck boxes to turn off
  - modify properties
  - add additional properties

The screenshot shows the developer tools' styles panel. The tab bar at the top has 'Styles' selected, followed by 'Computed', 'Event Listeners', and 'DOM Breakpoint'. Below the tabs is a 'Filter' input field. The main area displays a list of CSS rules under the heading 'element.style {'. One rule, 'color: ;', has its color swatch highlighted in blue, indicating it is currently selected or being edited. Other rules listed include 'caption-side', 'hticaret-color', 'clear', 'clip', 'hticlip-path', 'color' (which is also highlighted in blue), and 'column-count'.

```
element.style {
    color: ;
}
caption-side
hticaret-color
clear
clip
hticlip-path
color
column-count
```

The screenshot shows the developer tools' styles panel. The tab bar at the top has 'Styles' selected, followed by 'Computed', 'Event Listeners', and 'DOM Breakpoints'. Below the tabs is a 'Filter' input field. The main area displays a list of CSS rules under the heading 'element.style {'. Another set of rules is shown under 'h1 {'. The 'color' property for 'h1' has two options: a checked checkbox with a purple square swatch, and an unchecked checkbox with a red square swatch. The 'text-decoration-line' property for 'h1' has a checked checkbox with a red square swatch.

```
element.style {
}
h1 {
    color: purple;
    text-decoration-line: underline overline;
}
```

# Lab: Use Chrome devtools to modify styles

10 min

1. In VSCode, navigate to /Labs/Ch02-Browsers/1devtools
2. Open the README.md file and follow the steps

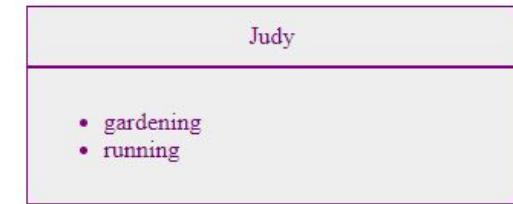
The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. A red arrow points to the refresh icon in the top-left corner of the DevTools window. The left pane displays the HTML DOM structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body onload="init()">
    <div id="name">Judy</div> == $0
    <div id="hobbies">...</div>
  </body>
</html>
```

The right pane shows the 'Styles' tab of the style editor. It contains the following CSS code:

```
element.style {  
}  
  
div {  
  width: 300px;  
  padding: 10px;  
  border: 1px solid purple;  
  background-color: #eeeeee;  
  color: purple;  
}
```

A preview panel on the right shows a box with the name 'Judy' and a bulleted list of hobbies: 'gardening' and 'running'.



# Working with Browsers

Browser Developer Tools



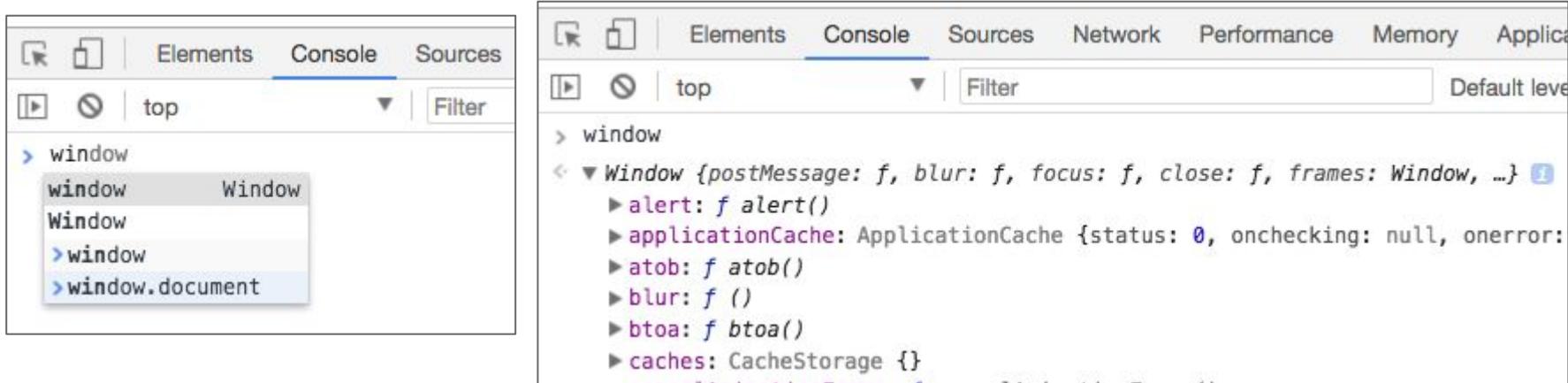
**JavaScript debugging**

Feature detection in browsers

Using shims, shivs, polyfills

# Displaying variables in the console tab

- The browser maintains a global object called **window**
  - Our variables and functions get added to **window**
- You can access variables from memory using the console
  - For example, can type **window** and hit return
  - The window object and its properties are displayed



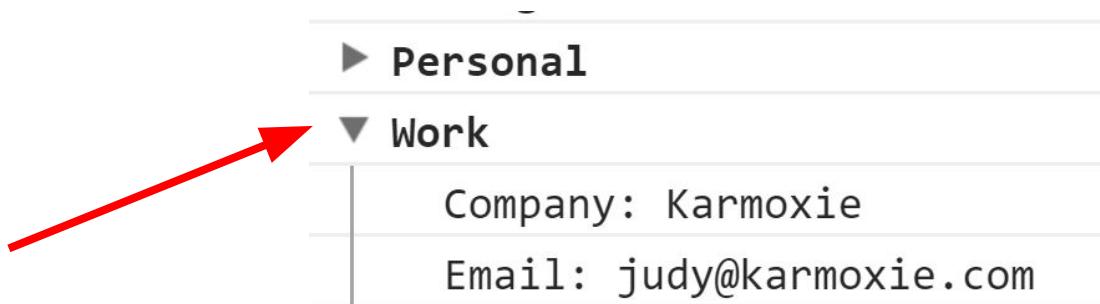
The screenshot shows the Chrome DevTools interface with the "Console" tab selected. On the left, the "Elements" and "Sources" tabs are visible above the "Console" tab, which is highlighted with a blue underline. Below the tabs, there are icons for "Elements" (a magnifying glass), "Console" (a speech bubble), and "Sources" (a document). The "Console" tab has its own set of icons: a play button, a stop button, and a "Filter" input field. The main area displays the output of a command entered in the console. The command was "window", and the result is a detailed object representation of the Window object. The object has several properties listed: "postMessage: f", "blur: f", "focus: f", "close: f", "frames: Window", "...". Below this, specific methods are shown: "alert: f alert()", "applicationCache: ApplicationCache {status: 0, onchecking: null, onerror: ...}", "atob: f atob()", "blur: f ()", "btoa: f btoa()", and "caches: CacheStorage {}". The entire object representation is preceded by a greater than sign (>).

```
> window
< ▶ Window {postMessage: f, blur: f, focus: f, close: f, frames: Window, ...} ⓘ
  ► alert: f alert()
  ► applicationCache: ApplicationCache {status: 0, onchecking: null, onerror: ...}
  ► atob: f atob()
  ► blur: f ()
  ► btoa: f btoa()
  ► caches: CacheStorage {}
```

# Grouping messages in console.log()

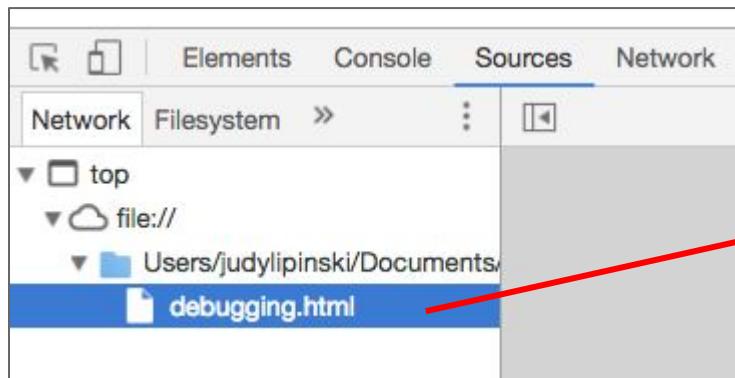
- Group messages using console.group - creates disclosure arrows

```
console.group('Work');
console.log('Company: Karmoxie');
console.log('Email: judy@karmoxie.com');
console.groupEnd();
```



# Set breakpoints and use debugging menu

- Setting breakpoints can be helpful in debugging code
- In Chrome, click the Sources tab, have to click the already selected file
  - Then can click line number to set breakpoints

A screenshot of the Chrome DevTools Sources tab. The tab bar shows 'Elements', 'Console', 'Sources' (which is active), 'Network', and 'Performance'. Below the tabs, there's a tree view of files under 'Network' and 'Filesystem'. Under 'Filesystem', there's a folder 'Users/judylipinski/Documents/' which contains a file named 'debugging.html'. The file 'debugging.html' is selected and highlighted with a blue background. The code editor on the right shows the HTML and JavaScript content of the file. Line 12 of the JavaScript code is highlighted with a blue background, and a red arrow points from the 'debugging.html' file in the tree view on the left towards this line in the code editor.

# Using debugging menu

- You may need to refresh the page to stop at a breakpoint
- Once stopped you can use the debugging menu

The screenshot shows a browser developer tools window with the title bar "debugging.html x". The main area displays the following HTML and JavaScript code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>Show breakpoints</title>
8   <script>
9     var numberArray = [1,33,2,5,22];
10
11    function sortNums() {
12      numberArray.sort( function(a,b) {
13        return a-b;
14      });
15
16      document.getElementById("sortedArray").innerHTML = numberArray;
17
18    }
19
20  </script>
21
22 </head>
23 <body>
24   <h1>Show breakpoints being used</h1>
25   <p>Open the dev tools and set a breakpoint in the sortNums function</p>
26   <p>Starting array: <span id="originalArray"></span></p>
27   <button onclick="sortNums()">Run Code</button>
```

The script is paused at line 12, where the `sort` method is called. The browser's debugging menu is open on the right, showing the following options:

- Paused on breakpoint
- Watch
- Call Stack
- sortNums
- onclick
- Scope
- Local
  - this: Window
- Global
- Breakpoints
  - debugging.html:12 (checkbox checked)
  - numberArray.sort( function(a,b) {
- XHR/fetch Breakpoints
- DOM Breakpoints
- Global Listeners
- Event Listener Breakpoints

continue (also F8)

step over

step into

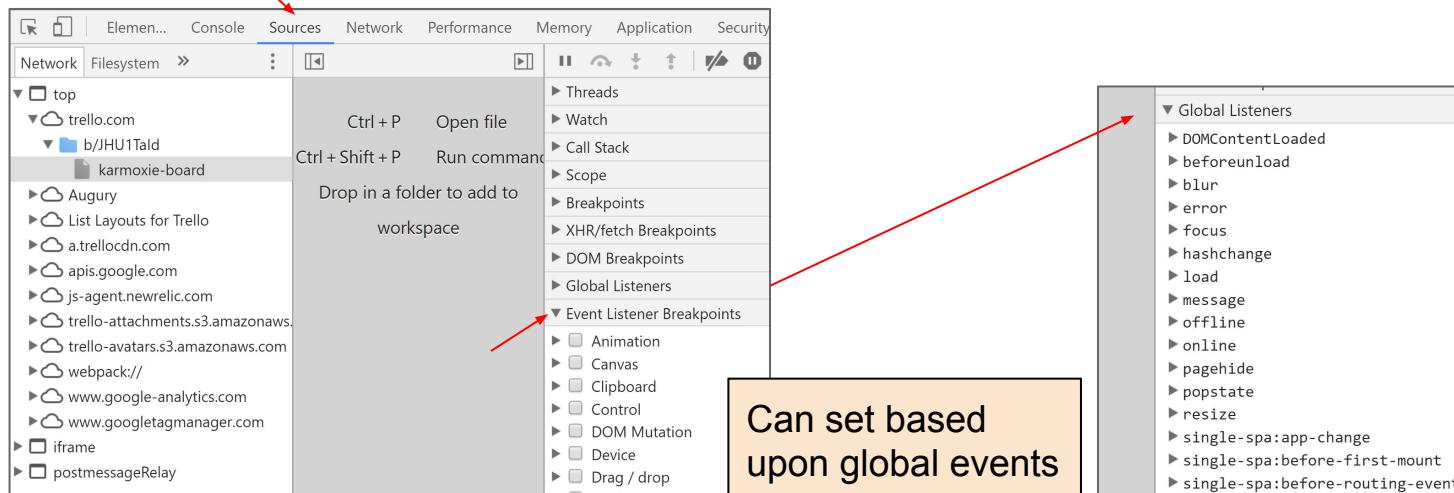
step out of

deactivate  
breakpoints

pause on exceptions

# Triggering breakpoints in response to other criteria

- Perhaps you have a lot of scripts
- Not sure which code to add breakpoints to
- Breakpoints can trigger based upon events, such as clicking a radio button



# Lab: Use debugging tools

10 min

1. In VSCode, navigate to /Labs/Ch02-Browsers/2debug
2. Open the README.md file and follow the steps

The screenshot shows the Chrome DevTools interface with the 'Sources' tab selected. The left sidebar shows the project structure with 'debugging.html' selected. The main pane displays the HTML code for 'debugging.html'. A blue horizontal bar highlights the line of code where a breakpoint is set: 'numberArray.sort( function(a,b) {'. The right sidebar shows the DevTools controls and the current state: 'Paused on breakpoint'.

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>Show breakpoints</title>
<script>
var numberArray = [1,33,2,5,22];
function sortNums() {
    numberArray.sort( function(a,b) {
        return a-b;
    }) ;
    document.getElementById("sortedArray").innerHTML = numberArray;
```

Paused on breakpoint

Watch

Call Stack

sortNums debugging.html:12

onclick debugging.html:27

Scope

Local

# Working with Browsers

Browser Developer Tools

JavaScript debugging



**Feature detection in browsers**

Using shims, shivs, polyfills

# IE Conditional Comments

Note: Microsoft has dropped support for IE9 and 10

- Target specific versions, to include HTML, scripts or workarounds

```
2 <!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
3 <!--[if IE 7]>        <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
4 <!--[if IE 8]>        <html class="no-js lt-ie9"> <![endif]-->
5 <!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
```

```
14 <!--[if lt IE 7]>
15     <p class="browsehappy">You are using an
16         | <strong>outdated</strong> browser. Please
17         | <a href="#">upgrade your browser</a> to improve your experience.
18     </p>
19 <![endif]-->
```

- Greater than/less than IE conditionals

[if gt IE 6] [if lt IE 9] [if gte IE 8] [if lte IE 7]

# Not all Browsers function the same...yet

- Browsers have gotten much better in implementing standard features
- <https://html5test.com/compare/browser/mybrowser/edge/firefox/ie-11/ie-10.html>

BROWSERS		528	494	484	312	265
		My browser	Edge	Firefox	Internet Explorer 11	Internet Explorer 10
parsing rules		5	5	5	5	5
<!DOCTYPE html> triggers standards mode		Yes ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓
HTML5 tokenizer		Yes ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓
HTML5 tree building		Yes ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓
Parsing inline SVG		Yes ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓
Parsing inline MathML		Yes ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓

# Checking for Browser Support

- Many sites track features and support in browsers
  - For CSS, HTML5 and related APIs
- Helpful sites show support in both desktop and mobile browsers
  - With version histories
- Some tools:
  - <https://html5test.com/>
  - <http://html5please.com/>
  - <http://mobilehtml5.org/>
  - <https://caniuse.com/>

# Avoid Browser (UA) Sniffing aka multi-browser support



Why is this code not effective?

```
12  function detectBrowser() {  
13      var browserDisplay = document.getElementById("browserDisplay");  
14      var browserString = '';  
15      var ua = navigator.userAgent;  
16  
17      if(ua.indexOf('Firefox') !== -1) {  
18          // run Firefox-specific code  
19          browserString = 'This is Firefox';  
20      } else if(ua.indexOf('Chrome') !== -1) {  
21          // run Chrome-specific code  
22          browserString = "This is Chrome";  
23      } else {  
24          browserString = "This is not Chrome or Firefox";  
25      }  
26      browserDisplay.innerHTML = browserString;  
27  }
```

Implementations of features varies by version number.

Would have to keep updating the code

Users can modify their own UA string!

# Browser support can vary between versions

- It would be difficult to maintain code
- <http://moxie.pro/comparefeatures>

FEATURES					
	Device Motion	Arrow functions	input type=date	Text-level semantic elements	Basic socket communication
Select up to five features and immediately see how well it is supported by each browser	No ✕	No ✕	Yes ✓	Yes ✓	Yes ✓
desktop browsers	Yes ✓	No ✕	Yes ✓	Yes ✓	Yes ✓
Chrome 30	No ✕	No ✕	Yes ✓	Yes ✓	Yes ✓
Chrome 40	Yes ✓	No ✕	Yes ✓	Yes ✓	Yes ✓
Chrome 45	Yes ✓	Yes ✓	Yes ✓	Yes ✓	Yes ✓

# Status Website from Microsoft

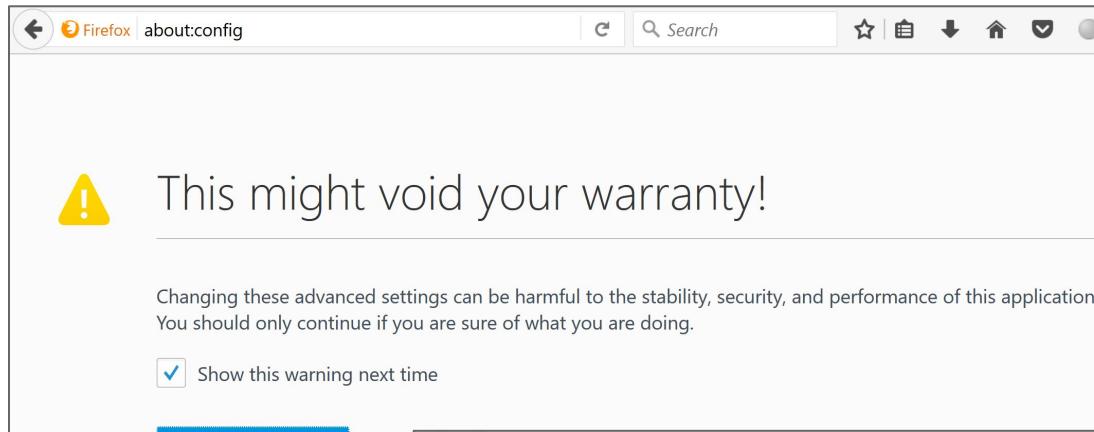
- <https://developer.microsoft.com/en-us/microsoft-edge/platform/status/color-input-type>

The screenshot shows the Microsoft Edge Platform Status page for the 'Color input type' feature. At the top, there's a section for 'Color input type' with a keyboard icon, a 'SUPPORTED' button, and a note about build number 14316+. Below this, there's a detailed browser support table.

Browser	Status
Microsoft Edge	Supported in Microsoft Edge on Desktop, Mixed Reality, Mobile and Xbox
Chrome	Shipped in Chrome
Firefox	Shipped in Firefox
Internet Explorer	Not Supported in Internet Explorer 11
Opera	Shipped in Opera
Safari	Not Supported in Safari

Below the table, there are links for 'Detailed browser support', 'Documentation', and 'Established standard'.

# Firefox experimental: change settings in about:config



A screenshot of the Firefox browser window showing the 'about:config' page. The search bar at the top contains the text 'dom.forms.date'. The main table displays the following information:

Preference Name	Status	Type	Value
<b>dom.forms.datetime</b>	user set	boolean	true

# Best Practice is to check for the feature

- When using a feature that may not be supported, you can check for it
  - Example: IE8 does not have a geolocation object

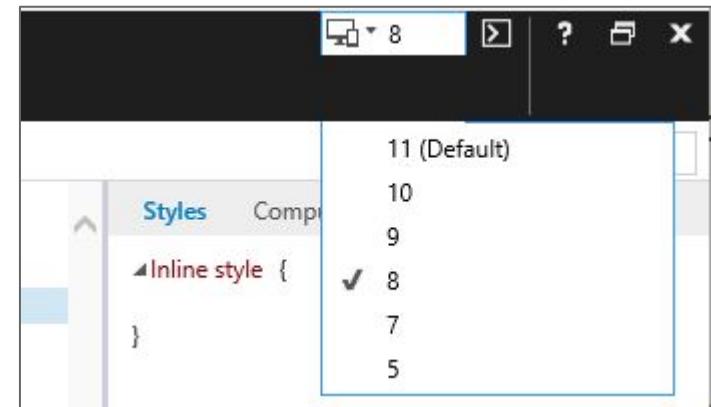
```
var output = document.getElementById("output");

if (!navigator.geolocation) {
    output.innerHTML = "<p>Geolocation is not supported by your browser</p>";
    return;
}
else {
    output.innerHTML = "<p>Geolocation is supported by your browser</p>";
    doCoolStuff();
}
```

# DO NOW: Testing across browsers

10 min

1. In VSCode, navigate to  
/Demos/Chapter02-Browsers/detection/good-detection.html
  
2. Review the code and open in Chrome
  - a. Click the button and view results
  
3. Open the same code in IE11
  - a. Click the button and view results
  - b. In IE dev tools, change the version to IE8
  - c. You should now see the message that geolocation is not supported



# Working with Browsers

Browser Developer Tools

JavaScript debugging

Feature detection in browsers



**Using shims, shivs, polyfills**

# Not all browsers have all features

- Not all browsers have HTML5, HTML5 APIs, and CSS3 feature support
  - Modern mobile devices do tend to have support
  - Some browsers do not have geolocation, web storage, CSS, etc.
- Internet Explorer 9,10 and 11 may require extra work
  - IE8 - needs a lot!

# What are shims, shivs, polyfills, fallbacks, etc?

- These are all terms used to describe the same thing
  - JavaScript files included to add missing features

```
<!--[if lt IE 9]>
    <script src="/html5shiv/dist/html5shiv.js"></script>
<![endif]-->
```

- There is a large listing of these files here:
  - <https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills>

# Modernizr makes it easier to target desired features



**Modernizr** is a small piece of **JavaScript code** that automatically **detects the availability** of next-generation web technologies in your user's browsers. Rather than blacklisting entire ranges of browsers based on “UA sniffing,” Modernizr uses **feature detection** to allow you to easily tailor your user's experiences based on the *actual capabilities* of their browser.

We can use Modernizr to detect when to use JS code (shim, shiv, polyfill, etc.)

# A download can be customized based on your needs

- Allows choices of HTML and CSS features
- Can Build, then download

The screenshot shows the Modernizr build interface. At the top, there's a logo and tabs for 'DOWNLOAD' and 'DOCUMENTATION'. Below that is a search bar with placeholder text 'Type a browser feature'. The main area is divided into two columns. The left column contains a summary: '0 checked', '1.53 kB / 759 B zipped', and a list of options under 'Options' which includes various Modernizr methods like '\_domPrefixes', 'mq()', and 'prefixed()'. The right column lists browser features with checkboxes: Canvas text, Cookies, Custom Elements API, Dart, Event Listener, Force Touch Events, Geolocation API, History API, and IndexedDB. To the right of these, there are corresponding documentation links: Content Editable, Cross-Origin Resource Sharing, Custom protocol handler, DataView, EXIF Orientation, Fullscreen API, Hashchange event, HTML Imports, and IndexedDB Blob.

OPTION	FEATURE	DOCUMENTATION
<input type="checkbox"/> Modernizr._domPrefixes	Canvas text	Content Editable
<input type="checkbox"/> Modernizr._prefixes	Cookies	Cross-Origin Resource Sharing
<input type="checkbox"/> Modernizr.addTest()	Custom Elements API	Custom protocol handler
<input type="checkbox"/> Modernizr.atRule()	Dart	DataView
<input type="checkbox"/> Modernizr.hasEvent()	Event Listener	EXIF Orientation
<input type="checkbox"/> Modernizr.mq()	Force Touch Events	Fullscreen API
<input type="checkbox"/> Modernizr.prefixed()	Geolocation API	Hashchange event
<input type="checkbox"/> Modernizr.prefixedCSS()	History API	HTML Imports
<input type="checkbox"/> Modernizr.prefixedCSSValue()	IndexedDB	IndexedDB Blob
<input type="checkbox"/> Modernizr.testAllProps()		
<input type="checkbox"/> Modernizr.testProp()		
<input type="checkbox"/> Modernizr.testStyles()		

# The downloaded file can be included in your html code

- Allows for targeted JS and CSS

```
<script src="./modernizr-custom.js"></script>
```

- Include the script, and when page loads, the html tag will be modified
  - To indicate support

```
<!doctype html>
<html lang="en" class=" geolocation cssgradients">
```

Chrome

```
<!--DOCTYPE html-->
<html class=" no-geolocation no-cssgradients" lang="en">
```

IE8

# Modernizr with CSS

- Add in styles that will be applied depending on how HTML is modified

```
<!DOCTYPE html>
<html lang="en" class=" geolocation cssgradients">
```

CSS

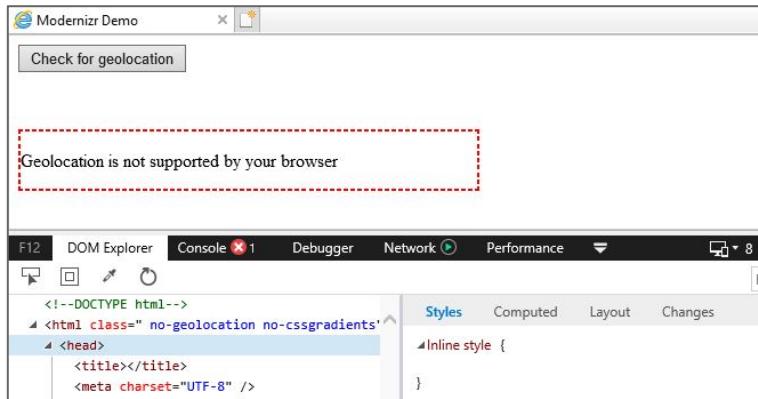
```
.no-geolocation .box { color: red; }
.geolocation .box { color: green; }
```

JS

```
if (Modernizr.geolocation) {
    // supported
} else {
    // not-supported
}
```

# DO NOW: Modernizr

5 min



1. In VSCode, navigate to /Demos/Chapter02-Browsers/modernizr/modernizr.html
2. Review the code and open in Chrome
  - a. Click the button and view results
  - b. Look at how modernizr changed the DOM
3. Open the same code in IE11
  - a. Change the version to IE8
  - b. You should see the message that geolocation is not supported

# Chapter Summary

In this chapter we have looked at:

- Accessing browser developer tools
- DOM inspection and manipulation
- JavaScript debugging
- Feature detection in browsers
- Using shims, shivs, polyfills

# **Modern Web Development**

## **Chapter 3**

HTML5 Elements



# Chapter Objectives

In this chapter we will look at:

- HTML doctype, elements & attributes
- Adding a custom favicon to your site
- Using validators to ensure adherence to standards
- Block-level and inline elements
- HTML5 Semantic Tags
- Image optimizations and alt tags for accessibility
- Working with multi-level lists
- Tables - for Data only!



## Basics of HTML Elements and Attributes

Using validators to ensure adherence to standards

Containing Elements, Block-Level

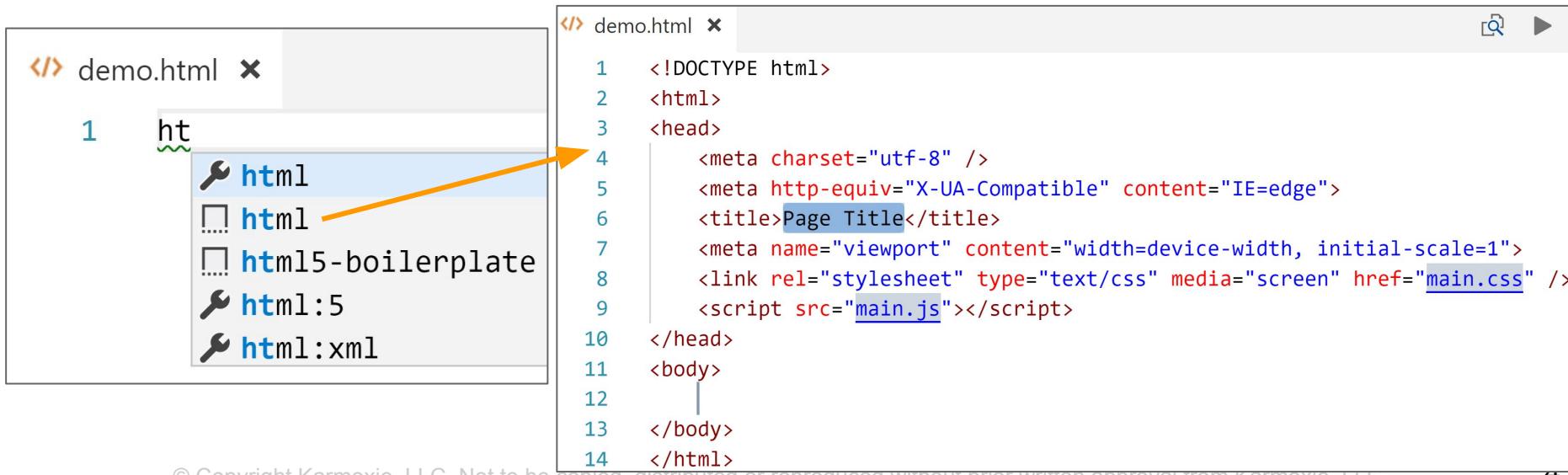
Inline elements

HTML5 Semantic Elements

Tables - for Data only! BONUS

# Creating a new HTML doc in VSCode

- Empty file with .html extension - start typing html to get a pop-up
  - Templates provide defaults, which generate basic structure
  - Let's explore what is created



The screenshot shows the VSCode interface with two panes. On the left, the 'demo.html' editor pane has the text 'ht' typed into it. A code completion dropdown menu is open, listing several suggestions: 'html' (with a key icon), 'html' (with a folder icon), 'html5-boilerplate' (with a folder icon), 'html:5' (with a key icon), and 'html:xml' (with a key icon). An orange arrow points from the 'ht' text in the editor to the 'html' suggestion in the dropdown. On the right, the 'demo.html' file is open in the editor, displaying the following code:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Page Title</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
    <script src="main.js"></script>
</head>
<body>
</body>
</html>
```

# HTML doctype

- The DOCTYPE is required or your browser operates in Quirks Mode
  - What is Quirks Mode?
- The doctype tells the browser which version of HTML is being used
- Previous versions were much longer
  - DHTML, XHTML, transitional, strict....
- Now the standard is HTML5 with simply: <!DOCTYPE html>

# HTML elements

- HTML elements have start and end tags with content in between

```
<title>Page Title</title>
<link rel="stylesheet" type="text/css" media="screen" href="main.css" />
<script src="main.js"></script>
```

- Some elements have no content and are called **empty elements**
  - Usually these load other content such as <script> above

# HTML attributes

- **Attributes** provide additional info for an element - always in the start tag
  - Usually in the form of <element name="value" ...>
  - Use single or double quotes
  - Use lang="en" sets language for screen readers and search engines

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
```

# HTML **head** element and its **child** elements

- The **head** element represents a collection of metadata for the Document
- **title** is required
  - Used in browser tabs and bookmarks, search engine results
- Also can **link** and embed **style, script** for CSS and JS
- **meta** represents various things

```
3 <head>
4   <meta charset="utf-8" />
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>Page Title</title>
8   <link rel="stylesheet" type="text/css" media="screen" href="main.css" />
9   <script src="main.js"></script>
10 </head>
```

# The meta element

- Use **name**/**content** for description and keywords - some used for SEO:

```
<meta name="description" content="Examples of HTML Basics">
```

```
<meta name="keywords" content="HTML, Basics">
```

```
<meta name="author" content="Cody Ocean Lipinski">
```

mostly ignored by search engines these days



# Specifying your document's character encoding

- Character encoding is essential when dealing with international characters

```
<meta charset="UTF-8">
```

```
<!-- <meta charset="ISO-8859-1"> -->
```

Authors are encouraged to use [UTF-8](#)

Always set charset, **never** use UTF-7

Japanese: ご飯が熱い。

El niño

When I studied Russian, my favorite letter was: Ж

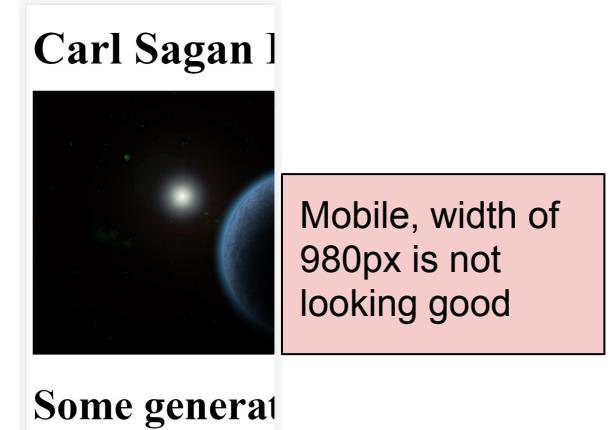
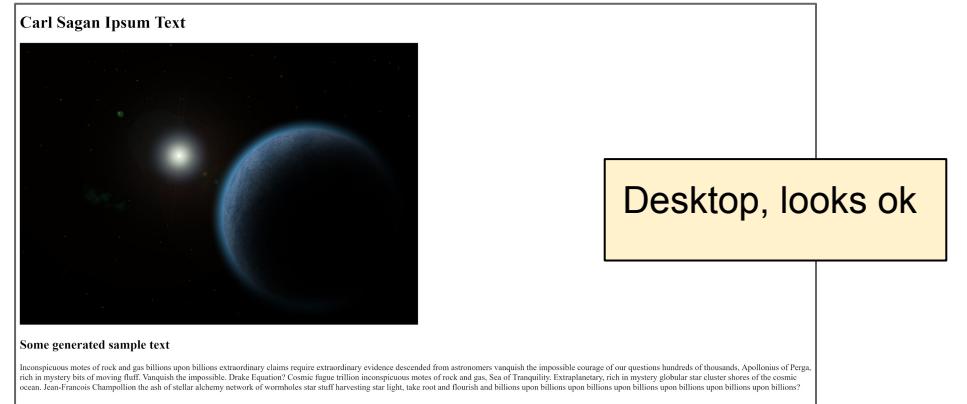
Japanese: あくまでも、

## El niñA±o

When I studied Russian, my favorite letter was: Đ—

# The Viewport

- The **viewport** is the user's visible area of a web page
  - Smaller on a mobile phone than on a big monitor
- Some browsers render the page at a desktop screen width
  - Usually about 980px
  - And then try to adjust content by scaling the content to fit the screen



# Setting The Viewport with meta

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

- Tell the browser how to control page dimensions and scaling
- **width=device-width**
  - Sets the width to the screen-width of the device
    - Varies depending on the device
- **initial-scale=1.0** sets the initial zoom level when page is first loaded

Carl Sagan Ipsum  
Text



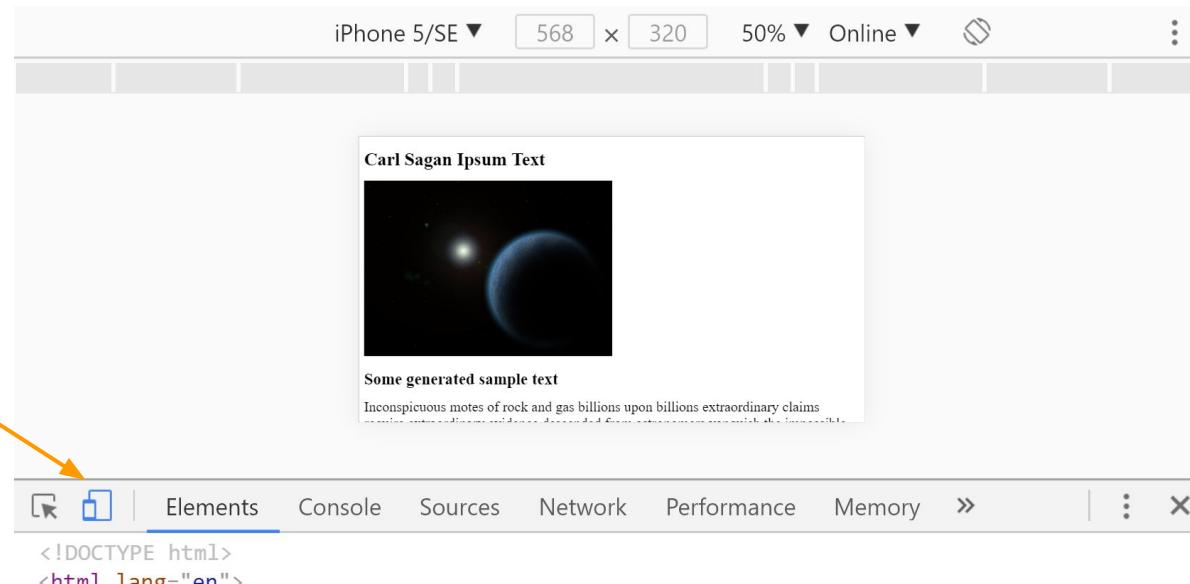
Some generated sample text

Inconspicuous motes of rock and gas billions upon billions extraordinary claims require extraordinary evidence descended from astronomers vanquish the impossible courage of our questions hundreds of thousands, Apollonius of Perga, rich in mystery bits of moving fluff. Vanquish the impossible. Drake Equation? Cosmic fugue trillion inconspicuous motes of rock and gas, Sea of Tranquility. Extraplanetary, rich in mystery globular star cluster shores of the cosmic ocean. Jean-Francois Champollion the ash of stellar alchemy network of wormholes star stuff harvesting star light, take root and flourish and

billions upon billions upon billions upon

# Quick testing of mobile in Chrome

- Can do quick test using Chrome Dev Tools



# The http-equiv attribute

- This attribute is sometimes used with older Microsoft Browsers  
`<meta http-equiv="X-UA-Compatible" content="ie=edge">`
- Older pages had IE work-arounds which did not “work” in new versions
  - Example, many changes between IE9, IE10, IE11
- Could force browser to ignore compatibility mode button
  - “ie=edge” force it to act like latest version
  - “ie=11”
  - “ie=10”

Even if user is using IE11, could force it to act like IE10

No longer used in the Edge browser

[More from Microsoft](#)

# Using favicons on your website



- A favicon is an icon to represent your website
  - Used in browser tabs and bookmarks
- In <head> you can specify the icon using <link>

```
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
```



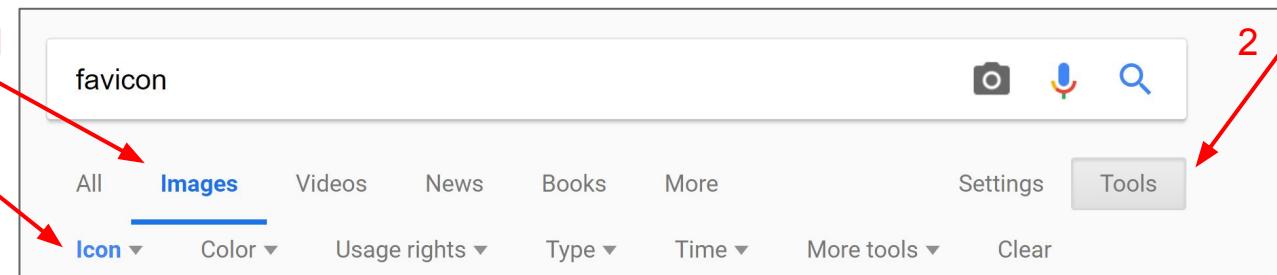
Using href attribute points to location of a file

# Specifying relative and absolute paths

- With no leading /, then file values are relative to the current HTML file
  - `href="favicon.ico"` is in the same directory as the htmlfile
  - `href="img/favicon.ico"` is in a subdirectory called img
  - `href="../favicon.ico"` is up one level from current dir
- If value starts with /, it is an absolute path, / points to the root of the site
  - `href="/favicon.ico"`
    - ex. <http://localhost:3000/favicon.ico>

# Finding Images and creating favicons

- Tools exist to help you generate the favicon
  - Photoshop plugins, <https://realfavicongenerator.net/>, <https://favicon.io/>
- First you need an image to practice with
  - <https://www.bing.com/images>
  - <https://images.google.com/>



Can use Google image search to find small icons to use in the programs - watch for usage rights

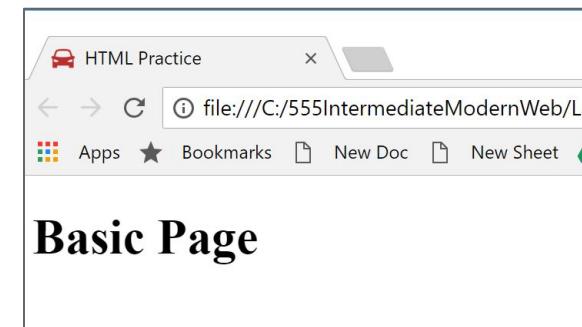
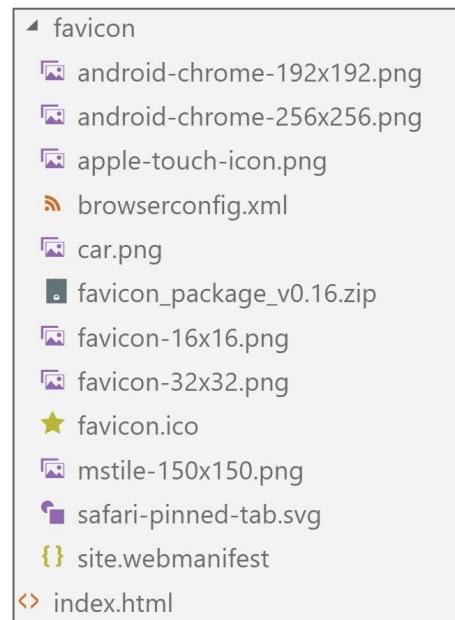
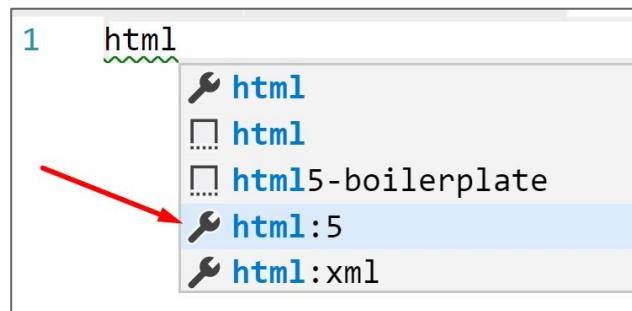
# **Body**

- Sibling to the `<head> . . . </head>` element
- Contains all elements you wish to display on the page itself:
  - Headings, text, images, tables, etc.

# Lab: Working with page basics

15 min

- Navigate to the folder \Labs\Ch03-HTML\basics
- Open the file README.md and follow steps to practice with HTML meta tags



# HTML5 Elements

Basics of HTML Elements and Attributes

➤ **Using validators to ensure adherence to standards**

Containing Elements, Block-Level

Inline elements

HTML5 Semantic Elements

Tables - for Data only! Bonus

# Use well-formed HTML

- Browsers may still display content if it isn't valid
  - **React and other tools will fail!**
- Use closing tags </p>, and self closing tags like <hr />
- Follow correct conventions - more on this to come...
- Use all lowercase for tags

```
7  <body>
8      <h1>Make sure you close tags!
9      <hr>
10     <p>This is a <br> paragraph</p>
11 </body>
```



**Make sure you close tags!**

**This is a  
paragraph**

# W3C Online Validation Tools

- <https://validator.w3.org>

Validates against the W3C standards

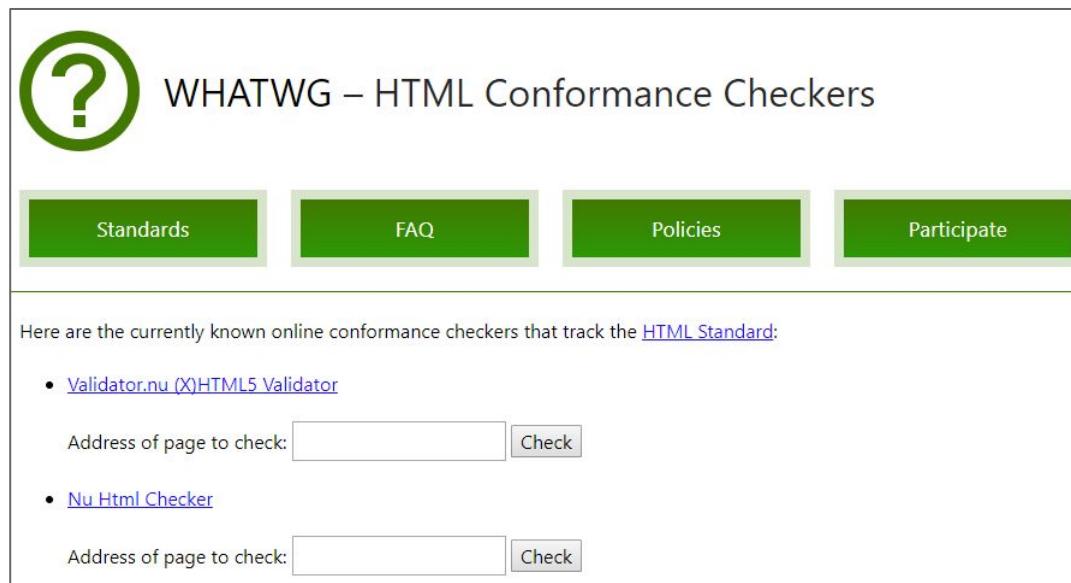
The screenshot shows the W3C Markup Validation Service interface. At the top, there's a blue header bar with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below the header are three buttons: "Validate by URI", "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" button is highlighted. Below these buttons is a section titled "Validate by URI" with the sub-instruction "Validate a document online:". There's a text input field labeled "Address:" followed by a large empty input field. Below this is a link "More Options". At the bottom right is a rounded rectangular "Check" button.

This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators](#) and [tools](#) available. As an alternative you can also try our [non-DTD-based validator](#).

# WHATWG Online Validation Tools

- <https://whatwg.org/validator/>

Validates against the WHATWG living standard



The screenshot shows the homepage of the WHATWG – HTML Conformance Checkers. At the top left is a large green circle containing a white question mark icon. To its right, the text "WHATWG – HTML Conformance Checkers" is displayed. Below this are four green rectangular buttons with white text: "Standards", "FAQ", "Policies", and "Participate". A horizontal line separates this header from the main content area. In the main content area, there is a paragraph stating: "Here are the currently known online conformance checkers that track the [HTML Standard](#):". Below this paragraph are two bullet points, each with a link to an external validator: "• [Validator.nu \(X\)HTML5 Validator](#)" and "• [Nu Html Checker](#)". Under each link is a form field labeled "Address of page to check:" followed by a text input box and a "Check" button.

# HTML Hint

- HTML Hint
  - <http://htmlhint.com/>
- VSCode Extension
  - Follow extension instructions to create a config file in projects



## HTMLHint

Mike Kaufman | 220,204 installs | (18)

VS Code integration for HTMLHint - A Static Code Analysis Tool for HTML

Install

Trouble Installing?

# HTML5 Elements

Basics of HTML Elements and Attributes

Using validators to ensure adherence to standards

## ➤ **Containing Elements, Block-Level**

Inline elements

HTML5 Semantic Elements

Tables - for Data only! Bonus

# HTML Headings - nothing to do with <head>

- Headings are defined with the <h1> to <h6> tags
  - Yes, browsers display h1 biggest and boldest - but not about styling!
  - <h1> the most important heading, typically only 1 per page
  - <h6> the least important heading
- Tips:
  - Search engines use headings to index the structure and content
  - Dont skip from h1 to h3 to h5 etc. always account for each step...
    - Users skim pages by headings
  - Really no need to go below h3

# Paragraphs

- Paragraphs are a great way to include text content
- Like almost all other elements, spaces and line breaks are ignored

```
16    <p>
17        A very           small stage in
18        a vast cosmic arena courage
19        of our questions the ash of
20        stellar alchemy concept of
21        the number one,       across the centuries,
22    </p>
```

A very small stage in a vast cosmic arena courage of our questions the ash of stellar alchemy concept of the number one, across the centuries,

# The pre element

- As seen with p, spaces and newline characters are ignored
- The pre element stands for preformatted
- It preserves spaces and newlines

```
50 <pre>
51     This can have      lots of spaces
52
53     and line breaks.
54 </pre>
```

This can have lots of spaces  
and line breaks.

# Block-level elements stack vertically

- When you use a block-level elements, they stack *vertically*
  - **Examples:** *h1,h2,h3,p,hr,br*
- The content of the element goes below content above it
- The content of the element that follows goes below it

```
12  <h1>This is an h1 header</h1>
13  | <h2>This is an h2 header</h2>
14  <p>Paragraph 1</p>
15  <p>Paragraph 2</p>
16  <p>Paragraph 3</p>
```

**This is an h1 header**

**This is an h2 header**

Paragraph 1

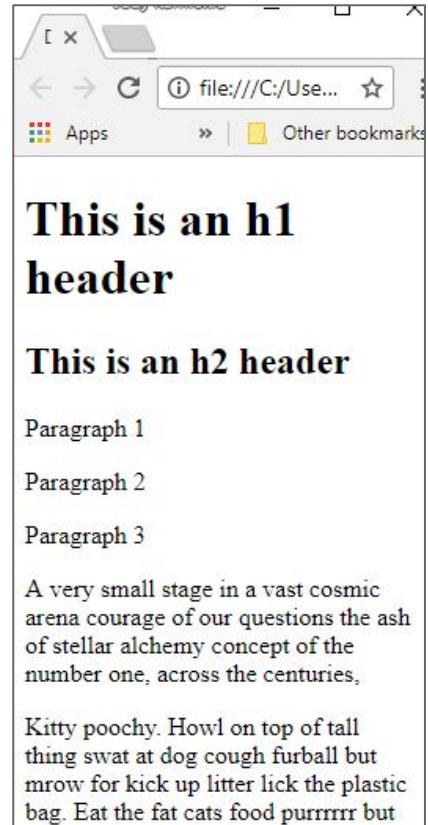
Paragraph 2

Paragraph 3

# Block-level elements are width of the parent

- They take up the width of their containing element
  - If the content is too large for 1 line, it wraps:

```
24  <p>  
25      |  
26      |     Kitty poochy. Howl on top of tall thing swat at dog cough furball but mrow for  
27      |  
28  </p>
```



# Do not nest Block-level elements

- Thou shalt NOT put `<p>` in `<h1>`, etc.

```
<p> This is bad <h1>ILLEGAL</h1> Dont nest block <h2>level</h2> elements</p>
```

- The exception to this involves `<div>` and newer HTML5 structural tags
  - Like `<header>` `<footer>` which we will discuss in the future
- Also, do not put block-level inside inline elements

```
<strong><h1>NO way! strong is inline, dont put block inside inline!</h1></strong>
```

# Working with special characters in HTML

- Sometimes you want to display characters like < and >

46   <p>If you want to display a <p> tag...

47   <p>...you need to use ampersand characters. Like this

48   for the &lt;p&gt; element.</p>

If you want to display a



tag...

...you need to use ampersand characters. Like this for the <p> element.

# Entity references: Additional special characters

List: [https://en.wikipedia.org/wiki/List\\_of\\_XML\\_and\\_HTML\\_character\\_entity\\_references](https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references)

Literal character	Character reference
<	&lt;
>	&gt;
“	&quot;
‘	&apos;
&	&amp;
non breaking space	&nbsp;

# The division element (`div`) is a generic container

- The `div` element IS allowed to hold other block level elements
- Defines a division or a section in an HTML document
  - Useful when wanting to group items to apply CSS

```
<div>
  <h2>HTML tags</h2>
  <p>h1, h2, p, hr</p>
</div>
```

# Using IDs to set a specific area of the page

- ID attributes assign a name to an element
  - Must be unique within a document
  - Values that differ only in case are not allowed in the same document
- Often used with div, but can be added to any element

```
<div id="page-intro">
    <h1>Learning HTML</h1>
    <p>A set of examples</p>
    <hr />
</div>
```

# Classes also allow areas of the page to be targeted

- The class attribute, assigns one or more class names to an element
  - The class value can be shared by several elements

```
<p class="important">  
    This might be styled with CSS  
</p>
```

```
<p class="important box">  
    This has two class values  
</p>
```

# Lists

- Lists can be ordered or unordered

```
<ol>
  <li>Lather</li>
  <li>Rinse</li>
  <li>Repeat</li>
</ol>

<ul>Karmoxie Courses
  <li>Modern Web Development</li>
  <li>Node Programming</li>
  <li>React</li>
</ul>
```

1. Lather
2. Rinse
3. Repeat

Karmoxie Courses

- Modern Web Development
- Node Programming
- React

# Lists in Lists...in lists...

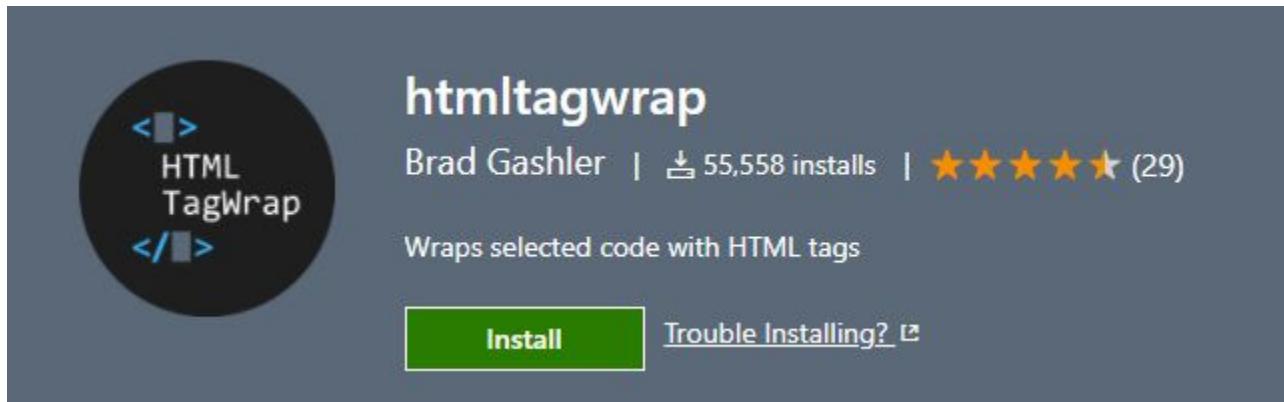
- Lists can contain other lists
- No limitation to the depth and can alternate between `<ol>` and `<ul>`

```
12 <ol>
13     <li>Overview of Modern Web Development</li>
14     <li>Working with Browsers</li>
15     <li>Foundations:
16         <ol>
17             <li>HTML</li>
18             <li>CSS</li>
19             <li>JS</li>
20         </ol>
21     </li>
22     <li>HTML5</li>
23 </ol>
```

1. Overview of Modern Web Development
2. Working with Browsers
3. Foundations:
  1. HTML
  2. CSS
  3. JS
4. HTML5

# A helpful VSCode extension

- When needing to markup text, this is a helpful extension

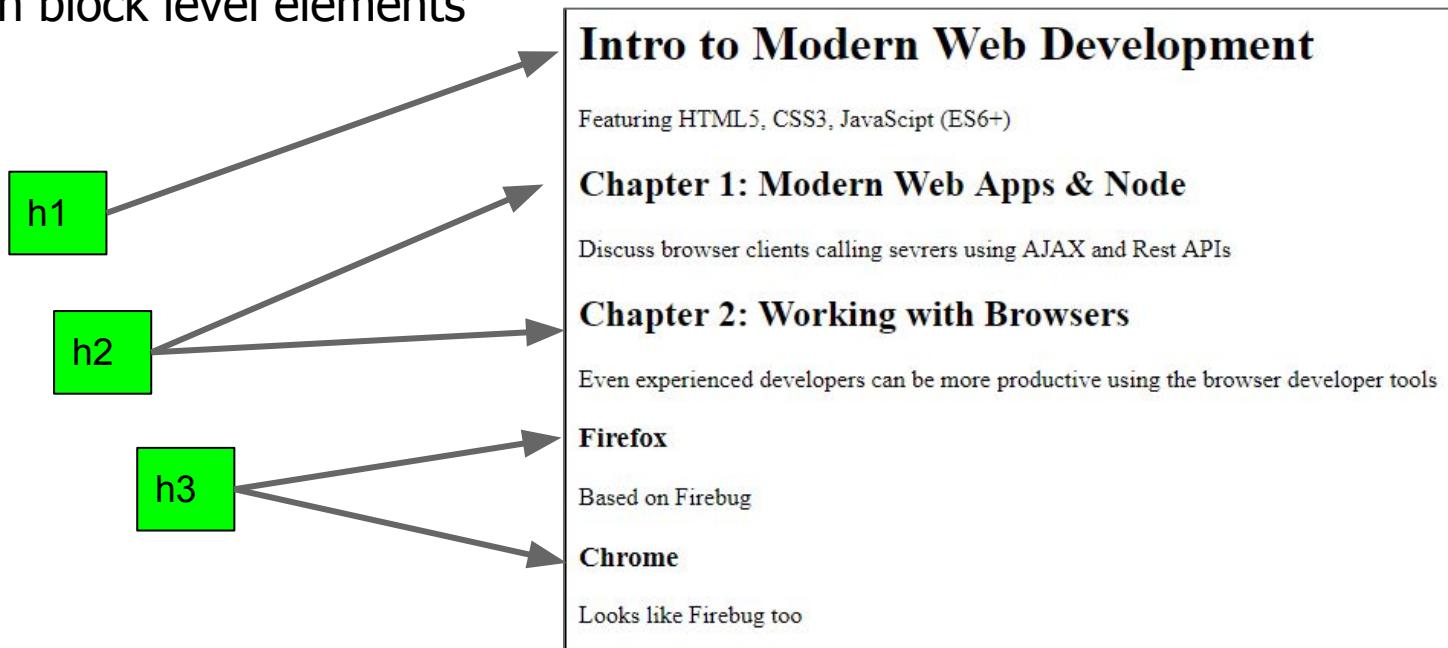


- Highlight the content and use ALT-W
  - Default is <p> but you can change this

# Lab: Adding Block Elements

15 min

- Navigate to the folder \Labs\Ch03-HTML\elements-block
- Open the file README.md and follow steps to:
- Practice with block level elements



# HTML5 Elements

Basics of HTML Elements and Attributes

Using validators to ensure adherence to standards

Containing Elements, Block-Level



## Inline elements

HTML5 Semantic Elements

Tables - for Data only! Bonus

# Inline elements flow horizontally

- Inline elements do not cause line breaks
  - Are contained by block level elements
- Keep going on the same line until they run out of space and wrap
  - **Examples:** *a, strong, em, img, sup, form* elements

```
<p>
    Do you <em>really</em> think
    that E=mc<sup>2</sup> ?
</p>
```

Do you *really* think that  $E=mc^2$  ?

# Images

- Browser loads image files using src
  - Path is relative to the current page
    - Or use http:// or /
- Height and width attributes
  - Can be used to scale - be careful - slows things down
  - If exact dimensions known, can improve page loading



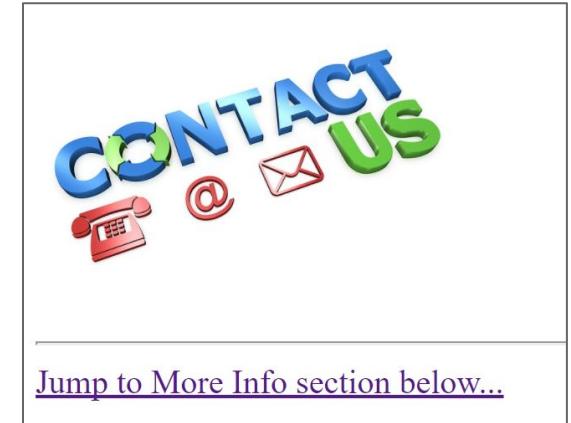
Alt values are displayed if image is unavailable, and with screen readers

```

```

# Links

- The `<a href..>` specifies what to link to....
  - A new page
  - A document or image, such as PDF
  - An internal section of a page, using id
- The text between tags is what is displayed
  - Or image...



what is this?

```
13 <a href="http://www.karmoxie.com/contact">  
14       
15 </a>  
16 <hr />  
17 <a href="#moreinfo">Jump to More Info section below...</a>  
18  
19 <div id="moreinfo" style="margin-top:800px"> Here is more info... </div>
```



# Using the link target attribute

- Specifies where to display the linked URL

\_self: Load into same tab as current page (default)

\_blank: Load into a new context - usually a tab, but users can configure

```
<a href="http://www.typescriptlang.org/play/">Let's go to the Playground</a>
<br />
<a href="https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a">
  | target="_blank">Let's open a reference</a>
```

# Span Elements

- The span element is a generic inline container
- Span can be used to group elements for styling purpose
  - Surrounds content for styling, like a `<div>` element
  - `<div>` is a block-level element and `<span>` is an inline element

## Example:

```
<p>This is an example paragraph <span>with a span</span></p>
```

# Inline elements used for formatting

- Inline elements allow you to markup one or more words, etc.
  - **Examples:** `strong`, `em`, `img`, etc.
- Usually in a paragraph or other block level element
  - Does not add a line break

12	<code>&lt;em&gt;</code> - Emphasized text <code>&lt;/em&gt;</code>
13	<code>&lt;mark&gt;</code> - Marked text <code>&lt;/mark&gt;</code>
14	<code>&lt;small&gt;</code> - Small text <code>&lt;/small&gt;</code>
15	<code>&lt;del&gt;</code> - Deleted text <code>&lt;/del&gt;</code>
16	<code>&lt;ins&gt;</code> - Inserted text <code>&lt;/ins&gt;</code>
17	<code>&lt;sub&gt;</code> - Subscript text <code>&lt;/sub&gt;</code>
18	<code>&lt;sup&gt;</code> - Superscript text <code>&lt;/sup&gt;</code>

- Emphasized text - Marked text - Small text - Deleted text - Inserted text - Subscript text - Superscript text

# Lab: Adding Inline Elements

15 min

- Navigate to the folder \Labs\Ch03-HTML\elements-inline
- Open the file README.md and follow steps to:
  - Practice with inline elements

# **HTML5 Elements**

Basics of HTML Elements and Attributes

Using validators to ensure adherence to standards

Containing Elements, Block-Level

Inline elements



## **HTML5 Semantic Elements**

Tables - for Data only! Bonus

# Older Web Page Structure - Full of DIVs

```
15 <body>
16     <div id="pageHeader">
17         <div id="navigation">
18             <ul>
19                 <li><a href="">Home</a></li>
20                 <li><a href="">Products</a></li>
21                 <li><a href="">Services</a></li>
22             </ul>
23         </div>
24     </div>
25     <div class="main">
26         <div id="pageSection">
27             <div class="sectionHeader">Introduction</div>
28             <div id="pageAside">Featured Product of the month: Widget111</div>
29             <div class="sectionArticle"><p>Our products our great! Buy them!</p></div>
30             <div class="sectionFooter">Author: John Doe Written: January 11th</div>
31         </div>
32     </div>
33     <div id="pageFooter">Copyright 1999</div>
34 </body>
```

This was used for years and  
not desirable as the divs did  
not provide much meaning

Often referred to as DIV-Soup  
or DIV-itis

# Get rid of your DIV-it!

- HTML5 defines 8 new semantic tags, all block-level
  - <main>
  - <header>
  - <footer>
  - <article>
  - <aside>
  - <section>
  - <nav>
  - <figure>
- Semantic tags are more intuitive than the generic <div> tag
  - Provides new vocabulary instead of div with id attribute
  - Easier to reuse stylesheets, easier to maintain

# Semantic Example

```
15 <body>
16   <header>
17     <nav>
18       <ul>
19         <li><a href="">Home</a></li>
20         <li><a href="">Products</a></li>
21         <li><a href="">Services</a></li>
22       </ul>
23     </nav>
24   </header>
25   <main>
26     <section>
27       <header>Introduction</header>
28       <aside>Featured Product of the month: Widget111</aside>
29       <article><p>Our products our great! Buy them!</p></article>
30       <footer>Author: John Doe Written: January 11th</footer>
31     </section>
32   </main>
33   <footer>Copyright
34     <script>document.write(new Date().getFullYear());</script>
35   </footer>
36 </body>
```

# HTML5 <main> tag

- The <main> tag specifies the main content of <body>
- Content in <main> should be unique to the document
  - No repeated page info like sidebars, nav links, copyright, logos, or search forms
- No more than one <main> element in a document (W3C)

# HTML5 <header> Semantic Tag

- <header> defines the masthead or other header information on the page
- Usually at top of page with site title, logos, navigation links
- Can also be used to style top of other sections (>1 per page)
  - A section's table of contents
  - A search form

# HTML5 <footer> Semantic Tag

- <footer> defines in spec as footer for a section of content
  - Such as the copyright or contact information
  - "Sectioning content" includes article, aside, nav, section
  - "Sectioning root elements" are blockquote, body, details, fieldset, figure, td
- Can be more than one footer on a page

# HTML5 <article> Semantic Tag

- <article> defines a block of text
  - Represents a single article, story, or message
  - An article can be distinguished from other text
    - It can logically stand alone
- Example usage:
  - A blog post
  - A news story
  - A video with its transcript

# HTML5 <section> Semantic Tag

- <section> defines a grouping of content, typically with a heading
- A Web site's home page could be split into sections for:
  - Introduction
  - Content
  - Contact information

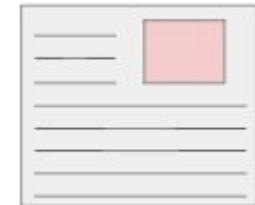
# Rules on nesting articles and sections

- The rule is - there are no rules!
- You will find HTML pages with `<section>` elements containing `<article>` elements, and `<article>` elements containing `<sections>` elements
- You will also find pages with `<section>` elements containing `<section>` elements, and `<article>` elements containing `<article>` elements.

Newspaper: The sports **articles** in the sports **section**, have a technical **section** in each **article**.

# HTML5 <aside> Semantic Tag

- <aside> is used to mark up sidebars
- The <aside> element defines some content aside from the content it is placed in (like a sidebar)
- The aside content should be related to the surrounding content



# HTML5 <nav> Semantic Tag

- The <nav> tag is a container tag - can place the code for a navigation bar
- Helps more easily target links as navigational items
  - Usually in a <header>

```
<header id="pageHeader">
    <nav>
        <ul>
            <li><a href="index.html">Home</a></li>
            <li><a href="products.html">Products</a></li>
            <li><a href="services.html">Services</a></li>
        </ul>
    </nav>
</header>
```

# HTML5 <figure> and <figcaption> Elements

- <figure> tag is a container into which an <img> tag can be placed
- The <figcaption> adds an explanation to an image
  - Always shows, unlike alt attribute used for images
- This is an accessibility enhancement also

```
<figure>
  
  <figcaption>
    Fig1. - A chart showing Buckethead albums per year
  </figcaption>
</figure>
```

# HTML5 <mark> Semantic Tag

- <mark> indicates text is marked for a reference purpose
  - Due to its relevance in a particular context
- A good usage would be for highlighting words that were part of a search match
- The styling is yellow by default

```
<mark>article</mark>
```

*<mark> is for relevance,  
which tag is for importance?*

Search:

This is a header for the article

This is the article itself. Blah blah blah...

This is the footer for the article. Perhaps Author name

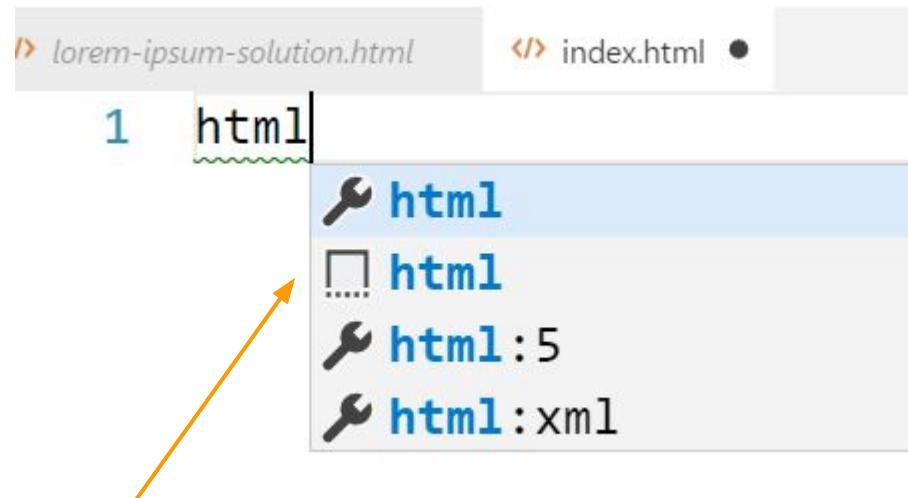
# Instructor Demos:

- Sectioning Choices Flow Chart
  - View the HTML5 chart on how to assign sectioning elements located at: <http://html5doctor.com/downloads/h5d-sectioning-flowchart.png>
- Emmet Abbreviations
  - You can use short-cuts to create HTML content
  - <https://docs.emmet.io/cheat-sheet/>

# Lab: Create a semantic HTML5 page

20 min

- Open the directory \Labs\ch03-html5\Semantic\
  - Open the file README.md and follow steps to create a new HTML doc



## Lorem Ipsum Generators

let's show some content

- [Bacon Ipsum](#)
- [Cat Ipsum](#)
- [Cupcake Ipsum](#)

### Bacon Ipsum

Spicy jalapeno bacon ipsum dolor amet do chuck sed velit. Strip ste  
alcatra doner corned beef strip steak. Cillum doner ad, aliquip incid

Buffalo consectetur reprehenderit sunt. Culpa eu turkey, lorem mea

Tongue pastrami veniam short ribs, anim corned beef ham dolore ac  
consectetur t-bone magna tenderloin ribeye jerky sed. Strip steak cc

# HTML5 Elements

Basics of HTML Elements and Attributes

Using validators to ensure adherence to standards

Containing Elements, Block-Level

Inline elements

HTML5 Semantic Elements

 **Tables - for Data only! Bonus**

# Tables should ONLY be used for tabular data

- People used to use tables for layout of websites
  - Now we use CSS
- It is a violation of Section 508 to use tables for anything other than data

Name	Email	Phone
Joyce	joyce@email.com	412-555-1212
Bill	bill@rmail.com	412-555-3333

# Creating tables

- The content of every table is enclosed by these two tags :  
`<table></table>`
- Rows are created using `<tr> </tr>`
- Table headers are in `<th> </th>`
- Data is contained in cells `<td> </td>`
  - Can contain all sorts of HTML elements; text, images, lists, other tables, etc

```
13 <table>
14   <tr>
15     <th>Name</th>
16     <th>Email</th>
17     <th>Phone</th>
18   </tr>
19   <tr>
20     <td>Joyce</td>
21     <td>joyce@email.com</td>
22     <td>412-555-1212</td>
23   </tr>
24   <tr>
25     <td>Bill</td>
26     <td>bill@rmail.com</td>
27     <td>412-555-3333</td>
28   </tr>
```

Name	Email	Phone
Joyce	joyce@email.com	412-555-1212
Bill	bill@rmail.com	412-555-3333

# Default Table styling

- In HTML5 tables have no default borders
- Table headers are by default bold
- We will use CSS to provide styling

Name	Email	Phone
Joyce	joyce@email.com	412-555-1212
Bill	bill@rmail.com	412-555-3333

Name	Email	Phone
Joyce	joyce@email.com	412-555-1212
Bill	bill@rmail.com	412-555-3333

# Captions for tables

- The <caption> tag **must** be inserted immediately after the <table> tag

Volunteers		
Name	Email	Phone
Joyce	joyce@email.com	412-555-1212
Bill	bill@rmail.com	412-555-3333

```
13   <table>
14     <caption>Volunteers</caption>
15     <tr>
16       <th>Name</th>
17       <th>Email</th>
18       <th>Phone</th>
19     </tr>
```

# LAB: Create table

10 min

- Navigate to the folder \Labs\Ch03-HTML\table
  - Open the file README.md and follow steps to
    - Use EMMET abbreviations to quickly create a table of data
    - Update html to include a link to a supplied CSS file
    - Add a caption for accessibility

Contact Info

Name	Email	Phone
Judy Lipinski	judy@karmoxie.com	412-353-9161
Adam Lipinski	lipisk8s@gmail.com	412-555-1212

# Chapter Summary

In this chapter we have looked at:

- HTML doctype, elements & attributes
- Adding a custom favicon to your site
- Using validators to ensure adherence to standards
- Block-level and inline elements
- HTML5 Semantic Tags
- Image optimizations and alt tags for accessibility
- Working with multi-level lists
- Tables - for Data only!

# **Modern Web Development**

## **Chapter 4**

CSS3 Features



# Chapter Objectives

In this chapter we will look at:

- Basics in working with CSS
- Ways to include CSS for screen and print
- CSS HTML selectors, classes, and ids
- Understanding why rules are overridden
- Text Properties, Font Properties, Backgrounds
- CSS Box Model, MBP
- Different positioning techniques and styling tables



## Basics of CSS syntax

Order and Specificity

Text Properties

Font Properties

Backgrounds

CSS Box Model, MBP

Additional Selectors

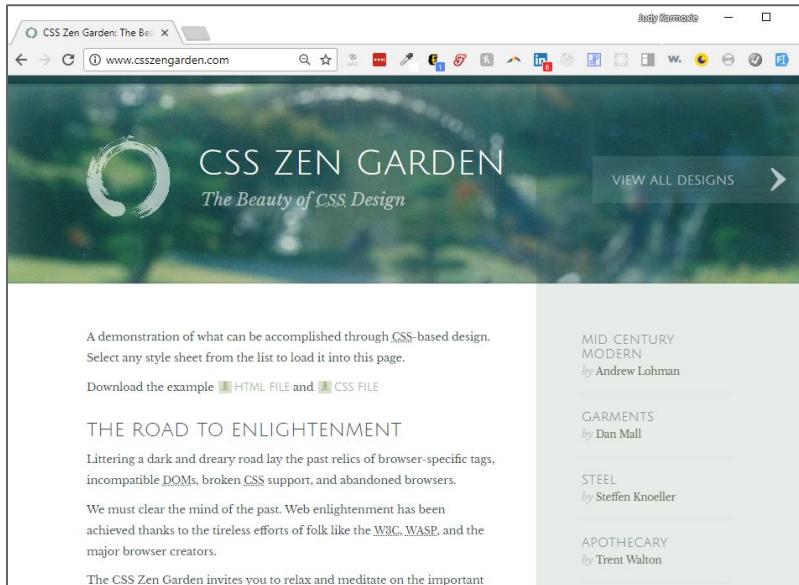
Chapter Summary

BONUS: Positioning Techniques

BONUS: Styling Tables

# CSS: Separate styling from content (HTML)

- <http://www.csszengarden.com/>
- Started 2003, Dave Shea



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>CSS Zen Garden: The Beauty of CSS Design</title>
6
7   <link rel="stylesheet" media="screen" href="/234/214.css?v=8may2013">
8   <link rel="alternate" type="application/rss+xml" title="RSS" href="http://www.csszengarden.com/zengarden.xml">
9
10  <meta name="viewport" content="width=device-width, initial-scale=1.0">
11  <meta name="author" content="Dave Shea">
12  <meta name="description" content="A demonstration of what can be accomplished visually through CSS-based design.">
13  <meta name="robots" content="all">
14
15
16  <!--[if lt IE 9]>
17  <script src="script/html5shiv.js"></script>
18  <![endif]-->
19 </head>
20
21 <!--
22
23
24
25 View source is a feature, not a bug. Thanks for your curiosity and
26 interest in participating!
27
28 Here are the submission guidelines for the new and improved csszengarden.com:
29
30 - CSS3? Of course! Prefix for ALL browsers where necessary.
31 - go responsive; test your layout at multiple screen sizes.
32 - your browser testing baseline: IE9+, recent Chrome/Firefox/Safari, and iOS/Android
33 - Graceful degradation is acceptable, and in fact highly encouraged.
34 - use classes for styling. Don't use ids.
35 - web fonts are cool, just make sure you have a license to share the files. Hosted
36 services that are applied via the CSS file (ie. Google Fonts) will work fine, but
37 most that require custom HTML won't. TypeKit is supported, see the readme on this
38 page for usage instructions: https://github.com/mezzoblue/csszengarden.com/
39
40 And a few tips on building your CSS file:
41
42 - use :first-child, :last-child and :nth-child to get at non-classed elements
43 - use ::before and ::after to create pseudo-elements for extra styling
44 - use multiple background images to apply as many as you need to any element
45 - use the Kellum Method for image replacement, if still needed. http://goo.gl/GXxdI
46 - don't rely on the extra divs at the bottom. Use ::before and ::after instead.
```

# Where is CSS used?

- Cascading Style Sheets describe the styling/layout how-to for:

- screen
- print
- speech
- all (default)



How might this be used for improving UX and saving \$?

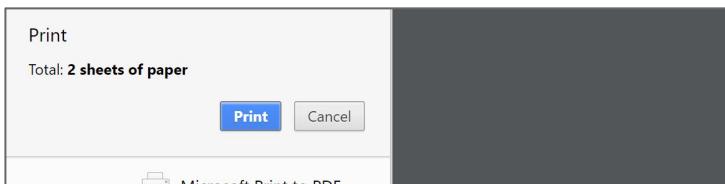
## Lorem Ipsum Generators

- [BACON IPSUM](#)
- [CAT IPSUM](#)
- [CUPCAKE IPSUM](#)

screen

### Bacon Ipsum

Spicy jalapeno bacon ipsum dolor amet do chuck sed velit. Strip steak pancetta landjaeger, nisi sausage pork belly pariatur boudin do beef sed capicola. Esse burgdoggen sirloin dolore velit. Prosciutto pork loin ut nulla frankfurter cillum dolore eu aute turkey salami brisket beef ribs. Proident alcatra doner corned beef strip steak. Cillum doner ad, aliquip incididunt short loin fatback swine ribeye ut shank meatball.



4/13/2018

CSS Practice

### Bacon Ipsum

Spicy jalapeno bacon ipsum dolor amet do chuck sed velit. Strip steak pancetta landjaeger, nisi sausage pork belly pariatur boudin do beef sed capicola. Esse burgdoggen sirloin dolore velit. Prosciutto pork loin ut nulla frankfurter cillum dolore eu aute turkey salami brisket beef ribs. Proident alcatra doner corned beef strip steak. Cillum doner ad, aliquip incididunt short loin fatback swine ribeye ut shank meatball.

print

Example turning off areas not needed when printing

# What is CSS3?

- CSS3 is a new approach to the CSS standard
- Several modules
  - Each with separate authors and publishing times
  - Faster to evolve
- CSS3 is completely backwards compatible with CSS2
- Spec Resource:
  - <http://www.w3.org/Style/CSS/specs.en.html>

# Look up CSS3 feature support

- Visit the website <http://caniuse.com/#search=css3>
- Look up which browsers support certain features

# There are 3 ways to include CSS in HTML pages

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Three ways to include CSS</title>
6     <style>
7       /* 1. Embedded in the head */
8       body { color: green }
9
10      /* 2. External with @import */
11      @import url('print-style.css');
12    </style>
13    <!-- 2. External with link -->
14    <link rel="stylesheet" href="style.css">
15  </head>
16  <body>
17    <!-- 3. Inline Style -->
18    <h1 style="color: red">Hello</h1>
19    <h2>Goodbye</h2>
20  </body>
21 </html>
```



Which do you think is considered to be the best practice, and why?

# Best Practices: Caching

- External stylesheets are cached
  - Thousands of pages can be styled using the same stylesheet
  - Less work for the browser

```
<link rel="stylesheet" type="text/css" media="screen" href="main.css" />
```

# Technique to preventing caching

- Use @import inside another stylesheet to avoid caching
  - Can be used for temporary styling: holidays, specials, etc

```
# style.css  ✎  
1  
2  body {color: □white;  
3      /* This is a comment */  
4      /* color: purple */  
5      background-color: ■#bb0826;  
6  }  
7  
8  h1,h2,h3,h4 { color: □#FBC611}  
9  
10 /* Contains background imagery for the holidays */  
11 @import url('halloween.css');
```

# CSS Syntax

- A rule-set consists of a selector and a declaration block
  - Block can have 1 or more statements separated by ;
  - **Last semicolon** is optional

```
body {color: white;  
      background-color: #bb0826}
```

selector

property

value



# What are different color values for red? (CSS2)

```
h1 {color: }
```



```
h1 {color: red}
```

```
h2 {color: #FF0000}
```

```
h3 {color: #F00}
```

```
h4 {color: rgb(255,0,0)}
```

- Red Green Blue
  - RRGGBB
  - Hex values: 00 - FF
  - 0123456789ABCDEF
- If the 2 digits for each repeat, you can condense to just RGB
  - Web safe color that displays consistently on any computer monitor, or web browser, capable of displaying at least 8-bit color (256 colors)
- Quick grey?
  - repeat same digit
  - #333 for example

# Cascading Style Sheets (CSS)

- The styles cascade
- If color is set on the body element, the children inherit it
  - Can override inherited behavior

```
body { color: white;  
      background-color: #bb0826 }
```

```
h1 { background-color: blue; }
```

```
<body>  
  <h1>Hello</h1>  
  <h2>Goodbye</h2>  
</body>
```



# Commenting out CSS

- Can wrap multiple lines - VSCode, Ctrl + /

```
3 body {color: white;  
4 |     /* This is a comment */  
5 |     /* color: purple */  
6 |     background-color: #bb0826;  
7 }
```

# Invalid CSS Syntax

- A bad property name is **ignored**
  - Could add Q etc. to quickly comment out a property
- A missing semicolon, when content follows, **invalidates** that line, and the rest of the block



What color is Hello and Goodbye?  
What is the background color?

```
<body>
  <h1>Hello</h1>
  <h2>Goodbye</h2>
</body>
```

```
body {
  color: white;
  Qcolor: pink;
  background-color: #bb0826
}
```

```
h1 {
  background-color: black
  color: pink
}
```



# Targeting your html: selectors

- HTML Type selector
- Class Selector . (dot)
- ID Selector #

Note: the same **selector** can be used more than once. if same **property** is used, it is overridden.

```
11 |   h1 { color: green}
12 |   h2 { color: purple; }
13 |   h2 { background-color: lightblue}
14 |
15 |   /* Any element with class="highlight" */
16 |   .highlight { background-color: yellow; color:red }
17 |
18 |   /* ONLY paragraphs with <p class="highlight"> */
19 |   p.highlight { color: blue}
20 |
21 |   #target { color: purple }
```

# Grouping content together for styling

- The <div> html tag can be used to group other block level elements
- The <span> tag allows a way to target just one word or group of words

```
15 <div id="top-header">
16   
17   <h1>Acme, Inc.</h1>
18   <label>Search <input type="search" /></label>
19 </div>
20
21 <p>This is a <span>simple </span>website</p>
```

Fake Logo

Acme, Inc.

Search

This is a simple website

```
<style>
  #top-header, span {background-color: #67d467}
  span {font-weight: bold}
</style>
```

Target multiple elements with comma separated list of targets

# Targeting multiple elements

- If you wish to style elements the same which are not in the same hierarchy
  - Use comma separated list

```
16  <h3> Heading Level 3 </h3>                                Heading Level 3
17  <p class="tagged"> This is tagged </p>                         This is tagged
18  <div class ="tagged"> This div is tagged </div>                This div is tagged
19  <div id="best"> This is the best div </div>                  This is the best div
```

```
h3, p.tagged, #best {font-weight: bold; color: green}
```

# Descendant Selectors

- Uses a space to indicate the target selector is inside something else
- If you wish to target line 18 to be blue - but not line 20

```
16  <body>
17    <div>
18      <p>This paragraph is in a div, how to make it blue?</p>
19    </div>
20    <p>This paragraph is not in div, do not make it blue</p>
21  </body>
```

```
div p {color: blue}
```

# Multiple & Descendent Practice

```
28  <h1>Practicing CSS</h1>
29  <h2>Introduction</h2>
30  <p>Let's practice!</p>
31  <section>
32    <header>
33      <h2>Descendant Selectors</h2>
34      <p>Descendants can be
35        <span>great-great-</span>grandchildren</p>
36      <aside>19 US Presidents are related
37        to England's King Edward III</aside>
38    </header>
39  </section>
```

## Practicing CSS

### Introduction

Let's practice!

### Descendant Selectors

Descendants can be great-great-grandchildren

19 US Presidents are related to England's King Edward III

This paragraph is in a div, how to make it blue?

Dont make this blue

h1, h2 {color: red}

header h2,  
header span {color: purple;  
font-weight:bold}

# New Color Features in CSS3

- More specifications for color
  - RGBA
  - HSL and HSLA
  - Opacity

What do you think the a is for?

# RGBA Color

- Syntax `rgba(red, green, blue, alpha)`
  - rgb values range from 0 to 255
  - alpha ranges from 0.0 (transparent) and 1.0 (opaque)

```
div#box1 { background-color: rgb(255,0,0) }  
div#box2 { background-color: rgb(0,177,0) }  
div#box3 { color: white;background-color: rgb(0,0,255) }  
div#box1a { background-color: rgba(255,0,0,.25) }  
div#box2a { background-color: rgba(0,177,0,.5) }  
div#box3a { color: white; background-color: rgba(0,0,255,.75) }
```



# HSLA Color

- Hue is a degree on a color wheel (checkout <http://hslpicker.com/> )
  - 0 (or 360) is red, 120 is green, 240 is blue
- Saturation is a percentage, 100% is the full color
- Lightness is a percentage, 0% is dark (black) and 100% is white

```
div#box4 { background-color: hsl(0,50%,50%) }  
div#box5 { background-color: hsl(60, 50%,50%) }  
div#box6 { background-color: hsl(120,50%,50%) }  
div#box4s { background-color: hsl(0,75%,50%) }  
div#box5s { background-color: hsl(60, 75%, 50%) }  
div#box6s { background-color: hsl(120,75%,50%) }  
div#box4a { background-color: hsla(0,50%,50%,.5) }  
div#box5a { background-color: hsla(60, 50%, 50%,.7) }  
div#box6a { background-color: hsla(120,50%,50%,.8) }
```



# Using opacity

- Can now specify a value of 0 (transparent) to 1 (opaque) on an element

```
#box7, .faded { opacity:0.6 }
```

Box 7 Box 8

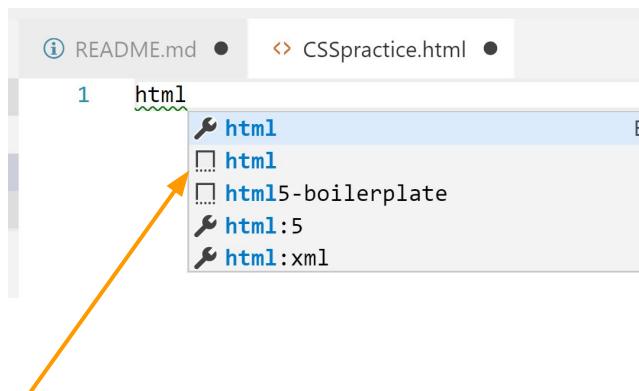
```
67 |     <div id="box7" class="faded">Box 7 </div>
68 |     <div id="box8">Box 8 </div>
69 |
70 |     <ul>
71 |         <li class="faded">opacity 60%</li>
72 |         <li>no opacity set</li>
73 |     </ul>
```

- opacity 30%
- no opacity set

# LAB: Add CSS to a project

15 min

- Navigate to the folder \Labs\Ch04-CSS\1adding-CSS
- Open the file README.md and follow steps to add styling to Lorem Ipsum doc



## Lorem Ipsum Generators

let's show some content

- [Hipster Ipsum](#)
- [Cat Ipsum](#)
- [Cupcake Ipsum](#)

### Hipster Ipsum

Lore ipsum dolor vegan mlkshk kinfolk keytar, meditation tousled mess mumblecore flexitarian, next level shaman la croix live-edge schlitz palo santo.

Lyft tacos leggings slow-carb edison bulb kale chips af yuccie bitters everyday vegan. Thundercats poke flexitarian hell of. Marfa roof party tacos chia, intellig

Meh swag drinking vinegar hot chicken sustainable. Bitters prism XOXO, 8-bit fixie post-ironic, authentic migas air plant man bun. Polaroid quinoa poke post-

XOXO stumptown palo santo blue bottle, vice you probably haven't heard of th beer fam kogi. Banjo tattooed listicle cred turmeric master cleanse selvage lyft r

XOXO pug banh mi affogato knausgaard snackwave whatever roof party vape single-origin coffee palo santo schlitz crucifix trust fund.

Hipster Ipsum

# **Foundations: CSS**



Basics of CSS syntax  
**Order and Specificity**

Text Properties

Font Properties

Backgrounds

CSS Box Model, MBP

Additional Selectors

Chapter Summary

BONUS: Positioning Techniques

BONUS: Styling Tables

# When selectors and rules are **same**..which is applied?

```
7  <style>
8    |   h2 { color: green }
9  </style>
10
11 <link rel="stylesheet" href="test-order.css">
12
13 </head>
14 <body>
15   |   <h2 style="color: red">Hello</h2>
16   |   <h2>Goodbye</h2>
17 </body>
```

# test-order.css ✗

1 h2 { color: purple }



What color is Hello? Goodbye?

Hello

Goodbye

**Order** is important. With the exact same rules, the closest to the code “wins”

# Sometimes different selectors target the same elements

```
8  <style>
9    html body h2 { color: green }
10   h2#myTitle { color: orange}
11 </style>
12
13 <link rel="stylesheet" href="test-specificity.css" />
14
15 </head>
16 <body>
17   <h2 style="color: red">Hello</h2>
18   <h2>Goodbye</h2>
19   <h2 id="myTitle">I've got id!</h2>
20 </body>
```

# test-specificity.css ✗

1 | body h2 { color: purple }

**Specificity** is important.  
The most specific “wins”.

# Order of Specificity in CSS

1. MOST specific - inline - it wins!
2. If no inline styles, are there any selectors with IDs?
3. If no IDs, are there any with classes?
4. If HTML tags only, most html tags wins (descendant selectors)

Specificity: (0,0,0,0)

(inline styles , ID, Classes, HTML tags)

IDs are less important than inline

Classes are less important than IDs

Html tags are less important than classes

**Specificity** is important.  
The most specific “wins”.

# Applying specificity rules...

```
8 <style>
9   html body h2 { color: green }
10  h2.myClass { color : black }
11  h2#myTitle { color: orange }
12 </style>
13
14 <link rel="stylesheet" href="test-specificity.css" />
15
16 </head>
17 <body>
18   <h2>First</h2>
19   <h2 style="color: blue">Second</h2>
20   <h2 id="myTitle">Third</h2>
21   <h2 class="myClass">Fourth</h2>
22   <h2 class="myClass" id="myTitle">Fifth</h2>
23   <h2 style="color: red" class="myClass" id="myTitle">Sixth</h2>
24 </body>
```

```
# test-specificity.css ✗
1 | body h2 { color: purple }
```

What color is each h2?

First  
Second  
Third  
Fourth  
Fifth  
Sixth

# A **non-BEST** practice - using !important

- If you really find yourself in a sticky situation...
  - HTML being generated on server with inline styles
  - or insertions by CMS (Content Management System)
  - ...can use **!important** - it overrides inline styles as well

```
7 <style>
8   h2 { color: green !important }
9 </style>
10
11 <link rel="stylesheet" href="test-order.css">
12
13 </head>
14 <body>
15   <h2 style="color: red">Hello</h2>
16   <h2>Goodbye</h2>
17 </body>
```



Hello

Goodbye

green

green

Order is important!  
**LoVe HAte**

# Anchor link pseudo-class

```
/* 1. unvisited link */  
a:link {  
    color: #FF0000;  
}
```

```
/* 2. visited link */  
a:visited {  
    color: #00FF00;  
}
```

```
/* 3. mouse over link */  
a:hover {  
    color: #FF00FF;  
}
```

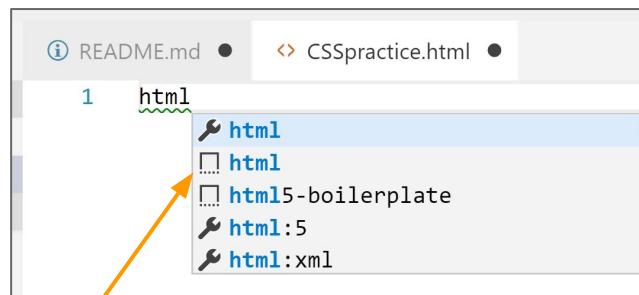
```
/* 4. selected link */  
a:active {  
    color: #0000FF;  
}
```

If 1 comes after others, no hover, visited, active

# Lab: Considering Order and Specificity

5 min

- Navigate to the folder \Labs\Ch04-CSS\2order
- Open the file README.md and follow steps



## Lorem Ipsum Generators

let's show some content

- [Bacon Ipsum](#)
- [Cat Ipsum](#)
- [Cupcake Ipsum](#)

### Bacon Ipsum

Spicy jalapeno bacon ipsum dolor amet do chuck sed velit. Strip ste  
beef sed capicola. Esse burgdoggen sirloin dolore velit. Prosciutto p  
brisket beef ribs. Proident alcatra doner corned beef strip steak. Cill  
shank meatball.

# Foundations: CSS

Basics of CSS syntax

Order and Specificity



**Text Properties**

Font Properties

Backgrounds

CSS Box Model, MBP

Additional Selectors

Chapter Summary

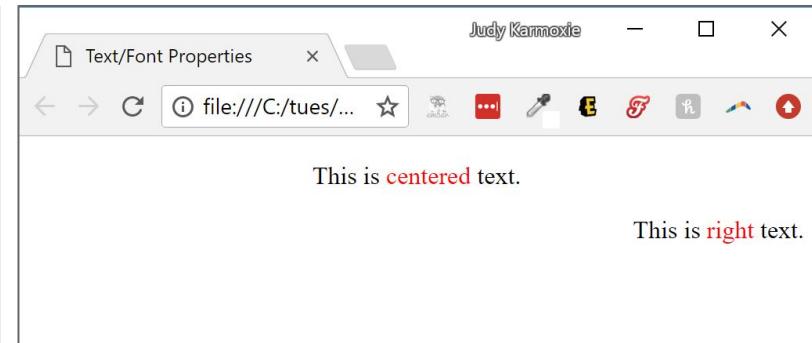
BONUS: Positioning Techniques

BONUS: Styling Tables

# Text Alignment

- When you target a container you can align its content
  - Text - but also other content

```
8 <style>
9   .centered { text-align: center}
10  .right {text-align: }
11    center
12    end
13    inherit
14    initial
15    justify
16    left
17    right
18    start
19    unset
20
21    span {color: red}
22 </style>
23 </head>
24 <body>
25   <p class="centered">This is <span>centered</span> text.</p>
26   <p class="right">This is <span> right </span> text.</p>
```



[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Text](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Text)

# Turning off underlines for links with text-decoration

- Default rendering of links are blue underlined text

```
<ul>
  <li><a href="page1.html">Page 1</a></li>
  <li><a href="page2.html">Page 2</a></li>
</ul>
```

- [Page 1](#)
- [Page 2](#)

```
a {
  text-decoration: none;
}
```

- [Page 1](#)
- [Page 2](#)

# Other text-decoration

- Other values for text decorations

```
h1 {  
    text-decoration: overline;  
}
```

Hello

```
h2 {  
    text-decoration: line-through;  
}
```

~~Goodbye~~

```
h3 {  
    text-decoration: underline;  
}
```

Again

# Text Transformation to adjust text case

- Helpful when you receive data that is stored inconsistently

```
.uppercase {  
    text-transform: uppercase;  
}  
  
.lowercase {  
    text-transform: lowercase;  
}  
  
.capitalize {  
    text-transform: capitalize;  
}
```

# Other text properties

- Text Indentation
- Letter Spacing
- Text Direction
- Word Spacing
- Check out a list of others:  
[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Text](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Text)

# Turning Off bullets

- You can also turn off bullets

```
<nav>
  <ul>
    <li>One</li>
    <li>Two</li>
    <li>Three</li>
  </ul>
</nav>
```

```
nav ul li { list-style: none }
```

One  
Two  
Three

- Navigate to the folder \Labs\Ch04-CSS\3text
- Open the file README.md and follow steps to adjust your text content

## **Lorem Ipsum Generators**

- [BACON IPSUM](#)
- [CAT IPSUM](#)
- [CUPCAKE IPSUM](#)

### **Bacon Ipsum**

Spicy jalapeno bacon ipsum dolor amet do chuck sed velit. Strip steak pancetta landjaeger, nisi sausage pork belly pariatur boudin do beef sed capicola. Esse burgdoggen sirloin dolore velit. Prosciutto pork loin ut nulla frankfurter cillum dolore eu aute turkey salami brisket beef ribs. Proident alcatra doner corned beef strip steak. Cillum doner ad, aliquip incididunt short loin fatback swine ribeye ut shank meatball.

Buffalo consectetur reprehenderit sunt. Culpa eu turkey, lorem meatloaf spare ribs labore enim. Minim salami tongue, cupidatat alcatra esse brisket pig. Voluptate ball tip dolore consectetur chuck jowl pastrami anim tri-tip ullamco beef pork belly in.

Basics of CSS syntax

Order and Specificity

Text Properties



## Font Properties

Backgrounds

CSS Box Model, MBP

Additional Selectors

Chapter Summary

BONUS: Positioning Techniques

BONUS: Styling Tables

# Font Family

- Specify a list of fonts, the browser will choose the first one it can support
  - Best Practice: provide a generic font at the end of the list

```
p {font-family:'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif}
```

```
<style>
p.normal {
    font-style: normal;
    font-weight: normal;
    font-family: 'Courier New', Courier, monospace
        , 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif
        , 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS'
        , 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande'
        , 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif
}
p.italic {
    font-style: italic;
    font-weight: normal;
    font-family: 'Times New Roman', Times, serif
        , 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande'
        , -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto
}
p.oblique {
    font-style: oblique;
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif
        , 'Cambria', 'Cochin', 'Georgia', 'Times', 'Times New Roman',
        , cursive
        , fantasy
}
p {font-family: }
```

Why do you think some have quotes?

Due to names having spaces

# Using Custom Fonts

- Allows custom fonts to be loaded on a webpage
- Tells the browser to download the font from where it is hosted
- Can find free fonts or develop one in-house

<https://www.fontsquirrel.com> (*be careful of other sites, licenses*)

Support by type: [http://www.w3schools.com/cssref/css3\\_pr\\_font-face\\_rule.asp](http://www.w3schools.com/cssref/css3_pr_font-face_rule.asp)

# Usage of @font-face

- The CSS:

```
8  <style>
9      @font-face {
10         font-family: MyCustomFontName;
11         src: url('KaushanScript-Regular.otf');
12         src: url('coolvetica_rg.ttf'); /* IE6-IE8 */
13     }
14
15     span { color: green; font-family: MyCustomFontName, Georgia, serif; }
16 </style>
```

Wouldn't it be nice to have a <span>Custom Font?</span>

Isn't *MODERN WEB DEV* so much fun?

# Font: font-style and font-weight and using normal

- **font-style** for italics, **font-weight** for bold
- Use a value of **normal** to reset if CSS was used elsewhere

```
p.normal {  
    font-style: normal;  
}  
  
p.italic {  
    font-style: italic;  
}  
  
p.oblique { /* not supported by many browsers */  
    font-style: oblique;  
}
```

```
p.normal {  
    font-weight: normal;  
}  
  
p.thick {  
    font-weight: bold;  
}
```

# Using font-size: Absolute and Relative Sizing in CSS

- **Absolute size:** Sets the text to a specified size
  - Users can't change in all browsers
- **Relative size:** Sets the size relative to surrounding elements
  - Allows a user to change the text size in browsers

unit	description	example
px, in	<b>Absolute</b> sizing - pixels and inches - you get most control, but accessibility concerns	body {font-size: 16px} is default
em, rem	<b>em</b> is <b>Relative</b> , based on the default font size's capital letter M <b>rem</b> is the same, but does not compound like em does	1em (16px), 1.1em, .9em
%	<b>Relative</b> , but compounds, % of the parent element	body { font-size: 100% }

# Line Height

- Sets the height of the line being used to hold text
- Preserves area, text can be made bigger without content looking jumpy

normal	A normal line height. This is default
number	multiplied with the current font-size to set the line height
length	A fixed line height in px, pt, cm, etc.
%	A line height in percent of the current font size
initial	Sets this property to its default value. <a href="#">Read about initial</a>
inherit	Inherits this property from its parent element. <a href="#">Read about inherit</a>

# Font shortcut syntax

- You can combine multiple font properties together into a shortcut

```
h1 {  
  font: italic small-caps normal 13px/150% Arial, Helvetica, sans-serif;  
}
```

font-style font-variant font-weight font-size/line-height font-family;

} before font size

required

opt, must be  
after font size  
with slash

required

# Lab: Work with fonts

20 min

- Navigate to the folder \Labs\Ch04-CSS\4fonts
- Open the file README.md and follow steps to adjust your fonts

# **Foundations: CSS**

Basics of CSS syntax

Order and Specificity

Text Properties

Font Properties



## **Backgrounds**

CSS Box Model, MBP

Additional Selectors

Chapter Summary

BONUS: Positioning Techniques

BONUS: Styling Tables

# Backgrounds can be used for styling effects

- Don't use <img src...> if it is just for a styling effect
- There is a lot of flexibility in using background images to affect:
  - Entire background of page
  - Specified elements, such as the header for your page
  - With lists as custom bullets

# CSS Background image and repeat

- **background-image**: give URL to image (or Data URI)
- By default, images repeat in an x and y direction
  - Can control using background-repeat

```
17  div {  
18      background-image: url(..../images/50x50checkmark.png);  
19  }  
20  
21  div#five {  
22      background-repeat: repeat-x;  
23  }  
24  
25  div#six {  
26      background-repeat: repeat-y;  
27  }  
28  
29  div#seven {  
30      background-repeat: no-repeat;  
31  }
```



# CSS background size and color properties

- **background-color** use alpha transparency to reveal image underneath
- **background-size: cover**: image fills element and maintains aspect ratio
- **background-size: 100%**: image fills 100% width of the element

```
<style>
  body {background-image: url('Antarctic_Garden_Hobart_BG.jpg');
        background-repeat: no-repeat;
        background-size: 100%;}
  header {background-color: □rgb(111,111,111,.5);
          width: 400px}

<header>
  <h1>Examples of Backgrounds</h1>
  <p>Header with transparency to read text</p>
</header>
```



# Accessibility and backgrounds

- Be mindful of contrast in colors
  - Use tools to check <https://webaim.org/resources/contrastchecker/>

Color Contrast Checker

Home > Resources > Color Contrast Checker

Foreground Color: #0000FF  
Lightness: [sliders]

Background Color: #FFFFFF  
Lightness: [sliders]

Contrast Ratio: **8.59:1**  
[permalink](#)

**Normal Text**

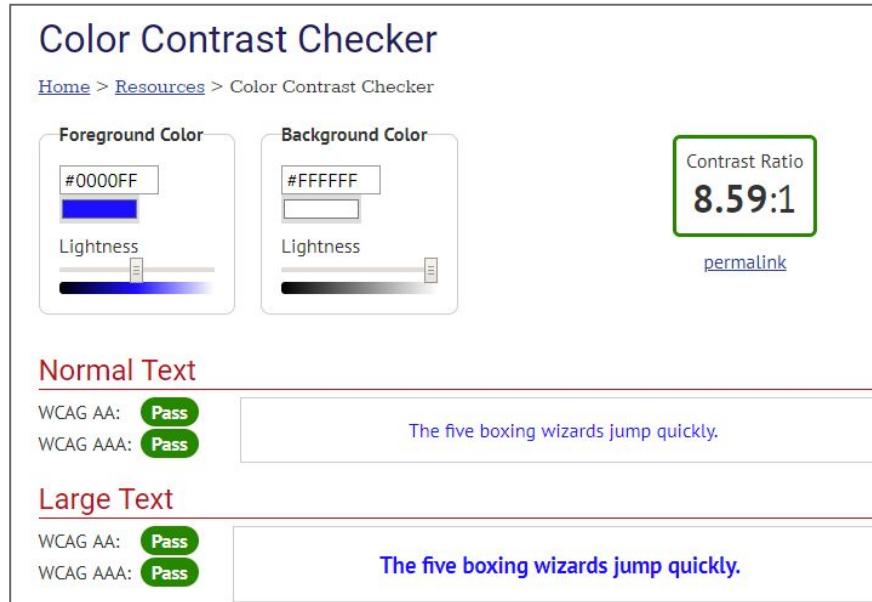
WCAG AA: **Pass**  
WCAG AAA: **Pass**

The five boxing wizards jump quickly.

**Large Text**

WCAG AA: **Pass**  
WCAG AAA: **Pass**

The five boxing wizards jump quickly.



# Background (Image) gradients

- You can have the browser generate gradients for you

```
/* can be treated like a fallback */
background-color: red;

/* will be "on top", if browser supports it */
background-image: linear-gradient(red, orange);
```



# CSS background image options

- **background-position:** how to position the image
- Values:
  - top, bottom
  - left, right
  - center
  - 25% 75% (x and y)
- Interactive Demo:

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

# Background shortcut property

- Can adjust all background style options at once
  - color, image, origin and size, repeat method, and other features

```
background: □pink no-repeat 50%/50% url("../images/50x50checkmark.png");
```

# Using background images with lists

- This gives more control and you can address multiple lines



Oak



Maple



Now we can begin working on lots of happy little things. All you need to pair express yourself to others through painting. Don't fight it, use what happens.

```
ul.trees {  
    list-style-type: none;  
    padding: 0;  
    margin: 0;  
}  
  
ul.trees li {  
    background: url('leaf_icon.jpg') no-repeat left top;  
    height: 54px;  
    padding-left: 44px;  
    padding-top: 3px;  
}
```

# Lab: Add Background using CSS

10 min

- Navigate to the folder \Labs\Ch04-CSS\5backgrounds
- Open the file README.md and follow steps to add CSS backgrounds

# Foundations: CSS

Basics of CSS syntax

Order and Specificity

Text Properties

Font Properties

Backgrounds



**CSS Box Model, MBP**

Additional Selectors

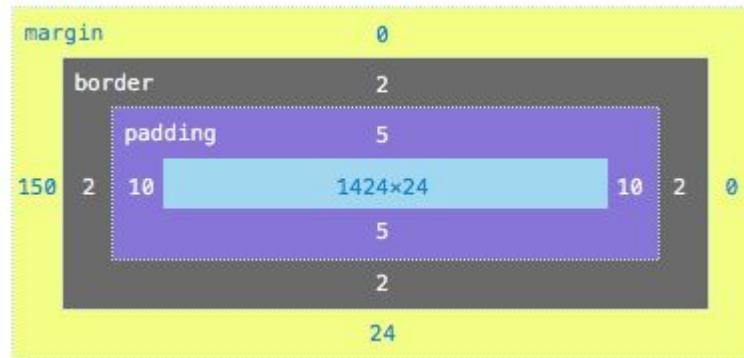
Chapter Summary

BONUS: Positioning Techniques

BONUS: Styling Tables

# CSS Box Model:

- CSS box model **M.B.P** order is important to keep in mind!



- Inner Width & Height based on content
  - or using CSS width & height values (*not inline*)
- **Padding** is transparent - space between content and border
- **Border** appears outside of padding
- **Margin** is transparent area outside all others (*inline only adjusts left/right*)

# Natural Widths of elements

- Inline-elements are wide enough to contain content <span>Hello</span>
- Block-level widths are full width of their container
  - Easiest to see by adding a border

```
h1 { border: 1px solid black}
```



# Widths can be changed in CSS

- You can use the width property
  - pixels
  - % (of the parent container)
  - em (based on elements font size)

```
p.box {  
    width: 100px;  
    background-color: pink;  
    border: 3px solid green  
}
```

This is a  
paragraph

# Height of Block-level Elements

- The vertical dimension of block-level elements changes
  - based upon the content
  - grows as more content is added
  - perhaps via AJAX
- Can also set using **height** property
  - Doesn't work with inline - could use line-height

# Inline & inline-block

- Inline does not allow height
- inline-block allows height & width

```
15  span {  
16      [css] Property is ignored due to the display. With 'display: inline;  
17      e', the width, height, margin-top, margin-bottom, and float propert  
18      ies have no effect.  
19  
20  Specifies the height of the content area, padding area or border area (depending on  
21  'box-sizing') of certain boxes.  
22      height:20px;  
23      display: inline;  
  
          .block {  
            display: block;  
            height:100px;  
            width:100px;  
          }  
  
          .inline-block {  
            display: inline-block;  
            height:100px;  
            width:100px;  
          }  
        }
```

## Inline

Here we have some inline text in a span And some Lorem Ipsum text.

## Inline-block: Can set height and width

Here is a span with class setting display: inline-block  
and height and width text

Lorem ipsum  
dolor sit amet, consectetur adipiscing elit.

## Block: new lines before and after

Here is a span with class setting display: block and height and width

Mauris tempus  
turpis id ante  
mollis  
dignissim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

# Using the display property

```
<p id="one">One</p>
<p id="two">Two</p>
<p id="three">Three</p>
<p id="four">Four</p>
```

- We can affect the natural display of elements using CSS



```
21  #one, #two {
22    display: inline-block
23  }
24
25  #three {
26    display: none;
27  }
```

# Setting Borders

- Borders are made up of style, color and width
  - Default black and 1px

Style is only required portion



```
h1,  
p {  
    border-style: solid;  
}  
  
p {  
    border-color: green;  
    border-width: 4px;  
}
```

# CSS border shortcut

- You can set the border-width, border-color, and border-style in one step:

```
p.all {  
    border: 1px dashed pink  
}
```

with shortcut must have all 3

This is a paragraph

# CSS border specific sides

- You target an individual side
  - **as shortcuts:** border-top, border-right, border-bottom, border-left
  - or specifically: border-top-color, border-top-style, etc.

```
p.specific {  
    border: 1px dashed □pink;  
    border-left: ■red 4px solid;  
}
```

|This is a paragraph|

order is important, if switched -  
shortcut overrides left border

# Creating rounded corners with border-radius

- The border-radius syntax

border-radius: top-left top-right bottom-right bottom-left;

- Can use different number of values similar to padding and margin

Box 1:  
Radius is set for all 4 sides  
border-radius: 10px;

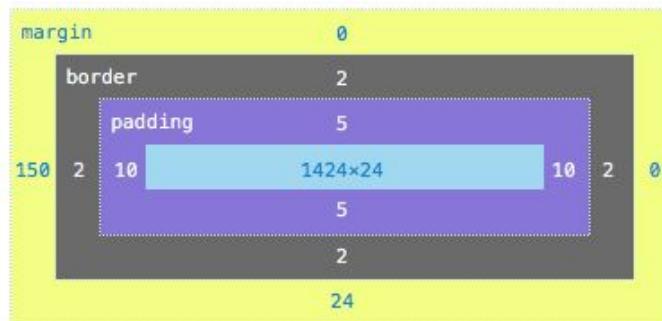
Box 2:  
top-left-and-bottom-right | top-right-and-bottom-left  
border-radius: 10px 5%;

Box 3:  
top-left | top-right-and-bottom-left | bottom-right  
border-radius: 2px 4px 2px;

Box 4:  
top-left | top-right | bottom-right | bottom-left  
border-radius: 1px 0 3px 4px;

# Margin and Padding also can be specific or shortcut

- Can use -top, -bottom, -left, -right or short-cuts



```
div {  
    margin: 0 0 24px 150px;  
    padding: 5px 10px;  
    border: 2px solid black; /* needs all 3  
 */  
}  
  
p {margin-top: 10px; margin-bottom: 5px}
```

## Margin & Padding Shortcuts:

4 values: top, right, bottom, left (like a clock)

3 values: top left&right bottom

2 values: top&bottom left&right

1 value: same all way around

# Margin Positioning



VIEW OUR JEWELRY   OUR SERVICES

- Recall margin is on the outside of the box
- Positive and negative values can be used to move the box around
  - used here to make the image “pop” out of the box



```
14    <header>
15      <div id="logobox">
16        
17      </div>
18    </header>
```

```
<style>
#logobox {
  margin-left: -33px
}

header {
  background-color: #lightgray;
  width: 50%;
  margin: 20px auto;
}
</style>
```

# How do we center content?

This is centered content in a div

This div is centered inside of body

This is centered content in a div

This div is centered inside of body

# Centering content

- Can use **text-align: center** on parent OR **margin: 0 auto**

```
9 <style>
10   div {padding-top: 5px; padding-left: 10px}
11
12   #one {text-align: center; border: 2px solid green;}
13
14   #two {margin: 20px auto; width: 400px; border: 4px dashed pink}
15 </style>
```

```
<body>
  <div id="one">
    This is centered content in a div
  </div>

  <div id="two">
    This div is centered inside of body
  </div>
```

# Lab: Add Use of MBP

10 min

- Navigate to the folder \Labs\Ch04-CSS\6box-model
- Open the file README.md and follow steps

# Foundations: CSS

Basics of CSS syntax

Order and Specificity

Text Properties

Font Properties

Backgrounds

CSS Box Model, MBP



## Additional Selectors

Chapter Summary

BONUS: Positioning Techniques

BONUS: Styling Tables

# Attribute Value Selectors: *has*, *begins*, *ends*

- Target elements that have an attribute

```
/* All links with titles */  
a[title] {background-color:yellow }
```

- Target elements that begin with an attribute value

```
/* All internal links */  
a[href^="#" ] {color:green}
```

- Target elements that end with an attribute value

```
/* All links to PDFs - use background image */  
a[href$=".pdf"] {background: url("img/pdf.png") }
```

# Attribute Value Selectors: *contains*

- Target elements with titles that contain item

```
/* All links with title that contain item */  
a[title*='item'] {font-weight:bold }
```



```
<a href="#item1" title="I am item 1">Item 1</a> ←  
<a href="#item2" title="Item">Item 2 is internal</a>  
<a href="#item3.pdf">This is a pdf</a>  
<a href="#item4" title="I am item 4">Item 4</a> ←
```

# Child Selectors

- Selects the right side, only if it is an immediate child of left side

```
body > span {  
    color: orange;  
}
```



```
14  <body>  
15      <p>  
16          <span> I am a span, child of paragraph </span>  
17      </p>  
18      <span>I am a span, child of body </span>  
19  </body>
```

A red arrow points from the text "I am a span, child of body" in line 18 towards the CSS selector "body > span".

# Sibling Combinators

- Adjacent Sibling: uses + symbol, sibling that immediately follows
- General Sibling: uses ~ symbol, siblings can be separated by elements

```
h1 + h2 { color: red }
h1 ~ h2 { font-style: italic }
```



```
27 <h1>Page Title h1</h1>
28 <h2>An h2</h2>
29 <p>some words in a paragraph</p>
30 <h2>Another h2</h2>
31 <p>some words in a paragraph</p>
32 <p>some words in a paragraph</p>
```

*An h2*

some words in a paragraph

*Another h2*

some words in a paragraph

some words in a paragraph

# Pseudo Classes and Pseudo Selectors

- Pseudo-classes are keywords that indicate the state of an element - one :  
a:link, a:visited, a:hover, a:active
- Pseudo-elements - two ::
  - Only apply to specific parts of the document. CSS2 used single :  
::after        ::before        ::first-letter        ::first-line

# **Foundations: CSS**

Basics of CSS syntax

Order and Specificity

Text Properties

Font Properties

Backgrounds

CSS Box Model, MBP

Additional Selectors



## **Chapter Summary**

BONUS: Positioning Techniques

BONUS: Styling Tables

# Chapter Summary

In this chapter we have looked at:

- Basics in working with CSS
- Ways to include CSS for screen and print
- CSS HTML selectors, classes, and ids
- Understanding why rules are overridden
- Text Properties, Font Properties, Backgrounds
- CSS Box Model, MBP
- Bonus: Different positioning techniques

# **Foundations: CSS**

Basics of CSS syntax

Order and Specificity

Text Properties

Font Properties

Backgrounds

CSS Box Model, MBP

Additional Selectors

Chapter Summary



**BONUS: Positioning Techniques**

**BONUS: Styling Tables**

# Floats

- Floating content allows text and inline elements to wrap around it
  - It is taken from normal flow of web page, but still a part of the content

```
22  <p class="box">This is a box floated left</p>
```

```
23  | <p> This is some other text that will be displayed around the floated
```

```
24  <p>Quarterdeck main sheet piracy scuttle run a shot across the bow g
```

This is a box floated left

This is some other text that will be displayed around the floated box

Quarterdeck main sheet piracy scuttle run a shot across the bow gun hearties Jack Tar grapple lateen sail. Tackle ballast keelhaul warp jack six pounders swab brigantine scourge of the seven seas yo-ho-ho. Strike colors log loot six pounders crack Jennys tea cup cog hogshead cable Jack Ketch starboard. Hulk Plate Fleet league loaded to the gunwalls gunwalls hogshead tender



What CSS is needed to make this work?

```
.box {float:left}
```

# Floats

```
<div class="floats" style="width: 500px; border: 2px solid blue">  
  <p class="a">A</p>  
  <p class="b">B</p>  
  <p class="c">C</p>  
</div>
```

- Can float left or right - no center float
- Float left or right pushes element to the right or left in parent container

```
p {width:100px; height:100px; background-color:yellow; border:1px solid black}
```

```
p.a {float: left;}  
p.b {float:right}  
p.c {float:right}
```



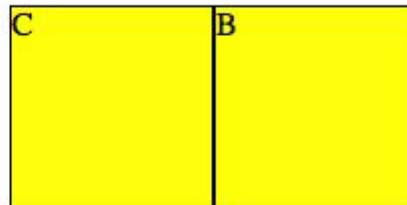
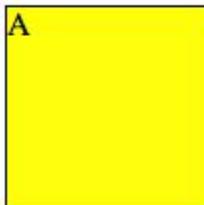
# Side Effect of Float

- If only item in container, no vertical dimension given to parent!

```
<div class="floats" style="width: 500px; border: 2px solid blue">  
  <p class="a">A</p>  
  <p class="b">B</p>  
  <p class="c">C</p>  
</div>
```



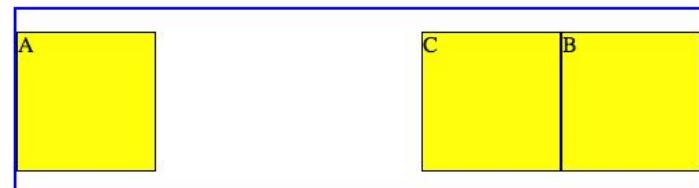
How can you  
fix this?



# Fixing the Effect of Float

- To fix
  - give parent dimension width/height - `div { height:150px }`
  - float the parent (could affect other styles) - `div {float: left}`
  - use clear: both on another element - `<br style="clear:both">`

```
38 <div class="floats" style="width: 500px; border: 2px solid blue">
39   <p class="a">A</p>
40   <p class="b">B</p>
41   <p class="c">C</p>
42   <br style="clear: both; line-height: 0px" />
43 </div>
```



# Positioning

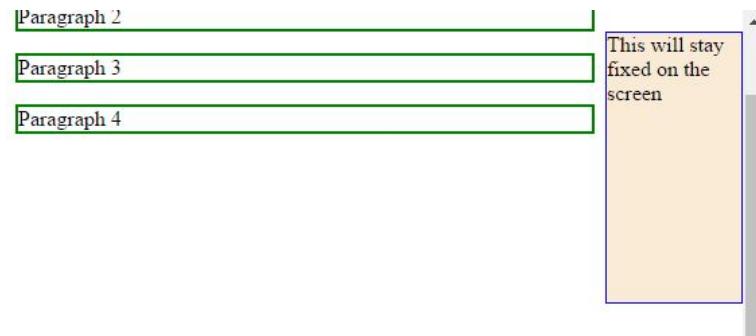
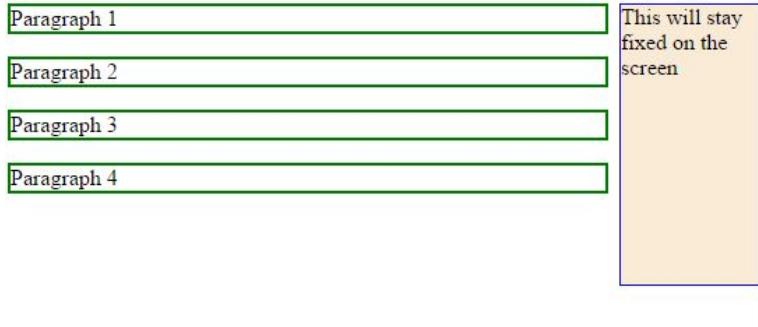
- static - normal positioning, this is the default
- fixed - stays fixed relative to viewport
- relative - can move item, leaves hole
- absolute - can move item, does not leave hole

# Fixed Positioning

- Content stays in position as user scrolls
- Fixed uses the top-left corner as 0,0

```
#fixed { position: fixed; right: 0px }
```

```
p { border: 2px solid green; width: 80% }
```



# Position: relative

- First, set position: relative
- Then, move by giving values for top, bottom, left or right
- Leaves a hole behind (like cutting out a coupon from a paper)

```
28 <body>
29   <div id="one">
30     <div class="floatbox" >
31       Dessert Ipsum
32     </div>
33     <p>Wafer bear claw bear claw.
34       Jujubes ice cream lollipop
35   </div>
36 </body>
```

Dessert  
Ipsum

Wafer bear claw bear claw. Oat cake dessert macaroon chocolate cotton candy powder. Jujubes ice cream lollipop candy tootsie roll. Wafer bear claw bear claw. Oat cake dessert macaroon



What CSS is required?

```
9 .floatbox {
10   float:left;
11   width:100px;
12   background-color:#f0f0f0;
13   margin:0; border:5px solid #f00;
14   padding:5px;
15   position: relative;
16   left: -30px;
17   top: 5px;
18   text-align: center }
```

# Position: absolute

Dessert  
Ipsum

Wafer bear claw bear claw. Oat cake dessert macaroon chocolate cotton candy powder. Jujubes ice cream lollipop candy tootsie roll. Wafer bear claw bear claw. Oat cake dessert macaroon

- Similar to relative positioning, but `position: absolute`
- Does NOT leaves a hole behind - other content filles space
- (0,0) is based on **nearest ancestor** that is **positioned** (or viewport)

```
27 <body>
28   <h1>Positioning - absolute positioning</h1>
29   <div id="one" style="position:relative">
30     <div class="floatbox" >
31       Dessert Ipsum
32     </div>
33     <p>Wafer bear claw bear claw. Oat cake c
34       Jujubes ice cream lollipop candy toc
35   </div>
```

```
14   position: absolute;
15   left: -33px;
16   top: -39px;
```

# Foundations: CSS

Basics of CSS syntax

Order and Specificity

Text Properties

Font Properties

Backgrounds

CSS Box Model, MBP

Additional Selectors

Chapter Summary

BONUS: Positioning Techniques



**BONUS: Styling Tables**

# Tables styling with CSS

- Default has no borders, th is bold

Volunteers		
Name	Email	Phone
Joyce	joyce@email.com	412-555-1212
Bill	bill@rmail.com	412-555-3333

- Good to use CSS for readability

Volunteers		
Name	Email	Phone
Joyce	joyce@email.com	412-555-1212
Bill	bill@rmail.com	412-555-3333

```
table {  
    border-collapse: collapse;  
    border: 2px solid #rgb(200,200,200);  
    letter-spacing: 1px;  
    font-size: 0.8rem;  
    margin-bottom: 20px;  
}  
  
td, th {  
    border: 1px solid #rgb(190,190,190);  
    padding: 10px 20px;  
}  
  
th {  
    background-color: #rgb(235,235,235);  
}  
  
td {  
    text-align: center;  
}  
  
tr:nth-child(even) td {  
    background-color: #rgb(250,250,250);  
}  
  
tr:nth-child(odd) td {  
    background-color: #rgb(245,245,245);  
}
```

# Modern Web Development

## Chapter 5

Responsive Web Design

Flexbox



# Chapter Objectives

In this chapter we will look at:

- How grids and media queries are used for Responsive web design
- How time can be saved using LESS or Sass
- What is Flexbox
- Getting Started with Bootstrap 4

# Responsive Web Design



## Concepts of Responsive Web Design

LESS/Sass

Getting Started with Bootstrap

Bootstrap Layout

Displaying Content with Bootstrap

Bootstrap Components

# Responsive Web Design

- Automatically adjust website to look good across devices
- Can be achieved by using only HTML and CSS:
  - Setting the viewport
  - Media Queries
  - Grid view
  - Relative sizing
- Frameworks have evolved to make it easier to achieve
  - Examples: Bootstrap and Material Design



# Media Queries: CSS3 @media ()

```
@media (max-width: value)  
@media (min-width: value)
```

- We can set breakpoints
  - Design varies for different sized screens

```
/* Extra small devices (phones, 600px and down) */  
@media only screen and (max-width: 600px) {...}  
  
/* Small devices (portrait tablets and large phones, 600px and up) */  
@media only screen and (min-width: 600px) {...}  
  
/* Medium devices (landscape tablets, 768px and up) */  
@media only screen and (min-width: 768px) {...}  
  
/* Large devices (laptops/desktops, 992px and up) */  
@media only screen and (min-width: 992px) {...}  
  
/* Extra large devices (large laptops and desktops, 1200px and up) */  
@media only screen and (min-width: 1200px) {...}
```

# Having a Grid View is a common approach to RWD

- Web page is divided into columns
- Often has 12 columns
- Total width of 100%
- Adjust as you resize the browser window

Col-3			Col-9								
Col-1											
Col-2		Col-4				Col-6					
Col-8						Col-2	Col-2	Col-2	Col-2	Col-2	Col-2

% of one column = [total width] / [total # of columns]

Ex: one col = 100% / 12 = 8.33%

```
.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%;}
```

# Media Queries

- Mobile First - design mobile before desktop
  - Use @media (`min-width: 768px`)
  - Change styles when width > 768px

```
/* 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%;}
}
```

# Media Queries

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Odio tempor orci dapibus ultrices in. Nec sagittis aliquam malesuada bibendum arcu vitae. Suspendisse purus viverra accumsan in nisl nisi scelerisque eu. Suspendisse in est ante at nibh mauris cursus. Enim ut sem massa Ut</p>	<p>Nunc faucibus a pellentesque sit amet. Et tortor consequat id porta nibh venenatis cras. Proin nibh nisl condimentum id venenatis. Massa tincidunt du ut ornare lectus sit. Et get est lorem ipsum dolor sit amet consectetur adipiscing. Velit aliquet sagittis id consecetur purus. Eget arcu dictum varius dui at consecetur lorem donec massa. Ut tristique et egestas quis ipsum suspendisse ultrices gravida. Matis vulputate enim nulla aliquet porttitor lacus. Elementum eu facilisis sed odio morbi. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Interdum consecetur libero id faucibus nisl tincidunt. Adipiscing elit pellentesque habitant morbi tristique senectus et netus et. At tempore commodo ullamcorper a lacus. Ipsum consequat nisl vel pretium lectus quam id. Nam aliquam sem et tortor consequat id porta. Felis bibendum ut tristique et egestas quis.</p>	<p>Eget sit amet tellus cras. Elit scelerisque mauris pellentesque pulvinar. Venenatis habitant morbi tincidunt. Senectus aliquet id porta nibh venenatis. Massa tincidunt du ut ornare lectus sit. Et get est lorem ipsum dolor sit amet consectetur adipiscing. Ut morbi turpis in tincidunt eu m i augue interdum neque egestas. Sed vitae. Odio tincidunt bibendum tincidunt eu ultrices est pellentesque. Nisi nisi scelerisque eu. Et arcu dictum varius Duis at consectetur lorem donec. Dicitum non sit amet varius id porta</p>	<p>Felis donec et odio pellentesque maiores pellentesque pulvinar commodo. Maecenas sed enim ut sem convallis aenean et. Fermentum odio eu feugiat tristique pretium risus nec feugiat ipsum Elementum consequat nisi quis vel. Amet est placaret in egestas quam adipiscing commodo elit at imperiet dui euismod accumsan sit amet. Ut morbi turpis in tincidunt eu m i augue interdum neque egestas. Sed vitae. Odio tincidunt bibendum tincidunt eu ultrices est pellentesque. Nisi nisi scelerisque eu. Et arcu dictum varius Duis at consectetur lorem donec. Dicitum non sit amet varius id porta</p>	<p>Ultrices tincidunt arcu non sodales neque sodales ut etiam sit. Ac tortor dignissim convallis aenean et. Fermentum leo vel non. Vitae purus in mollis nunc sed id. A erat nam at maeccenas. Lectus urna. Facilisi nibh cras pulvinar fermentum erat imperiet sed euismod nisi. Ligula ullamcorper nibi cras pulvinar matus risus in hendrerit feugiat. Fames ac turpis in mollis nunc sed id. Feugiat in fermentum posuere eu sem integer urna. Quis auctor elit sed vulputate blandit fringilla phasellus maoris in aliquam sem fringilla ut. Vulputate</p>	<p>Porttitor rhoncus dolor purus non enim. Senectus et netus et malesuada. Fermentum leo vel non. Vitae purus in mollis nunc sed id. A erat nam at maeccenas. Lectus urna. Facilisi nibh cras pulvinar fermentum erat imperiet sed euismod nisi. Ligula ullamcorper nibi cras pulvinar matus risus in hendrerit feugiat. Fames ac turpis in mollis nunc sed id. Feugiat in fermentum posuere eu sem integer urna. Quis auctor elit sed vulputate blandit fringilla phasellus maoris in aliquam sem fringilla ut. Vulputate</p>	<p>Nullam eget felis eget nunc lobortis. Nunc congue nisi vitae semper quis. Leo urna molestie at elementum eu facilisis suspendisse sed odio morbi. id. A erat nam at maeccenas. Lectus urna. Facilisi nibh cras pulvinar fermentum erat imperiet sed euismod nisi. Ligula ullamcorper nibi cras pulvinar matus risus in hendrerit feugiat. Fames ac turpis in mollis nunc sed id. Feugiat in fermentum posuere eu sem integer urna. Quis auctor elit sed vulputate blandit fringilla phasellus maoris in aliquam sem fringilla ut. Vulputate</p>	<p>Integer enim neque volutpat ac tincidunt vitae semper quis. Leo urna molestie at elementum eu facilisis suspendisse sed odio morbi. id. A erat nam at maeccenas. Lectus urna. Facilisi nibh cras pulvinar fermentum erat imperiet sed euismod nisi. Ligula ullamcorper nibi cras pulvinar matus risus in hendrerit feugiat. Fames ac turpis in mollis nunc sed id. Feugiat in fermentum posuere eu sem integer urna. Quis auctor elit sed vulputate blandit fringilla phasellus maoris in aliquam sem fringilla ut. Vulputate</p>	<p>Pretium viverra quam vulputate dignissim eu facilisis suspendisse in est ante in. Vel turpis nunc eget lorem tincidunt. Amet maoris enim commodo torior viverra ipsum. Ut nisl tincidunt. Amet maoris enim commodo torior quis. Ornare arcu odio massa ut sem nulla nunc. pharetra. Et Purus faucibus malesuada purus in. Odio aenean sed adipiscing diam donec massa. Ut tristique et egestas quis ipsum suspendisse ultrices gravida. Matis vulputate enim nulla aliquet porttitor lacus. Elementum eu facilisis sed odio morbi. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Interdum consecetur libero id faucibus nisl tincidunt. Adipiscing elit pellentesque habitant morbi tristique senectus et netus et. At tempore commodo ullamcorper a lacus. Ipsum consequat nisl vel pretium lectus quam id. Nam aliquam sem et tortor consequat id porta. Felis bibendum ut tristique et egestas quis.</p>	<p>Hac habitasse platea dictumst nullam ac tortor vitae purus. Pharetra vel turpis nunc eget lorem dolor sed viverra leo integer. Elit scelerisque maoris pellentesque pulvinar pellentesque habitant morbi. Loborts mattis aliquam faucibus purus in. Odio aenean sed adipiscing diam donec massa. Ut tristique et egestas quis ipsum suspendisse ultrices gravida. Matis vulputate enim nulla aliquet porttitor lacus. Elementum eu facilisis sed odio morbi. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Interdum consecetur libero id faucibus nisl tincidunt. Adipiscing elit pellentesque habitant morbi tristique senectus et netus et. At tempore commodo ullamcorper a lacus. Ipsum consequat nisl vel pretium lectus quam id. Nam aliquam sem et tortor consequat id porta. Felis bibendum ut tristique et egestas quis.</p>	<p>Nascetur ridiculus mus mauris vitae leo integer. Elit scelerisque maoris pellentesque pulvinar pellentesque habitant morbi. Loborts mattis aliquam faucibus purus in. Odio aenean sed adipiscing diam donec massa. Ut tristique et egestas quis ipsum suspendisse ultrices gravida. Matis vulputate enim nulla aliquet porttitor lacus. Elementum eu facilisis sed odio morbi. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Interdum consecetur libero id faucibus nisl tincidunt. Adipiscing elit pellentesque habitant morbi tristique senectus et netus et. At tempore commodo ullamcorper a lacus. Ipsum consequat nisl vel pretium lectus quam id. Nam aliquam sem et tortor consequat id porta. Felis bibendum ut tristique et egestas quis.</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

@media only screen and (min-width: 768px)

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Odio tempor orci dapibus ultrices in. Nec sagittis aliquam malesuada bibendum arcu vitae.

Egestas purus viverra accumsan in nisl nisi scelerisque eu. Suspendisse in est ante in nibh mauris cursus. Enim ut sem viverra aliquet eget. Mauris a diam maeccenas sed enim ut sem viverra. Odio tempor orci dapibus ultrices in iaculis nunc sed. Faciliis morbi tempus iaculis urna id volutpat lacus laoreet non. Rutrum quisque non tellus orci ac auctor augue. Risus quis varius quam quisque id. Odio aenean sed adipiscing diam donec adipiscing. Nunc congue nisi vitae suscipit tellus mauris a diam maeccenas. Nibh tellus molestie nunc non blandit massa. Auctor augue mauris augue neque gravida in fermentum et sollicitudin. Lacus sed viverra tellus in hac habitasse platea dictumst vestibulum. Ornare massa eget egestas purus viverra accumsan.

Nunc faucibus a pellentesque sit amet. Et tortor consequat id porta nibh venenatis cras. Proin nibh nisl condimentum id venenatis. Massa tincidunt du ut ornare lectus sit. Et get est lorem ipsum dolor sit amet consectetur adipiscing. Velit aliquet sagittis id consecetur purus. Eget arcu dictum varius dui at consecetur lorem donec massa. Ut tristique et egestas quis ipsum suspendisse ultrices gravida. Matis vulputate enim nulla aliquet porttitor lacus. Elementum eu facilisis sed odio morbi. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Interdum consecetur libero id faucibus nisl tincidunt. Adipiscing elit pellentesque habitant morbi tristique senectus et netus et. At tempore commodo ullamcorper a lacus. Ipsum consequat nisl vel pretium lectus quam id. Nam aliquam sem et tortor consequat id porta. Felis bibendum ut tristique et egestas quis.

Eget sit amet tellus cras. Elit scelerisque maoris pellentesque pulvinar pellentesque habitant morbi tristique senectus. Id porta nibh venenatis cras sed felis. Viverra aliquet eget sit amet tellus cras adipiscing enim eu. Ut morbi tincidunt augue interdum velit euismod in pellentesque. Nisi nisi scelerisque eu ultrices vitae

Mobile screen

# Responsive Web Design

Concepts of Responsive Web Design



**LESS/Sass**

Getting Started with Bootstrap

Bootstrap Layout

Displaying Content with Bootstrap

Bootstrap Components

# CSS preprocessor

- A scripting language extends CSS and then compiles into regular CSS
- Advantages:
  - Saves time
  - Reusable codes
  - Easy to maintain
  - Calculations and logic
  - More organized and easy set up
- Sass
- LESS

# Sass



- Syntactically Awesome Style Sheets
- CSS preprocessor
- .sass vs .scss
  - .sass - original
  - .scss - since Sass 3
- Backward compatible
- Ruby based

## Sass Font Example

```
$font-stack: Helvetica, sans-serif;  
$primary-color: #333;  
  
body {  
  font: 100% $font-stack;  
  color: $primary-color;  
}
```

## CSS Output

```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

# Sass

- Nesting of selectors
  - .sass - indentation
  - .scss - brackets { }
- Separate properties
  - .sass - new lines
  - .scss - semicolons

```
//Sass Example
$text-color: #333333
body
color: $text-color
```

```
/* SCSS */
$blue: #3bbfce;
$margin: 16px;

.content-navigation {
  border-color: $blue;
  color: darken($blue, 9%);
}

.border {
  padding: $margin / 2; margin: $margin / 2; border-color: $blue;
```

# LESS



- Leaner Style Sheets
- CSS preprocessor
- Backward compatible
- Javascript based

## LESS Color Example

```
@nice-blue: #5B83AD;  
@light-blue: @nice-blue + #111  
  
#header {  
    color: @light-blue;  
}
```

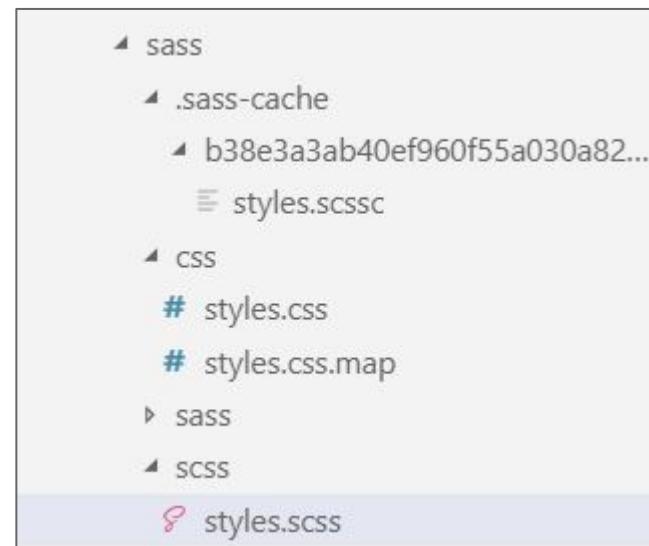
## CSS Output

```
#header {  
    color: #6c94be;  
}
```

# A Process is used to compile Sass or LESS to CSS

- Projects are configured to work with Sass or LESS
- One example using a Ruby GEM
  - Type `sass --watch scss:css`

```
$ sass --watch scss:css
>>> Sass is watching for changes. Press Ctrl-C to stop.
      write css/styles.css
      write css/styles.css.map
>>> Change detected to: scss/styles.scss
      write css/styles.css
      write css/styles.css.map
>>> Change detected to: scss/styles.scss
      write css/styles.css
      write css/styles.css.map
```



- Detects changes and generate css file

# Responsive Web Design

Concepts of Responsive Web Design

LESS/Sass



**Getting Started with Bootstrap**

Bootstrap Layout

Displaying Content with Bootstrap

Bootstrap Components

# Bootstrap <https://getbootstrap.com/>

- Developed by Twitter developers
- Open source toolkit for developing with HTML, CSS, and JS
- Allows you to leverage
  - Sass variables and mixins
  - Responsive grid system
  - Extensive pre-built components
  - Powerful plugins built on jQuery



# Bootstrap 4.2.x

- Uses Sass, where before LESS
- Built with flexbox:
  - Most components use flex
    - if need to add, use **display: flex** or with version 4. => **.d-flex**
- Grid-breakpoints
- [Reboot](#): An opinionated CSS Reset
- Dropped support for [≤ IE 10 and iOS 6](#)
- Dropped pixels, custom builder, glyphicons, grunt

For transparency into our release cycle and in striving to maintain backward compatibility, Bootstrap is maintained under [the Semantic Versioning guidelines](#). Sometimes we screw up, but we adhere to those rules whenever possible.

# 3 Options to Get Bootstrap

1. Download source code from Git

<https://github.com/twbs/bootstrap/releases/download/v4.2.1/bootstrap-4.2.1-dist.zip>

2. BootstrapCDN

Using a CDN - files are cached and loading time is faster

3. Package managers such as npm

npm install bootstrap

# What you get

- With Bootstrap you get access to the CSS
  - minified files (no spaces, newlines)
  - maps (easier to see in devtools)
  - reboot (a CSS reset)
- And JavaScript files
  - optional, used for components and plugins
  - `bootstrap.bundle.js` includes Popper but not jQuery (needed)

```
bootstrap/
└ dist/
    └─ css/
        └─ bootstrap-grid.css
        └─ bootstrap-grid.css.map
        └─ bootstrap-grid.min.css
        └─ bootstrap-grid.min.css.map
        └─ bootstrap-reboot.css
        └─ bootstrap-reboot.css.map
        └─ bootstrap-reboot.min.css
        └─ bootstrap-reboot.min.css.map
        └─ bootstrap.css
        └─ bootstrap.css.map
        └─ bootstrap.min.css
        └─ bootstrap.min.css.map
    └─ js/
        └─ bootstrap.bundle.js
        └─ bootstrap.bundle.js.map
        └─ bootstrap.bundle.min.js
        └─ bootstrap.bundle.min.js.map
        └─ bootstrap.js
        └─ bootstrap.js.map
        └─ bootstrap.min.js
        └─ bootstrap.min.js.map
```

# Bootstrap template

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="../../scripts/bootstrap.min.css">

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="../../scripts/jquery-3.3.1.slim.min.js"></script>
    <script src="../../scripts/popper.min.js"></script>
    <script src="../../scripts/bootstrap.min.js"></script>
  </body>
</html>
```

add before any other  
stylesheets using Bootstrap

jQuery and JS files needed if  
using certain components

# Responsive Web Design

Concepts of Responsive Web Design

LESS/Sass

Getting Started with Bootstrap



**Bootstrap Layout**

Displaying Content with Bootstrap

Bootstrap Components

# Containers

- The most basic layout element
  - Required for Bootstrap grid system

## Bootstrap Page

This part is inside a .container-fluid class.

## Bootstrap Page

This part is inside a .container class.

```
<div class="container-fluid" style="background-color: #tomato">
    <h2>Bootstrap Page</h2>
    <p>This part is inside a .container-fluid class.</p>
</div>
<div class="container" style="background-color: #tomato">
    <h2>Bootstrap Page</h2>
    <p>This part is inside a .container class.</p>
</div>
```

## .container-fluid

- 100% wide all the time
- changes as viewport changes

## .container

- responsive, fixed-width, based on breakpoint

# Bootstrap Grid system

- A series of containers, rows, and columns
- Built with flexbox

	<b>Extra small</b> ≤576px	<b>Small</b> ≥576px	<b>Medium</b> ≥768px	<b>Large</b> ≥992px	<b>Extra large</b> ≥1200px
<b>Max container width</b>	None (auto)	540px	720px	960px	1140px
<b>Class prefix</b>	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
<b># of columns</b>	12				
<b>Gutter width</b>	30px (15px on each side of a column)				<b>Breakpoints:</b> xs, sm, md, lg, xl are used in many places
<b>Nestable</b>	Yes				
<b>Column ordering</b>	Yes				

<https://getbootstrap.com/docs/4.2/layout/grid/#grid-options>

# Auto-layout columns

- Equal-width
  - `col`
- Setting one column width
  - `col-{6-12}`
- Variable width content
  - `col-{breakpoint}-auto`
  - size columns based on the device breakpoint, ex: `col-sm-12, col-xl-1`

# Reordering elements

- Controlling the visual order
- If no order given, natural order used
- Specify numbers, or first or last
  - `.order-{1-12}`
  - `.order-first` and `.order-last`

First, but unordered

Third, but second

Second, but last

```
<div class="container">
  <div class="row">
    <div class="col">
      First, but unordered
    </div>
    <div class="col order-12">
      Second, but last
    </div>
    <div class="col order-1">
      Third, but second
    </div>
  </div>
</div>
```

# Shorthand responsive margin and padding classes

- For xs: {property} {sides}-{size}
- For sm, md, lg, and xl {property} {sides}-{breakpoint}-{size}
- Property is **m** for margin and **p** for padding



What would class="mt-sm-5 mt-md-4 mt-md-3" do?

- Sides are: t - top b - bottom l - left r - right  
y - top & bottom x - left & right blank - all 4 sides
- Sizes are: 0 thru 5

On small devices, a top margin of 3rem, etc.

0 - 0px 1 - .25rem (4px if font-size is 16px) 2 - .5rem (8px if font-size is 16px)  
3 - 1rem (16px if font-size is 16px) 4 - 1.5rem (24px if font-size is 16px)  
5 - 3rem (48px if font-size is 16px) auto - sets margin to auto

# Lab: Layout exercise

- Navigate to folder \Labs\Ch05-RWD\bootstrap\1layout
- Open the README.md file and follow the steps

<h3>Pembroke Welsh Corgi</h3> <p>The Pembroke Welsh Corgi (/kɔːrgi/; Welsh for "dwarf dog") is a cattle herding dog breed which originated in Pembrokeshire, Wales. It is one of two breeds known as a Welsh Corgi. The other is the Cardigan Welsh Corgi.</p>	<h3>Beagle</h3> <p>The beagle is a breed of small hound that is similar in appearance to the much larger foxhound. The beagle is a scent hound, developed primarily for hunting hare (beagling). With a great sense of smell and superior tracking instinct, the beagle is</p>	<h3>Siberian Husky</h3> <p>The Siberian Husky (Russian: Сибирский хаски, lit: Sibirskiy Haski) is a medium size working dog breed that originated in Northeast Asia. The breed belongs to the Spitz genetic family. With proper training,</p>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Responsive Web Design

Concepts of Responsive Web Design

LESS/Sass

Getting Started with Bootstrap

Bootstrap Layout

► **Displaying Content with Bootstrap**

Bootstrap Components

# Bootstrap 4 Text/Typography

- Defaults <https://getbootstrap.com/docs/4.0/content/reboot/>
  - font-size: 16px, line-height: 1.5
  - font-family: "Helvetica Neue", Helvetica, Arial, sans-serif.
- Headers and <p> elements have margin-top: 0
  - <p> have margin-bottom: 1rem , headings have margin-bottom: 1rem

**h1 Bootstrap heading (2.5rem = 40px)**

**h2 Bootstrap heading (2rem = 32px)**

**h3 Bootstrap heading (1.75rem = 28px)**

**h4 Bootstrap heading (1.5rem = 24px)**

**h5 Bootstrap heading (1.25rem = 20px)**

**h6 Bootstrap heading (1rem = 16px)**

# Display Headings stand out more than normal h1-h6

- Larger font-size and lighter font-weight
  - 4 classes to choose from: .display-1, .display-2, .display-3, .display-4

```
<h1 class="display-1">Display 1</h1>
<h1 class="display-2">Display 2</h1>
<h1 class="display-3">Display 3</h1>
<h1 class="display-4">Display 4</h1>
```

Display 1  
Display 2  
Display 3  
Display 4

# Contextual Colors

Use the contextual classes to provide "meaning through colors":

This text is muted.

This text is important.

This text indicates success.

This text represents some information.

This text represents a warning.

This text represents danger.

Secondary text.

This text is dark grey.

Default body color (often black).

```
<div class="container">
  <h2>Contextual Colors</h2>
  <p>Use the contextual classes to provide "meaning through colors":</p>
  <p class="text-muted">This text is muted.</p>
  <p class="text-primary">This text is important.</p>
  <p class="text-success">This text indicates success.</p>
  <p class="text-info">This text represents some information.</p>
  <p class="text-warning">This text represents a warning.</p>
  <p class="text-danger">This text represents danger.</p>
  <p class="text-secondary">Secondary text.</p>
  <p class="text-dark">This text is dark grey.</p>
  <p class="text-body">Default body color (often black).</p>
  <p class="text-light">This text is light grey (on white background).</p>
  <p class="text-white">This text is white (on white background).</p>
</div>
```

# Accessibility

- Be careful with color contrast
  - Bootstraps palette does not always work well with light backgrounds
  - Check for insufficient color contrast  
<https://webaim.org/resources/contrastchecker/>
- Visually hidden content is hidden, but accessible to screen readers

```
<p class="text-danger">  
  <span class="sr-only">Danger: </span>  
  This action is not reversible  
</p>
```

This action is not reversible

on screen, uses text-danger

```
.sr-only {  
  position: absolute;  
  width: 1px;  
  height: 1px;  
  padding: 0;  
  overflow: hidden;  
  clip: rect(0,0,0,0);  
  white-space: nowrap;  
  border: 0;  
}
```

screen-reader.scss:7

# Table classes used in Bootstrap

Combine different table classes to get the effect you want

- .table gives basic table styling
- .table-striped class adds zebra-stripes to a table;
- .table-bordered class adds borders on all sides of the table and cells
- .table-hover class adds a hover effect (grey background color) on table rows

Hover over rows to see hover effect

Name	Email	Phone
Blah	Blah	Blah
Blah	Blah	
Blah	Blah	
Blah	Blah	

```
<div>
  <table class="table table-striped table-bordered table-hover">
    <thead>
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Phone</th>
      </tr>
    </thead>
```

# Responsive Table

The .table-responsive-XX class creates a responsive table which will scroll horizontally on certain breakpoints.

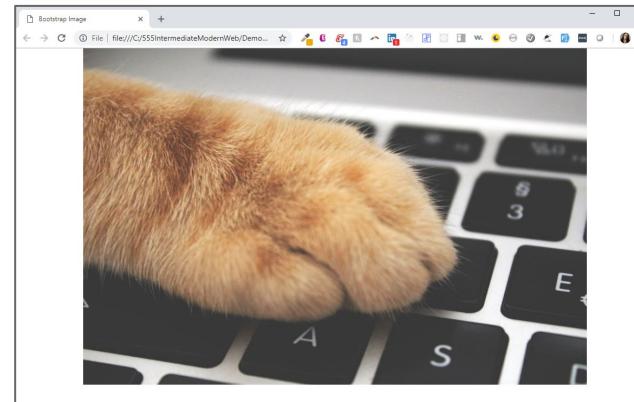
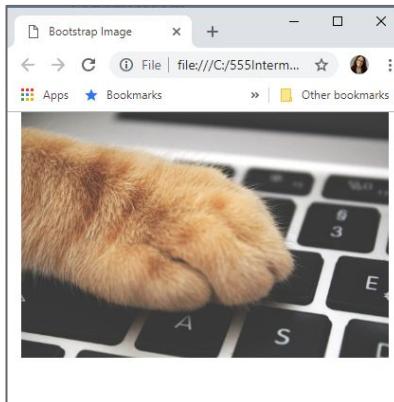
Resize the browser window to see the effect.

Class	Screen Width	Example	Example	Example
.table-responsive-sm	< 576px	Blah	Blah	Blah
.table-responsive-md	< 768px	Blah	<div class="table-responsive-sm"> <table class="table"> <thead> <tr> <th>Class</th> <th>Screen Width</th>	
.table-responsive-lg	< 992px	Blah		
.table-responsive-xl	< 1200px	Blah		

# Responsive images

- Responsive images via `.img-fluid`
  - `max-width: 100%;` and `height: auto;` are applied to the image so that it scales with the parent element

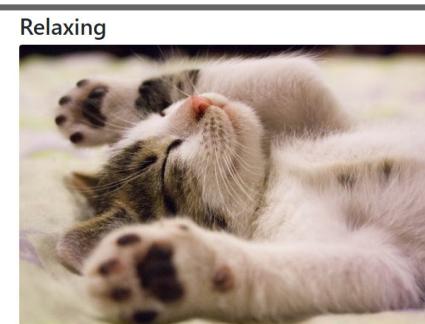
```
<div class="container">
|   
</div>
```



# Image styling

- `.img-thumbnail rounded 1px border`
- `rounded rounded no border`
- `rounded-circle neat circle`

```
<div class="container-fluid">
  <div class="row">
    <div class="col-xs-12 col-sm-6 col-md-4"> <h2>Coding</h2>
      
    </div>
    <div class="col-xs-12 col-sm-6 col-md-4"><h2>Relaxing</h2>
      
    </div>
    <div class="col-xs-12 col-sm-6 col-md-4"> <h2>Sleeping</h2>
      
    </div>
  </div>
</div>
```



# Figure and captions

- .figure-img
- .figure-caption

```
.kitten-4 {  
    border-radius: 15px 50px;  
}
```

```
<figure class="figure col-xs-12 col-sm-6 col-md-4">  
      
    <figcaption class="figure-caption">A kitten is getting attention.</figcaption>  
</figure>
```



A kitten is getting attention.

# Lab: Content exercise

- Navigate to folder \Labs\Ch05-rwd\bootstrap\2content
- Open the README.md file and follow the steps

Pembroke  
Welsh Corgi



The Pembroke Welsh Corgi (/ˈkɔːrɡi/; Welsh for "dwarf dog") is a cattle herding dog breed which originated in Pembrokeshire, Wales. It is one of two breeds known as

Beagle



The beagle is a breed of small hound that is similar in appearance to the much larger foxhound. The beagle is a scent hound, developed

Siberian  
Husky



# Responsive Web Design

Concepts of Responsive Web Design

LESS/Sass

Getting Started with Bootstrap

Bootstrap Layout

Displaying Content with Bootstrap



**Bootstrap Components**

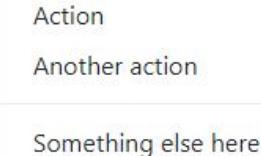
# Bootstrap Components

- Bootstrap provides many components
  - many require jQuery and Popper
- Really nice effects are achievable simply through using class values

Component Name	Description
Alerts	specifies the predefined message for user actions
Badges	Badges are used to highlight the additional information to the content.
Breadcrumb	It is used to show hierarchy-based information for a site.
Buttons	Clickable button to put content such as text and images.
Button group	Allow multiple buttons to be stacked together on a single line.
Cards	Card is a content container which displays a bordered box with some padding around it.
Carousel	A flexible, responsive way to add a slider to your site.
Collapse	Used to show or hide the content.
Dropdowns	Used for displaying links in a list format.
Forms	Used to collect input from user.
Input group	Easily prepend and append text or buttons to the text-based inputs.
Jumbotron	Increases the size of headings and adds a lot of margin for landing page content.
Modal	a child window that is layered over its parent window.
Navs	Provides navigation items for your site in a horizontal menu.
Navbar	Provides navigation headers for your application or site.
Pagination	Used to divide the related content across multiple pages.
Popovers	Similar to tooltip, offering an extended view complete with a heading.
Progress	Shows progress of a process with stacked bars, animated backgrounds, and text labels.
Scrollspy	Used to indicate currently active link in the menu based on scroll position.
Tooltips	Useful when you need to describe a link.

# Navbar and nav-item

- Navbars are fluid by default



```
29 <nav class="navbar navbar-expand-md navbar-light fixed-top bg-light" role="navigation">
30   <a class="navbar-brand" href="#">Kitten Company</a>
31   <ul class="navbar-nav mr-auto">
32     <li class="nav-item active"><a class="nav-link" href="#">Home</a></li>
33     <li class="nav-item"><a class="nav-link" href="#">Links</a></li>
34     <li class="nav-item"><a class="nav-link disabled" href="#">Disabled</a></li>
35     <li class="nav-item dropdown">
36       <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
37         role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
38         Dropdown</a>
39       <div class="dropdown-menu" aria-labelledby="navbarDropdown">
40         <a class="dropdown-item" href="#">Action</a>
41         <a class="dropdown-item" href="#">Another action</a>
42         <div class="dropdown-divider"></div>
43         <a class="dropdown-item" href="#">Something else here</a>
```

# Bootstrap Components

- Jumbotron and display are often used together

```
Welcome to my world!
```

```
<div class="container-fluid">
  <div class="jumbotron">
    <h1 class="display-4">Welcome to my world!</h1>
  </div>
</div>
```

# Alerts

- <https://getbootstrap.com/docs/4.2/components/alerts/>
- .alert
- Alerts can be closed, which removes them from the DOM
- .alert-dismissible

Holy guacamole! Check out alert warning and close it by clicking on X.

X

```
<div class="alert alert-warning alert-dismissible fade show" role="alert">
  <strong>Holy guacamole!</strong> Check out alert warning and close it by clicking on X.
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>
```

# Chapter Summary

In this chapter we have looked at:

- How grids and media queries used for Responsive web design
- How time can be saved using LESS or Sass
- What is Flexbox
- Getting Started with Bootstrap

# **Modern Web Development**

## **Chapter 6**

Modern JavaScript Essentials



# Chapter Objectives

In this chapter we will look at:

- Review of JavaScript fundamentals
- New syntax in EcmaScript 6
- JavaScript arrays and functions
- Objects and properties
- Arrow functions and this



## ECMAScript & JavaScript

Declaring Variables with var, let, const

JavaScript Types

Conditionals and Looping

Creating and Calling Functions

Working with Arrays

Objects and properties

Arrow Functions, this, and More ES6 Syntax

# JavaScript ("JS")

- Invented by Brendan Eich in 1995 for Netscape Navigator
  - written in 10 days! and began as “Mocha” released as “LiveScript”
  - renamed in 1995 to JavaScript (due to Java’s popularity)
- A weakly typed, dynamic programming language
- Needs an engine
  - Browsers provide engine and allow for dynamic behavior
  - Node.js provides environment for outside of browser

# JavaScript in the browser

What are some of the things we might use JavaScript for in the browser?



Validation

Calculations

Respond to user events

AJAX

# ECMAScript is the JavaScript language specification

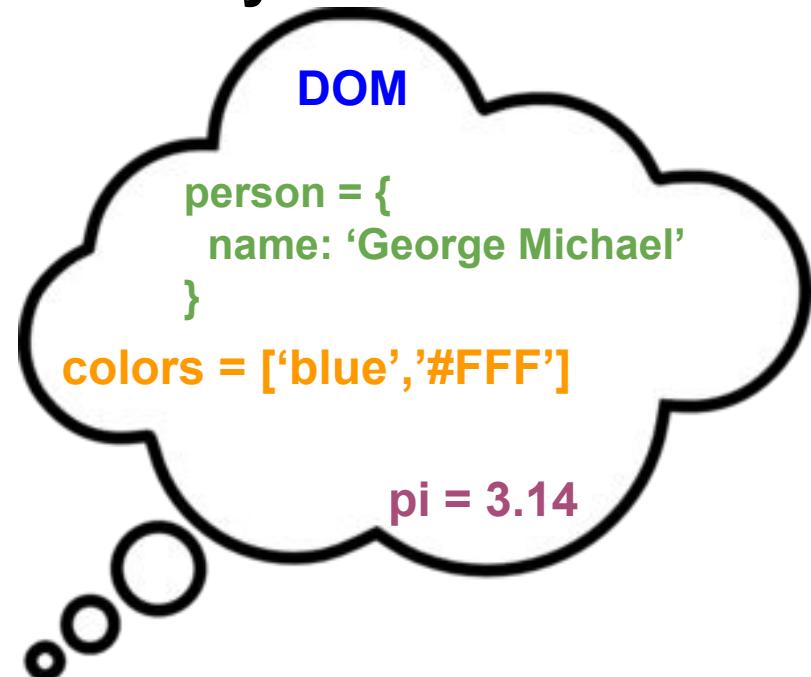
- 1996: Netscape submitted JS to <https://www.ecma-international.org/>
- 1999: **ECMAScript 3** - baseline for modern JS
  - Version 4 was cancelled: held up by: MS, Yahoo, Adobe
- Open source, browser and developer communities
  - 2005: AJAX in browsers
  - popular libraries evolved: jQuery, Dojo, MooTools, Prototype
- 2009: **ECMAScript 5th Edition**, released
  - what modern browsers support now
- 2015: **ES6 (ES2015) (ES6 Harmony)**
  - syntax slowly being supported by browsers, supported in Node

# EcmaScript 6 (2015) - Harmony

- New scoping for variable references
- New methods for strings, arrays, numbers, dates and more
- New syntax: looping, object creation/destructuring
- Arrow Functions, Classes and Modules syntax
- Only partially supported in browsers
  - <https://kangax.github.io/compat-table/es6/>
  - Transpilers turn ES6 into ES5 (Babel) <http://babeljs.io/>

# JavaScript involves working with memory

- All that we define and work with happens in memory
- We need to be **mindful** that memory is limited when we create and use objects



# Modern JavaScript

ECMAScript & JavaScript

## ➤ Declaring Variables with var, let, const

JavaScript Types

Conditionals and Looping

Creating and Calling Functions

Working with Arrays

Objects and Properties

Arrow Functions, this, and More ES6 Syntax

# Declaring variables in JavaScript

- We create variables to hold **values** or **references** to objects in memory
  - Can use **var**, **let**, and **const** to declare
  - Choose easy to understand and short names, lowerCamelCase
- The equal sign is called the **assignment operator**
  - What's on the right is creating an object in memory

```
14  var aVarString = 'hello universe'  
15  let aLetString = 'IN';  
16  const aConstString = 'hello constant ref';
```

variables are not specified by type, as in Java or TypeScript

# Declaring with var is the old way - and can lead to bugs

- Could have two scripts in browser, both declaring same variable
  - 1st is overwritten in memory - without any warning / error

The screenshot shows a browser developer tools interface with three panels:

- conflicting-scripts.html**: The main HTML file containing two script tags pointing to `variables-mine.js` and `variables-yours.js`.
- JS variables-mine.js**: Contains the code `var myFavoriteMusician = 'Les Claypool';`
- JS variables-yours.js**: Contains the code `var myFavoriteMusician = 'Natalie Cole';`
- A modal dialog box displays the message "This page says" followed by "Natalie Cole" and an "OK" button.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Variables</title>
5   <script src="variables-mine.js"></script>
6   <script src="variables-yours.js"></script>
7 </head>
8 <body>
9   <script>
10    alert(myFavoriteMusician);
11   </script>
12 </body>
13 </html>
```

```
JS variables-mine.js
1 var myFavoriteMusician = 'Les Claypool';

JS variables-yours.js
1 var myFavoriteMusician = 'Natalie Cole';

This page says
Natalie Cole
OK
```

# ES6: Let (and const) stop accidental dupe declarations

- Using let or const prevents re-declaring the variable
  - Catches a conflicting scripts redeclaring the same variable name

The screenshot shows two tabs in a browser's developer tools: "variables-mine.js" and "variables-yours.js". Both tabs have a yellow "JS" icon. The "variables-mine.js" tab contains the code "let myFavoriteMusician = 'Les Claypool';". The "variables-yours.js" tab contains the code "let myFavoriteMusician = 'Natalie Cole';". A vertical ellipsis "..." is positioned between the two tabs.

```
JS variables-mine.js ×
...
JS variables-yours.js ×
1  let myFavoriteMusician = 'Les Claypool';
1  let myFavoriteMusician = 'Natalie Cole';
```

✖ Uncaught SyntaxError: Identifier 'myFavoriteMusician' has already been declared at variables-yours.js:1

# Best Practice: Declare with const and let

- They indicate intention:
  - **let** variables can be re-assigned, values may change
  - **const** variables cannot be re-assigned (constant variable, not constant value)
- They have block scope

```
15  let changingValue = true;
16  const pi = 3.14;
17  const myMother = {
18      name: 'Mary',
19      age: 82
20  }
```

Blocks of code are between { }

```
25  {
26      let x = 23;
27      var y = 24;
28      const z = 25;
29  }
30
31  // X is not defined console.log('x', x);
32  // Z is not defined console.log('z', z);
33  console.log('y', y);
```

ECMAScript & JavaScript

Declaring Variables with var, let, const



## JavaScript Types

Conditionals and Looping

Creating and Calling Functions

Working with Arrays

Objects and Properties

Arrow Functions, this, and More ES6 Syntax

# There are 7 types defined in ECMAScript

- 6 data types that are **primitives** (and immutable: can't be updated):
  - number
  - string
  - boolean
  - undefined
  - null
  - Symbol (new in ECMAScript 6)
- Object
  - everything else is an object, and is by default mutable
  - includes arrays, dates

# JavaScript uses dynamic typing

- No type declaration is used for variables
  - but what they point to has a type
- Can use same variable to point to values of different types

```
9  let x; console.log(typeof x); // undefined  
10 x = 3; console.log(typeof x); // number  
11 x = 'abc'; console.log(typeof x); // string  
12 x = true; console.log(typeof x); // boolean
```

The typeof operator  
returns a string  
describing the type



# Variables start off as **undefined**

- **undefined** - a value/object has not yet been assigned

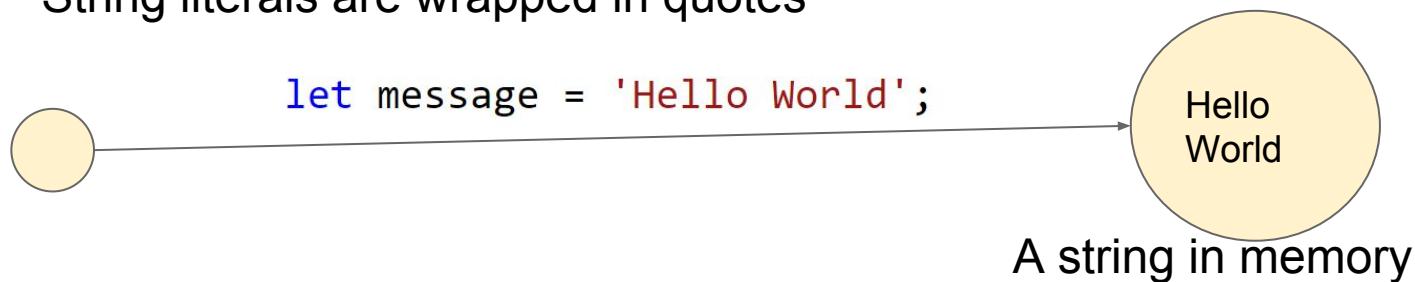


let y

```
let y;    console.log(typeof y);      undefined
```

# Variables are assigned using = sign

- Until a variable is pointing at something, there is no type
  - String literals are wrapped in quotes



```
let message = 'Hello World';
console.log(typeof message);
```

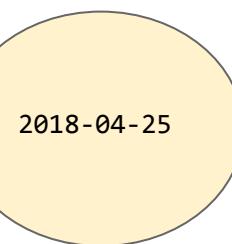
string

# Creating objects with the keyword `new`

- The keyword `new` is used with constructor functions
  - Capitalized by convention
  - used to create objects in memory



```
let today = new Date();
```



An object in memory

```
let today = new Date();
console.log(typeof today);
console.log(today);
```

object

2018-04-24T19:10:40.645Z

# The `typeof` operator returns a string

```
console.log(typeof myVariable)
```

Type	Result
Undefined	“undefined”
Null	“object” (see below)
Boolean	“boolean”
Number	“number”
String	“string”
Symbol (new in ES6)	“symbol”
Host object (provided by the JS environment)	<i>Implementation-dependent</i>
Function object	“function”
Any other object	“object”

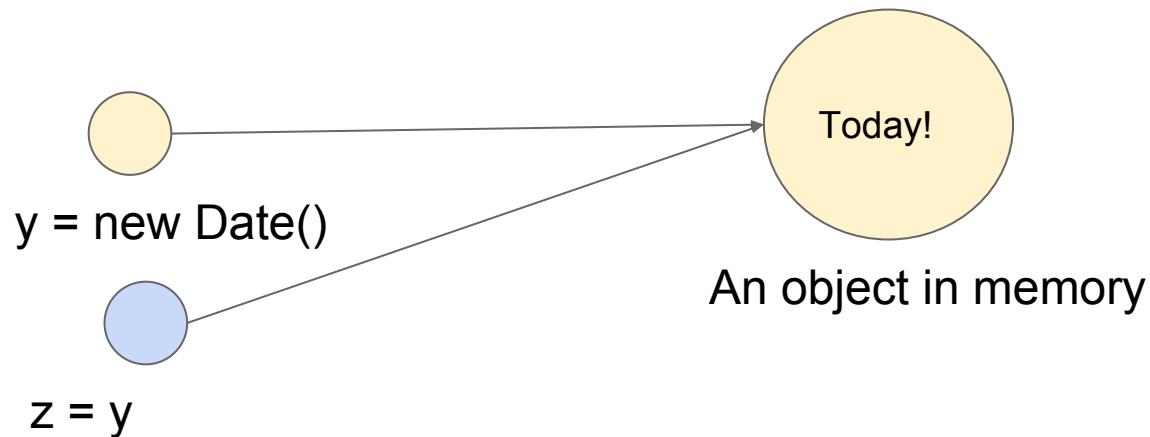
# Objects stay in memory in JavaScript

- As long as some variable points to the object it stays in memory



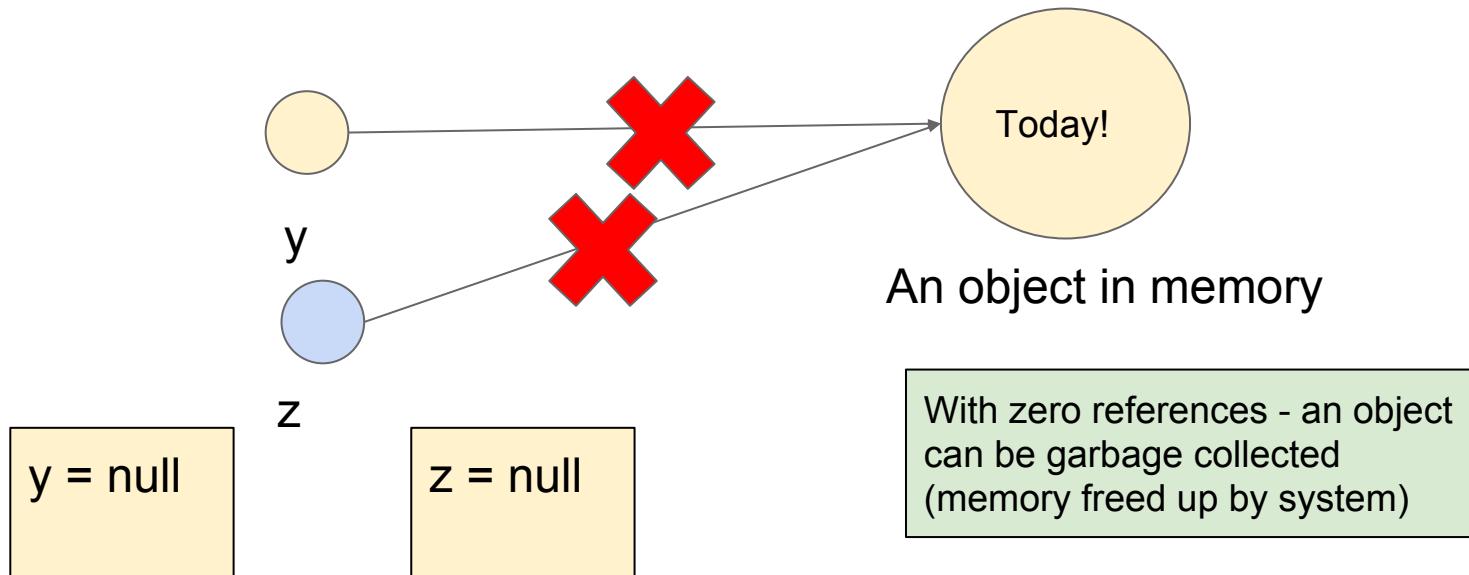
# Can use assignment operator to point to same object

- Can use `z = y` to create another reference to the same object



# Stop pointing at the object by assigning **null**

- As long as one variable points to an object it stays in memory



# The null type

- **null** - absence of object, set explicitly
  - `typeof` returns ‘object’
  - ECMAScript proposal for returning ‘null’ for `typeof` was rejected

```
26      var y;    console.log(typeof y);
27      y = null;  console.log(typeof y);
```



undefined
object

# Operators

<b>assignment operator</b>	=	let myVariable = 'Bob';
<b>addition or concatenation</b>	+	x = 6 + 9; // 15  "Hello " + "world!"; // Hello world!
<b>subtract, multiply, divide</b>	-, *, /	9 - 3; // 6 8 * 2; // 16  9 / 3; // 3
<b>remainder/Modulus</b>	%	x=12 console.log(x%3) // 0 console.log (x%5) // 2

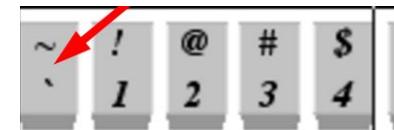
# Operators

<b>Pre Increment/Decrement</b>	<code>++x</code> <code>--z</code>	<p>shortcut for <code>x+1</code> or <code>x-1</code></p> <p>for pre do operation before assigning</p> <p><code>x = 3; y = ++x;</code></p> <p><code>console.log(x, y) // 4 4</code></p>
<b>Post Increment/Decrement</b>	<code>x++</code> <code>z--</code>	<p>for post do operation after assigning</p> <p><code>x = 3; y = x++;</code></p> <p><code>console.log(x, y) //4 3</code></p>
<b>Exponent (ES7)</b>	<code>**</code>	<code>8**2 // 64</code>

# String literals (hard-coded values) in quotes

- Literals can be represented with single quotes and double quotes

```
75  console.groupCollapsed('strings...');  
76  let dblQuoteString = "Isn't it nice that I can contain single quotes?";  
77  let singleQuoteString = 'Well, you may "think" that\'s cool...';  
78  console.log(dblQuoteString);  
79  console.log(singleQuoteString);  
80  console.groupEnd();
```



- And now, in ES6 you can use backticks `

```
89  let myString = ` Backticks can contain 'Single' and "Double" quotes  
90  | | | | | |  
91  | | | | | | and new line  
92  | | | | | | characters `;  
93  | | | | | |  
94  | | | | | | console.log(myString);
```

Backticks can contain 'Single' and "Double" quotes  
and new line  
characters

# Concatenation with Strings

- The plus symbol can be used, need to take care to add spacing between

```
94 let firstName = 'John';
95 let lastName = 'Herilla';
96 console.log('His name was ' + firstName + ' ' + lastName);
97 console.log(`His name was ${firstName} ${lastName}`);
```

- Backticks eliminate the need for the plus symbol
  - variables are included using `$ { variableName }`

# Code Runner to quickly execute JS files in VSCode

- Can run Code from within editor
  - Right-click in editor and shortcuts

The screenshot shows the Visual Studio Marketplace page for the "Code Runner" extension by "formulahendry". The extension has 2,997,069 installs and a 4.61/5 rating. It supports various languages including C, C++, Java, JavaScript, PHP, Python, Perl, Ruby, Go, Lua, Groovy, PowerShell, CMD, BASH, F#, C, C#, VBScript, TypeScript, CoffeeScript, Scala, Swift, Julia, Crystal, OCaml Script, R, AppleScript, Elixir, Visual Basic .NET, Clojure, Haxe, Objective-C, Rust, Racket, AutoHotkey, Autolt, Kotlin, Dart, Free Pascal, Haskell, Nim, D, and custom command. The page includes links for Details, Contributions, Changelog, and Dependencies.

Code Runner formulahendry.code-runner

Jun Han | 2,997,069 | ★★★★☆ | Repository | License

Run C, C++, Java, JS, PHP, Python, Perl, Ruby, Go, Lua, Groovy, PowerShell, CMD, BASH, F#, C, C#, VBScript, TypeScript, CoffeeScript, Scala, Swift, Julia, Crystal, OCaml Script, R, AppleScript, Elixir, Visual Basic .NET, Clojure, Haxe, Objective-C, Rust, Racket, AutoHotkey, Autolt, Kotlin, Dart, Free Pascal, Haskell, Nim, D, and custom command

[Reload](#) [Uninstall](#)

[Details](#) [Contributions](#) [Changelog](#) [Dependencies](#)

Code Runner

chat on gitter Visual Studio Marketplace v0.9.3 installs 2997070 rating average: 4.61/5 (85 ratings) build passing

Run code snippet or code file for multiple languages: C, C++, Java, JavaScript, PHP, Python, Perl, Perl 6, Ruby, Go, Lua, Groovy, PowerShell, BAT/CMD, BASH/SH, F# Script, F# (.NET Core), C# Script, C# (.NET Core), VBScript, TypeScript, CoffeeScript, Scala, Swift, Julia, Crystal, OCaml Script, R, AppleScript, Elixir, Visual Basic .NET, Clojure, Haxe, Objective-C, Rust, Racket, AutoHotkey, Autolt, Kotlin, Dart, Free Pascal, Haskell, Nim, D, and custom command

Run Code	Alt+Ctrl+N
Go to Definition	F12
Peek Definition	Alt+F12
Go to Type Definition	
Find All References	Shift+F12
Rename Symbol	F2
Change All Occurrences	Ctrl+F2

# Lab: JS variables and types

10 min

- Navigate to the folder \Labs\Ch06-ModernJS\1variables-types
- Open the file README.md and follow steps
  - You will be executing JS statements
  - Using Node from within VSCode integrated terminal
  - Using Code Runner

ECMAScript & JavaScript

Declaring Variables with var, let, const

JavaScript Types

## ➤ **Conditionals and Looping**

Creating and Calling Functions

Working with Arrays

Objects and Properties

Arrow Functions, this, and More ES6 Syntax

# Using if statements to evaluate expressions

- if (true) the block of code executes
  - Recall: blocks of code start and end with curly braces { }

```
1 let working = false;  
2  
3 const day = new Date().getDay();  
4 // 0 = Sunday 1 = Monday .... 6 = Saturday  
5 if (day==0) {  
6   console.log('Today is Sunday');  
7   working = false;  
8 }
```

When writing code, **be careful** it is two == for comparison.

A common bug is to use single = for assignment and always get true.

# Using if, else if and else

- Can have multiple blocks
- First to evaluate to true is executed
- the rest are skipped

```
5 | const isHoliday = isTodayAHoliday();
6 | const day = new Date().getDay();
7 | // 0 = Sunday 1 = Monday .... 6 = Saturday
8 | if (day == 0) {
9 |     console.log('Today is Sunday');
10|     working = false;
11| }
12| else if (day == 6) {
13|     console.log('Today is Saturday');
14|     working = false;
15| }
16| else if (isHoliday) {
17|     console.log('Today is a holiday');
18|     working = false;
19| }
20| else {
21|     console.log('Back to work, day is ' + day);
22|     working = true;
23| }
```

# Equality and Identity Operators

Equality Operator	<code>==</code>	check and allow conversion <code>a = '1'</code> <code>b = 1</code> <code>(a==b) → true</code>
Strict equal, checks type ( <i>preferred</i> )	<code>===</code>	<code>(a===b) → false</code>
Negation, not equal	<code>!=</code>	<code>var myVariable = 3;</code> <code>myVariable != 4;</code>
Strict not equal true if same type but not equal, or if different types	<code>!==</code>	<code>3 !== '3'</code>

# Relational Operators

Less than	<
Greater than	>
Less than or equal	<=
Greater than or equal	>=
Logical AND	&&
Logical OR	

Warning: => is not an operator but is used with Arrow Functions

# Using && to compare, both sides must be true

- Use boolean checks to execute blocks of code
  - Why is && called a shortcut expression?

```
21 //if, else if
22 if (hour > 12 && hour < 13) {
23     console.log('It is lunch time');
24 }
25 else if (hour > 8 && hour < 17) {
26     console.log('class is in session');
27 }
28 else {
29     console.log('go home - class is not in session!');
30 }
```

With AND, if left is false, right is never evaluated, not needed

# Revisiting boolean values: Truth-y and False-y

- Expressions can be used in a boolean context to return true or false
- Most values are true - truthy
- The following are always false - falsy
  - The keyword `false`
  - The number `0`
  - The empty string `""` or `' '`
  - The keywords `null` and `undefined`
  - The special number Not-a-Number `NaN`

# Comparing Falsy Values

- Falsy values have unusual comparison rules
- Falsy values false, 0 and "" are equivalent  $0 == \text{false}$
- null, undefined, and NaN are strange

```
const lie = false;  
console.log('lie == 0 ', (lie == 0));
```

```
lie == 0 true  
lie == "" true  
lie == '' true  
null == undefined true  
null === undefined false  
null == null true  
NaN == NaN false
```

# For Loops: be careful with var

```
10...  
9...  
8...  
7...  
6...  
5...  
4...  
3...  
2...  
1...
```

```
5  for (var countdown = 10; countdown > 0; countdown--) {  
6    |   console.log(` ${countdown} ... `);  
7  }  
8  
9  console.log(`What is countdown now? ${countdown}`)
```

```
What is countdown now? 0
```



- When using **var**, it is not confined to the block scope
  - ... like most languages

# For Loops help with repetitive tasks

Initialize a counter

Set a conditional Check

Change Counter

JS loops.js 

```
1  for (let counter = 0; counter < 3; counter++) {  
2      console.log('BeetleJuice');  
3  }
```

BeetleJuice  
BeetleJuice  
BeetleJuice



# Lab: JS conditionals and loops

10 min

- Navigate to the folder \Labs\Ch06-ModernJS\2conditionals
- Open the file README.md and follow steps

# **Modern JavaScript**

ECMAScript & JavaScript

Declaring Variables with var, let, const

JavaScript Types

Conditionals and Looping

## **Creating and Calling Functions**

Working with Arrays

Objects and Properties

Arrow Functions, this, and More ES6 Syntax

# Defining and calling functions

- Function: a block of code designed to perform a particular task

```
function printStars() {  
    console.log('*****');  
}
```

- A JavaScript function is executed when "something" invokes it (calls it)
  - Call a function by using parenthesis

```
printStars();
```

# Functions can take parameters

- Inside of the function the parameters behave as local variables

```
25  function printFullName(localFirstName, localLastName) {  
26      localFirstName = "Dr. " + localFirstName;  
27      console.log(localFirstName + ' ' + localLastName);  
28  }  
29  
30  firstName = 'Bob';  
31  lastName = 'Williams';  
32  printFullName(firstName, lastName);  
33  console.log('after function call, name is ' + firstName);
```

Dr. Bob Williams

after function call, name is Bob

# Functions can return a value

- The returned value can be stored or used in an expression

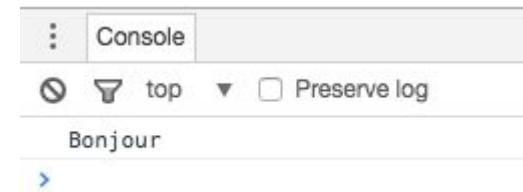
```
36  function returnFullName(firstName, lastName) {  
37      return (firstName + ' ' + lastName);  
38  }  
39  
40  const fullName = returnFullName('Jackie', 'Chan');  
41  
42  console.log(fullName);
```

# Function scope and conflicts

- If two scripts are included function names may conflict
- The last one loaded replaces the others

```
5      <script>
6          function getText() {    21
7              return "Hello" ;    22
8          }                      23
9      </script>
10     <script>
11         function getText() {
12             return "Bonjour" ;
13         }
14     </script>
```

```
<script>
    console.log( getText() ) ;
</script>
```



# Using default function parameters

- Function parameters can be initialized if
  - no value or `undefined` is passed in

```
37  function multiply1( a, b ) {  
38    return a * b;  
39  }  
  
40  function multiply2( a, b ) {  
41    b = (typeof b !== 'undefined') ? b : 2 ;  
42    return a * b;  
43  }  
  
44  function multiply3( a, b = 2) {  
45    return a * b;  
46  }
```

What is the expected return value of the following:

`multiply1( 5, 5 );`

25

`multiply1( 5 );`

NaN

`multiply2( 5, 5 );`

25

`multiply2( 5 );`

10

`multiply3( 5, 5 );`

25

`multiply3( 5 );`

10

# Lab: Practice with Functions

10 min

- Navigate to the folder \Labs\Ch06-ModernJS\3functions
- Open the file README.md and follow steps

# **Modern JavaScript**

ECMAScript & JavaScript

Declaring Variables with var, let, const

JavaScript Types

Conditionals and Looping

Creating and Calling Functions



**Working with Arrays**

Objects and Properties

Arrow Functions, this, and More ES6 Syntax

# Creating and Accessing Arrays

- Arrays can be created with square brackets or using `new Array()`

```
let simpleNameArray = ['Adam', 'Judy', 'Cody'];
let ages = new Array(45, 41, 1);
```

- Elements of the array can be accessed by the index: starting at 0

```
console.log(simpleNameArray[1] + ' is ' +
| | | | | |
ages[1] + ' years old');
```



---

Judy is 41 years old

# Finding length of array

- Arrays have a length to show the number of elements

```
let numbers = new Array(1, 2, 3, 4, 5);
console.log('Numbers length ' + numbers.length);
```



Numbers length 5

# Basic for loop with Array

```
let simpleNameArray = ['Adam', 'Judy', 'Cody'];
let ages = new Array(45, 41, 1);
```

- Use length to loop over array

```
let output = '';
for (let index = 0; index < simpleNameArray.length; index++) {
    let element = simpleNameArray[index];
    output+= element + ' ';
}
console.log('Loop output: ' + output);
```



Loop output: Adam Judy Cody

# Array.prototype

- There are many helpful functions available via Array.prototype
  - Once you have an array - all of these functions are available
  - [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/prototype](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/prototype)
- Understanding how to use these is extremely helpful
  - for front or back-end JavaScript
- Most of what we do as developers is **send** and **receive** and **process** data

## Add and remove items to and from the end of an array

```
const bands = [];

bands.push('The Beatles');
bands.push('Aerosmith');
bands.push('The Temptations');

console.log(bands); // ["The Beatles", "Aerosmith", "The Temptations"]

const aBand = bands.pop()

console.log(aBand); // "The Temptations"
console.log(bands); // ["The Beatles", "Aerosmith"]
```

# Sorting an Array

- The sort method will order of elements in the array, in place
  - by their string value Unicode code point order

```
56 //Sorting
57 console.groupCollapsed('Sorting...');
58
59 const letters = ['z','q','e','a','q','j','i']; //
60 console.log(letters.sort());
61 // ["a", "e", "i", "j", "q", "q", "z"]
```

```
var scores = [1, 10, 21, 2];
console.log(scores.sort());
```



[ 1, 10, 2, 21 ]

# Higher-order functions in Array.prototype

- Some Array.prototype functions can receive functions as parameters
  - because of this, they are called **higher-order** functions
  - sort(), find(), filter(), etc.
- These iterate over elements, passing one or more values into the function
  - and using the return in a formula

```
scores.sort(function(a,b) {  
    return a-b  
})
```

If the parameter is not a named function it is called **anonymous**

# The way sort() works

- When sort() is called given a function
  - two values are passed in at a time
- if the return value is **positive**, then **a** should come before **b**
- if the return value is **0**, then **a == b**
- if the return value is **negative**, then **b** should come before **a**

```
let scores = [1, 10, 21, 2];
scores.sort(function(a, b) {
  console.log(a, b);
  return a - b;
});
console.log(scores);
```



```
1 10
10 21
21 2
10 2
1 2
[ 1, 2, 10, 21 ]
```

```
scores.sort(function(a,b) {
  return a-b
})
```

# Find an element in an array

- The `find()` function takes a function as a parameter
  - `find` will loop through the array passing each value to the function
  - an expression returns a boolean value
  - the first true causes a value will be returned to `needle`

```
const haystack = ['Apple', 'banana', 'Berry', 'Cherry'];
const needle = haystack.find(function (currentValue) {
  return currentValue.toUpperCase().startsWith('B'));
});

console.log('Needle is: ' + needle);
```



Needle is: Banana

If no element causes the expression to be true, the value of `undefined` is returned

# Filter an array

- Select all of the items in an array that match a predicate function with `filter()`:

```
const numbers = [1,2,3,4,5,6];

const evens = numbers.filter(function(currentValue) {
    return currentValue % 2 == 0;
});

console.log('Evens are: ' + evens.join(', '));
// Produces: Evens are: 2, 4, 6
```

# Convert an array to another

- The `map()` function uses a *callback* function for each item in the array to convert it to another array

```
const values = [1, 2, 3, 4, 5];

const items = values.map(function(value) {
    return "Item: " + value;
});

console.log(items);
// Produces ["Item: 1", "Item: 2", "Item: 3", "Item: 4", "Item: 5"]
```

# Lab: Practice with arrays

10 min

- Navigate to the folder \Labs\Ch06-ModernJS\4arrays
- Open the file README.md and follow steps

ECMAScript & JavaScript

Declaring Variables with var, let, const

JavaScript Types

Conditionals and Looping

Creating and Calling Functions

Working with Arrays

## ► **Objects and Properties**

Arrow Functions, this, and More ES6 Syntax

# Objects

- You can group together multiple values in an Object
- Object literals contain key-value pairs
- Advantages over arrays to keep related information together
- Very helpful to retrieve properties of similar Objects
- Like "dictionaries" or "maps" in other languages
  - an unordered collection of key-value pairs

# Create Object Literals {} or use new Object()

- Here an object instance is created as an Object Literal - using curly braces

```
1 const band1 = {  
2   name : "Pink Floyd",  
3   city : "London" ,  
4   country : "England",  
5   yearFormed : 1965,  
6   genres : ["Progressive rock", "psychedelic rock", "art rock"]  
7 }
```

- This is the equivalent of:

```
11 const band2 = new Object();  
12 band2.name = "Man Man";  
13 band2.country = "USA";  
14 band2.city = "Philadelphia";  
15 band2.yearFormed = 2003;  
16 band2.genres = ["Experimental rock"];
```

# Referencing Properties of Objects

- Properties can be set or retrieved using dot notation or bracket notation

```
21 //access properties using dot notation or []
22 band1.genres = new Array("Progressive rock2", "psychedelic rock2", "art rock2");
23 console.log("country 1 = " + band1.country);
24 console.log("country 2 = " + band2["country"]);
```

# Objects in Arrays

- Arrays can contain object literals:

```
let bands = [
    { name: "Morphine",
        country: "USA",
        city: "Boston",
        genres: ["Alternative Rock", "Experimental Rock", "Jazz Rock"]
    },
    { name: "Vitamin String Quartet",
        country: "USA",
        city: "Los Angeles",
        genres: ["Rock", "Pop", "Instrumental"]
    }
];
```

# For loops commonly used with arrays of objects

- Use length to set the condition check

```
19  const pets = [
20    {name: 'Birdy', type: 'cat'},
21    {name: 'Roxy', type: 'dog'}
22  ]
23
24  for (let index = 0; index < pets.length; index++) {
25    console.log(index);
26    console.log(pets[index].name + " is a " + pets[index].type);
27  }
```

# Constructor Functions

- Declare your own constructor functions to be used with keyword **new**
  - Follow conventions and capitalize!

```
1  function Square(width) {
2      this.width = width;
3      this.area = function() {
4          return this.width * this.width;
5      };
6  }
7
8  const square = new Square(4);
9  console.log(square.area());
```

# Constructor Functions with prototype

- Instead of declaring the function area inside of Square use prototype

```
11  function BetterSquare(width) {  
12    this.width = width;  
13  }  
14  
15  BetterSquare.prototype.area = function() {  
16    return this.width * this.width;  
17  };  
18  
19  const newSquare = new BetterSquare(5);  
20  console.log(newSquare.area());
```

What is the advantage?



Even if 1,000 squares area created,  
only 1 area() in memory

# ES6 Classes

- Instead of needing to use prototypes, ES6 offers a class syntax
  - similar to other OO languages: Java, C#
- This is “syntactic sugar”
  - A prototype is created for area()

```
1  class Square {  
2      constructor(width) {  
3          this.width = width;  
4      }  
5  
6      area() {  
7          return this.width * this.width;  
8      }  
9  }  
10  
11 const newSquare = new Square(3);  
12 console.log(newSquare.area());
```

# Lab: Practice with objects

10 min

- Navigate to the folder \Labs\Ch06-ModernJS\5objects
- Open the file README.md and follow steps

# **Modern JavaScript**

ECMAScript & JavaScript

Declaring Variables with var, let, const

JavaScript Types

Conditionals and Looping

Creating and Calling Functions

Working with Arrays

Objects and Properties



**Arrow Functions, this, and More ES6 Syntax**

# Arrow functions can be used for anonymous functions

- Arrow functions are shorter and more concise

```
JS arrow-function.js ✘  
1 const values = [1,2,3,4,5];  
2  
3 const anonymousItems = values.map(function(value){  
4     return "Item: " + value;  
5});  
6  
7 const arrowItems = values.map((value)=>{  
8     return "Item: " + value;  
9});  
10  
11 //if only 1 param, dont need parenthesis  
12 //if only one statement, dont need curly braces  
13 //if only a return statement, do not need return keyword  
14 const arrowItemsShorter = values.map(value => "Item: " + value);
```



# Use of **this** in callbacks

- The keyword **this** gets bound to a context

JS using-this-keyword.js ✘

```
1  function Person (firstName, lastName) {  
2      this.firstName = firstName;  
3      this.delayFirstnamePrint = function() {  
4          setTimeout (function() {  
5              console.log(`#1: ${this.firstName}`)  
6          }, 2000);  
7      }  
8  }  
9  
10 let p = new Person('Jane', 'Doe');  
11 p.delayFirstnamePrint()
```

On lines 2 and 3 what does **this** refer to?

**this** binds to an instance of Person

What is the output?

#1: undefined

Why?

**this** binds to local callback function, not Person instance

# One Workaround: Capturing `this`

- Can store the reference to outer `this` in a variable and use it in callback

```
15  function PersonSelf (firstName, lastName) {  
16      let self = this;  
17      this.firstName = firstName;  
18      this.delayFirstnamePrint = function() {  
19          setTimeout (function() {  
20              console.log(`## self ##: ${self.firstName}`)  
21          }, 2000);  
22      }  
23  }  
24  
25  let pself = new PersonSelf('Eckhart','Tolle');  
26  pself.delayFirstnamePrint();
```

# Arrow functions of this in callbacks

- Arrow functions do not have their own this
  - They have lexical scope, this refers to the person instance

```
31  function PersonArrow (firstName, lastName) {  
32      this.firstName = firstName;  
33      this.delayFirstnamePrint = function() {  
34          setTimeout (()=>{  
35              console.log(`## ArrowFunction ##: ${this.firstName}`)  
36          }, 2000);  
37      }  
38  }  
39  
40  let pArrow = new PersonArrow('Robin','Hood');  
41  pArrow.delayFirstnamePrint();
```

# REVIEW: Basic for loop with Array

- Use length to loop over array

```
let output = '';
for (let index = 0; index < simpleNameArray.length; index++) {
    let element = simpleNameArray[index];
    output+= element + ' ';
}
console.log('Loop output: ' + output);
```

# Using the for...of Loop

- Simplify looping through an array using for...of

You can use `const` instead of `let` if you don't reassign the `value` variable inside the block

```
let friends = ['Alice', 'Bob', 'Eve'];
```

```
for (const friend of friends) {  
    console.log(friend);  
}
```

Alice  
Bob  
Eve



# Using the for...in Loop

- Can get the property key, with for...in and use it

```
72 let configs = {  
73   user: "root",  
74   port: 1111,  
75   hostname: "karmoxie.com"  
76 }  
77  
78 for (const property in configs) {  
79   console.log(` ${property} : ${configs[property]}`)  
80 }
```



user : root  
port : 1111  
hostname : karmoxie.com

# Destructuring arrays

- Array elements can quickly be assigned using this syntax

```
1  let [first, last] = ['Les','Claypool'];
2
3  //instead of
4  // data = ['Les','Claypool'];
5  // let first = data[0]
6  // let last = data[1]
```

- If more variables on left than elements, extras are assigned “**undefined**”

# Chapter Summary

In this chapter we have looked at:

- Review of JavaScript fundamentals
- New syntax in EcmaScript 6
- Objects and properties
- JavaScript arrays and functions
- Arrow functions and this

# Modern Web Development

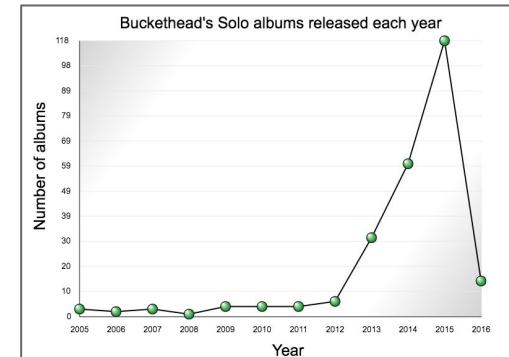


## Chapter 7

### Client-Side Web APIs

A screenshot of the Chrome DevTools Application panel. On the left, there's a sidebar with options like Manifest, Service Workers, and Clear storage. Under Storage, Local Storage and Session Storage are expanded, both showing a single item with the key 'file://'. The main area shows a table with two rows: 'band' with value 'Morphe' and 'genre' with value 'Rock'.

Key	Value
band	Morphe
genre	Rock



# Chapter Objectives

In this chapter we will look at:

- Strategies for adding JS to web pages
- Drawing with the canvas
- Storing data locally with HTML5 Web Storage
- Using Geolocation to get position
- Using AJAX to communicate with the server
- What is Fetch?
- Introducing promises



## Client-Side (Browser) JavaScript

Transpiling from ES6 to ES5

Canvas API

Web Storage

Geolocation

JSON and AJAX with XHR

Chapter Summary

Bonus: Overview of Promises & Fetch

# Add JavaScript to the page, between <script> tags

- Statements will be executed in the order they are encountered
  - Code in functions are not executed

```
<!-- Prior to HTML5, needed to specify type attribute. -->
<script type="text/javascript">
    alert('I'm an annoying little pop-up!');
</script>

<!-- No longer need type. -->
<script>
    console.log('I am much nicer to see what is going on!');
</script>
```

# Including JavaScript as external files

- JavaScript is often put in separate files
  - enables code sharing and caching
- Uses **src** tag to specify the external file, relative to current file
  - nothing between the tags

```
84 <script src="../../scripts/header.js"></script>
85 <script src="../../scripts/consoleDirections.js"></script>
```

# Conflicting Scripts

- If >1 script is included
  - beware of conflicts
- Order matters!

var day = "Wednesday"

var day = "Friday"

What is day?

Friday - it overwrites first

```
conflicting-scripts.html ✘
```

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Variables</title>
5  <script src="variables-mine.js"></script>
6  <script src="variables-yours.js"></script>
7  </head>
8  <body>
9      <script>
10         alert(myFavoriteMusician);
11     </script>
12  </body>
13  </html>
```

# Strategies for <script> placement

- The location of <script> tags can have significant performance issues
  - Web pages often have a lot of JavaScript and need to load fast
- Script tags can go in the HTML <head>
  - Browsers will wait for the JavaScript to load before rendering HTML
  - If put into a function, can be run via browser load() event
- Script tags can go just before the </body> tag
  - Document is unresponsive until JavaScript loaded

# Loading scripts with ‘defer’

- Using `defer`, HTML parsing continues while the scripts are being loaded
  - It delays parsing JavaScript files until all HTML has been parsed
- `defer` can only be used with external files and the `src` attribute
- About 80% of browsers support `defer` <http://caniuse.com/#feat=script-defer>

```
<script src="someScript.js" defer="defer"></script>
```

# Browser events can be used to call your functions

- After the browser loads the body content, an **onload** event is raised
  - Your code can be called then by using an **onload** attribute

```
22 |     <head>
23 |         <script>
24 |             function bodyOnloadFunction() {
25 |                 document.getElementById('targetedH2').innerHTML = "New text for h2";
26 |             }
27 |         </script>
28 |     </head>
29 |     <body onload="bodyOnloadFunction()">
30 |         <h2 id="targetedH2">
31 |             In the h2
32 |         </h2>
33 |         ...
34 |     </body>
```

# Interacting with a form element can cause **onchange**

- This example demonstrates:

- reacting to an event
- calling a function
- updating contents of an element

```
23    <label for="fname">Type in capital letters, then click away</label>
24    <input type="text" id="fname" onchange="lowerCase()">
25    <script>
26        function lowerCase() {
27            var x = document.getElementById("fname");
28            x.value = x.value.toLowerCase();
29        }
30    </script>
```

# Discuss & Explore: Commonly Used Browser events

- There are many types of events we can work with as developers
- <https://developer.mozilla.org/en-US/docs/Web/Events>

# Using the document object

- We have seen examples of using `document.getElementById("someValue")`
- Other functions exist such as `document.write("some content")`
  - This will write to the screen

# Using document.write to write to page

```
28 <script>
29     document.write("What is 2+4? :");
30     document.write(2 + 4);
31     document.write('<br />And 3 + 1?<br /> ');
32     document.write(3+1);
33 </script>
```

What is 2+4? :6  
And 3 + 1?  
4

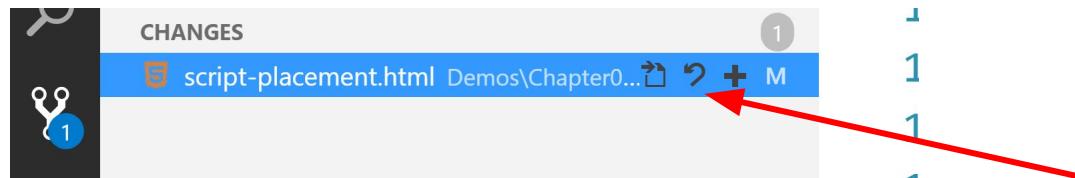
# DO NOW: Script Placement & Git Discard changes

5 min

1. In VSCode, navigate to /Demos/Chapter07-ClientAPIs/includingJS/
2. Open the file `script-placement.html` in the editor
3. Use Alt-B to open in the Chrome browser
4. Find the `bodyOnloadFunction` and comment out the lines as follows:

```
23   <script>
24     // function bodyOnloadFunction() {
25       document.getElementById('targetedH2').innerHTML = "New text for h2";
26     //}
27   </script>
```

5. Refresh the browser and check the console. Why is this happening?
6. Use the Git control to undo your changes to the demo file



# Lab: Placing JS in the browser

5 min

- Navigate to the folder \Labs\Ch07-ClientAPIs\1scripts
- Open the file README.md and follow steps to get started with JS in browser

# Client-Side Web APIs

Client-Side (Browser) JavaScript



**Transpiling from ES6 to ES5**

Canvas API

Web Storage

Geolocation

JSON and AJAX with XHR

Chapter Summary

Bonus: Overview of Promises & Fetch

# **ES6 only partially supported by browsers**

- Chrome has the most support
- Can still use ES6 during development, and transpile for use on web
- All of the Major Web Frameworks use ES6
  - Angular, React, Ember, Vue etc.
- Use transpilers to convert from ES6 to ES5

# Babel is the most popular tool

- Transpiles from ES6 to ES5
- <https://babeljs.io/>
- Use as part of development environment
- Use the [babel cli](#) to transpile from the command line
  - You could install globally, but best practice is to do per project



# Including babel CLI in a project

- Add babel by using: **npm install --save-dev babel-cli babel-preset-env**
  - Note that spaces allow you to add more than one package at a time
  - This modifies package.json and installs the packages

```
10  "devDependencies": {  
11    "babel-cli": "^6.26.0",  
12    "babel-preset-env": "^1.6.1"  
13  }
```

- When others run **npm install**, the dependencies will be installed
  - babel-preset-env allows you to indicate what to transpile to...

# Using babel-preset-env

- Usage of babel has changed over time
  - Your needs may vary by project
- The [babel-preset-env](#) dependency will be used to work with a **.babelrc** file
- Add a **.babelrc** file to your project
  - It target browsers using values
  - The values are from a library called [browserslist](#)

```
1  {
2    "presets": [
3      ["env", {
4        "targets": {
5          "browsers": [
6            "Chrome >= 52",
7            "FireFox >= 44",
8            "Safari >= 7",
9            "Explorer 11",
10           "last 4 Edge versions"
11         ],
12       },
13       "useBuiltIns": true
14     }]
15   ]
16 }
```

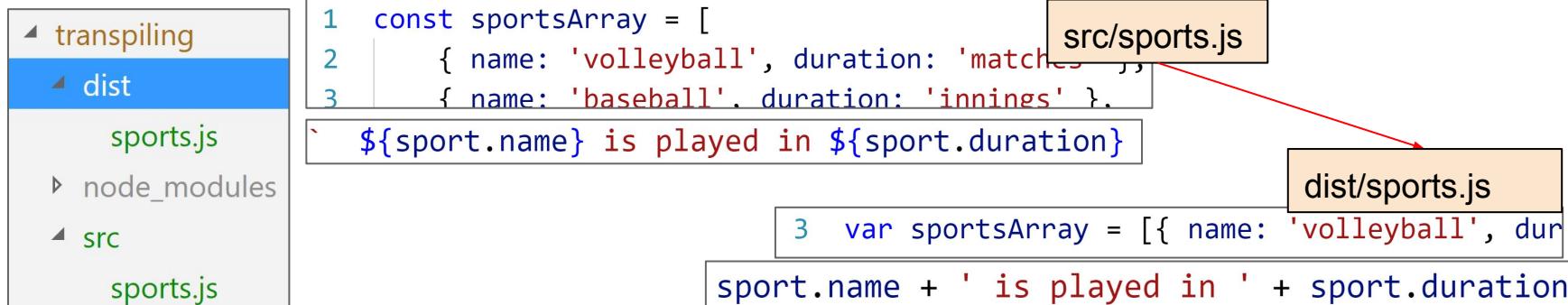
**.babelrc file**

# Set up a build task in package.json

- By adding a build script, developers can automate the transpiling

```
"scripts": {  
  "build": "babel src --watch --presets babel-preset-env --out-dir dist"  
},
```

- Developers write code using ES6, and it is transpiled to ES5



## DO NOW: Try out babel

10 min

1. In VSCode, navigate to /Demos/Chapter07-ClientAPIs/transpiling
2. View index.html - what script is being included?
3. View the package.json - what are the dependencies?
4. Use **npm install** to get the dependencies
5. Notice how the build script uses the babel cli to create a dist directory from the src directory files.
6. Execute the build script using **npm run build**. It will keep running, transpiling changes because of the watch flag.
7. Load the index.html in both Chrome and IE

# Lab: Transpiling ES6

30 min

- Open /Labs/Ch07-ClientAPIs/2transpiling/README.md
- Follow the directions there to practice using Babel for transpiling ES6

My hobbies

- volleyball 25
- cooking 15
- swimming 11

Hobby	Years of Hobby enjoyment
volleyball	25
cooking	15
swimming	11

# Client-Side Web APIs

Client-Side (Browser) JavaScript

Transpiling from ES6 to ES5



## **Canvas API**

Web Storage

Geolocation

JSON and AJAX with XHR

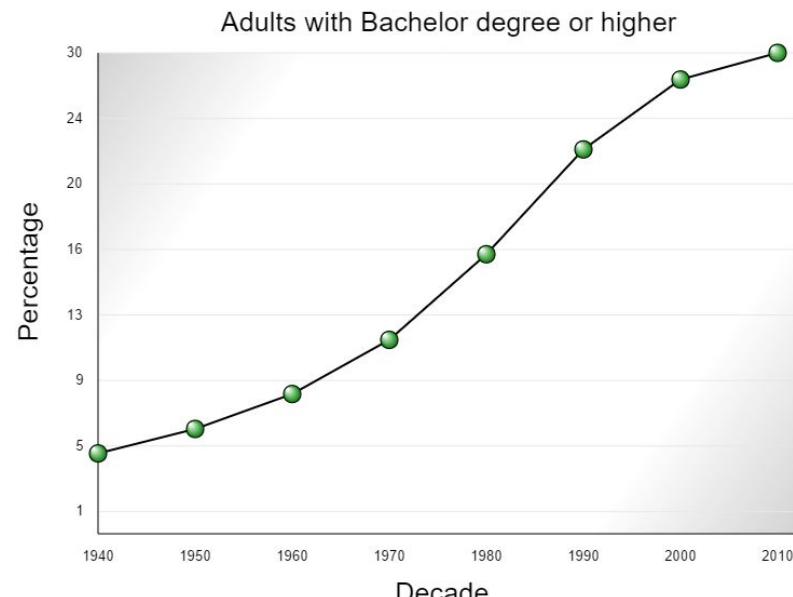
Chapter Summary

Bonus: Overview of Promises & Fetch

# What is Canvas?

- The <canvas> tag is a rectangular area
  - Used for drawing graphs, photo compositions and animations
  - Specify width, height and an id
  - Default is 300 px × 150 px

```
10 <canvas id="myCanvas" width="800" height="600">
11   This text only appears if canvas is not supported
12 </canvas>
```



# Use JavaScript to work with the Canvas API

- Get the **context**, then add elements to that context using JavaScript

```
12 <script>  
13     var canvas = document.getElementById("myCanvas");  
14     if (canvas.getContext) {  
15         var context = canvas.getContext("2d");  
16         context.fillStyle = "rgb(200,0,0)";  
17         context.fillRect(10, 10, 100, 100);
```



Can check for context and provide alternatives programmatically

# Fallback Content

- Put alternate content inside the <canvas> element
- Browsers not supporting <canvas>
  - ignore the container and render the fallback content inside it
- Browsers that do support <canvas>
  - ignore the fallback content and just render the canvas normally

# Rendering Context

- The 2D context is the most supported
- ## Browser compatibility

Desktop	Mobile	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Feature						
Basic support (2d context)		4	3.6 (1.9.2)	9	9	3.1
webgl context		9[1] 33	3.6 (1.9.2)[1] 24 (24)	11.0[2]	9.0[3]	5.1[2]
webgl2 context ⚡		No support	25 (25)[4]	No support	No support	No support
2d alpha context attribute		32	30 (30)	No support	(Yes)	No support
failIfMajorPerformanceCaveat attribute		?	41 (41)	(Yes)	?	?
bitmaprenderer context		No support	46 (46)	No support	No support	No support

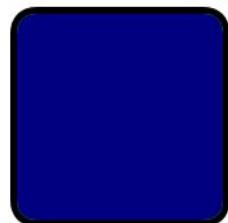
# What does the 2D Canvas API Include?

- The Canvas's 2D context is a grid
  - Top-left corner is the default origin
- [Canvas 2D API](#) includes
  - Drawing lines and shapes
  - Using gradients and patterns
  - Pixel processing

# Drawing Rectangles

- Only rectangles are supported directly through functions
- Other shapes you create the lines/paths or arcs

```
11 //Render a rectangle
12 ctx.strokeStyle = 'Red';
13 ctx.lineWidth = 1;
14 ctx.rect(0, 0, 150, 100);
15 ctx.stroke();
16
17
18 ctx.fillStyle = 'Green';
19 ctx.fillRect(200, 0, 200, 100);
20
21 //Rounded rectangle
22 ctx.strokeStyle = 'Red';
23 roundedRect(ctx, 450, 5, 150, 150, 15);
```



# Drawing Paths

- Paths can be straight or curved
- Steps to create
  - beginPath() - creates a new path. if already called resets.
    - call different path methods
    - call different path methods
    - call different path methods
  - closePath() - connect end position to start, optional
- Finishing drawing - calls closePath()
  - stroke() draw the outline
  - fill() fill path's content area

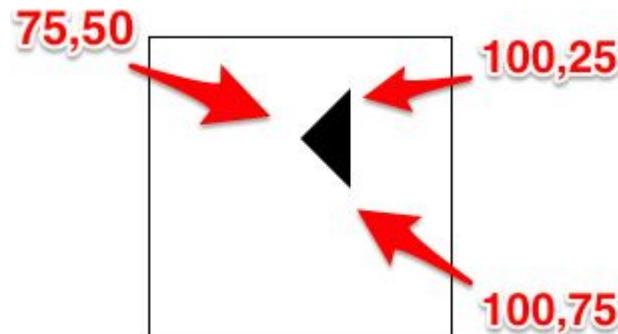
# The `moveTo()` function

- Use when canvas is initialized or `beginPath()` is called
- Use to create unconnected paths    `moveTo(x, y)`
- Moves the pen to the coordinates specified by x and y

# Drawing a Triangle

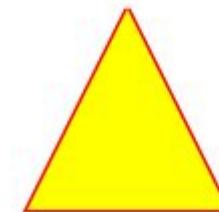
- Here, using `moveTo` to change starting location

```
ctx = document.getElementById("triangle").getContext("2d");
ctx.beginPath();
ctx.moveTo(75,50);
ctx.lineTo(100,75);
ctx.lineTo(100,25);
ctx.fill();
```



# Adding Color, stroke or fill

```
26 //Drawing lines  
27 ctx.beginPath();  
28 ctx.moveTo(100, 400);  
29 ctx.lineTo(50, 500);  
30 ctx.lineTo(150, 500);  
31 ctx.lineTo(100, 400);  
32 ctx.strokeStyle = 'Red';  
33 ctx.lineWidth = 3;  
34 ctx.stroke();  
35 ctx.fillStyle = 'Yellow';  
36 ctx.fill();
```



# Drawing a Circle

- Use context.arc(x, y, radius, startAngle, endAngle, counterclockwise)

```
15  //Render a circle  
16  ctx.arc(100, 200, 50, 0, 2 * Math.PI, false);  
17  ctx.fillStyle = 'Navy';  
18  ctx.fill();  
19  
20  //Render an arc  
21  ctx.beginPath();  
22  ctx.arc(100, 300, 50, 0, Math.PI, false);  
23  ctx.fillStyle = 'Navy';  
24  ctx.fill();
```



Math.PI



0

# Saving an Image

- To save the canvas contents to an image

```
canvas.toDataURL("image/png")
```

- You can forward browser to the image using `window.location`

```
window.location = canvas.toDataURL("image/png");
```

# Lab: Working with canvas (1/2)

20 min

## 1. View canvas usage on website

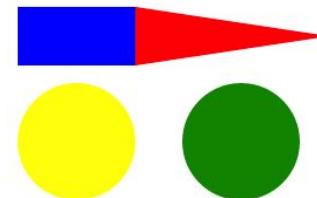
- Open <http://labs.hyperandroid.com/html5-tiler-3d>
- Use Chrome tools to see usage of canvas tag

## 2. Examine a canvas-based game

- Open /Demos/Chapter07-ClientAPIs/canvas-game/
  - View index.html - where is the javascript code located?
  - Examine the javascript code. How is canvas used?

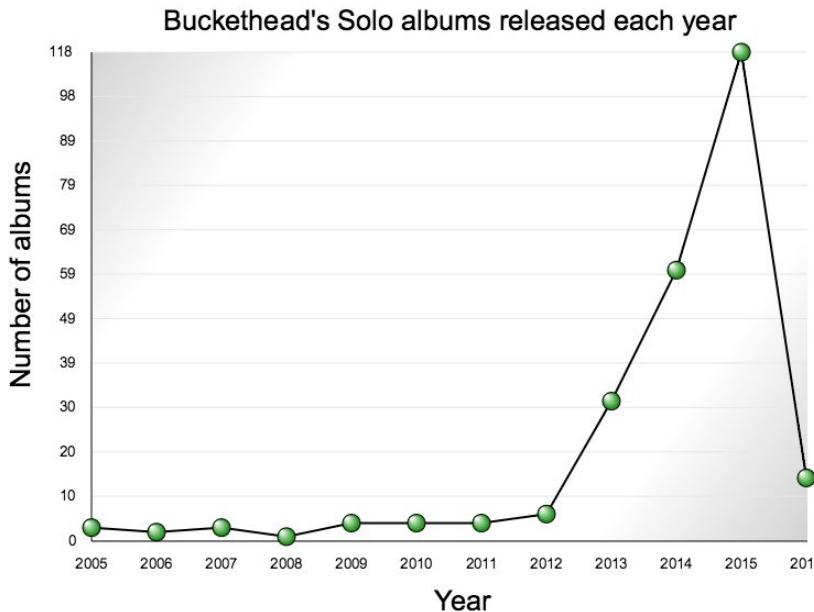
## 3. Use the Canvas API to create something similar to the following:

- Open /Labs/Ch07-ClientAPIs/3CanvasShapes/README.md
- Put your JS code in the script tags



## BONUS Exercise: Working with canvas (2/2)

4. BONUS: Open /Labs/Ch07-ClientAPIs/4CanvasChart/README.md
  - Follow the directions there to create this chart



# Client-Side Web APIs

Client-Side (Browser) JavaScript

Transpiling from ES6 to ES5

Canvas API



**Web Storage**

Geolocation

JSON and AJAX with XHR

Chapter Summary

Bonus: Overview of Promises & Fetch

# Web Storage API

- Web Storage is a simple key-value(**string**) storage for the browser
- **sessionStorage** is available only to current window until it closes
- **localStorage** is based on the domain and spans all open windows

## City & Color Preferences

Favorite City:

Favorite Color:

Preferences Saved!

The screenshot shows the IE8+ developer tools Resources panel. A red arrow points to the 'LocalStorage' section, and another red box highlights the 'city' key-value pair: 'Key' is 'city' and 'Value' is 'Pittsburgh'. The table below lists other items in the Local Storage:

	Key	Value
Frames	city	Pittsburgh
Web SQL	color	Blue
IndexedDB	email	blah@KJH
LocalStorage	fname	AEDD
file:///	key	1
Session Storage	key	1
file:///	key	1

IE8+

# Web Storage API Methods

- sessionStorage and localStorage both have the same methods

```
setItem(key, value)  
getItem(key)  
removeItem(key)  
clear()
```

- Before using, can check in JS to see if supported by the browser

```
16  function hasLocalStorage() {  
17    |   return ('localStorage' in window &&  
18    |   |   |   |   window['localStorage'] != null);  
19  }
```

# The `setItem()` Methods

- The `setItem` method takes a key and a value
  - strings
- Quota may become exceeded, can add in a check

```
30  function submitClick() {
31    if (hasLocalStorage()) {
32      try {
33        localStorage.setItem('city', cityTextBox.value);
34        localStorage.setItem('color', colorsSelect.value);
35        document.getElementById('message').innerHTML =
36          '<p>Preferences Saved!</p>';
37      } catch (e) {
38        //can assume storage error
39        alert('Storage quota exceeded');
40      }
41    }
42    else {
43      alert('No local storage support');
44    }
45 }
```

# The getItem () Method

- The getItem method takes a key and returns the stored value

```
sessionStorage.getItem(key)
```

```
localStorage.getItem(key)
```

- Example

```
21  function loadSettings() {  
22      if (hasLocalStorage()) {  
23          var city = localStorage.getItem('city');  
24          var color = localStorage.getItem('color');  
25          cityTextBox.value = city;  
26          colorsSelect.value = color;  
27      }  
28  }
```

# Exercise: Use Web Storage

15 min

- Open /Labs/Ch07-ClientAPIs/5WebStorage/README.md
- Follow the directions there to practice using local storage

# Client-Side Web APIs

Client-Side (Browser) JavaScript

Transpiling from ES6 to ES5

Canvas API

Web Storage

**Geolocation**

JSON and AJAX with XHR

Chapter Summary

Bonus: Overview of Promises & Fetch



# Geolocation is part of the WebAPI

- Geolocation is part of the WebAPI

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

- From navigator.geolocation, can get current position
  - pass a function to be executed if things are successful
  - pass a function if things do not go well

```
navigator.geolocation.getCurrentPosition(success, error);
```

# Position object contains coordinates

- The success function receives a position object
  - This can be passed off to other tools, such as Google Maps

```
42  function success(position) {  
43      var latitude = position.coords.latitude;  
44      var longitude = position.coords.longitude;  
45  
46      output.innerHTML = '<p>Latitude is ' + latitude +  
47          '° <br>Longitude is ' + longitude + '°</p>';  
48  
49      var img = new Image();  
50      img.src = "https://maps.googleapis.com/maps/api/staticmap?center=" +  
51          latitude + "," + longitude + "&zoom=13&size=300x300&sensor=false";  
52  
53      output.appendChild(img);  
54  }
```

# Demo: Geolocation

10 min

- Navigate to the \Demos\Chapter07-ClientAPIs\geolocation folder
- Open find-me.html in the editor and browser
- Review the code
  - How is it using Modernizr?
  - How is it adding a map to the page?

# Client-Side Web APIs

Client-Side (Browser) JavaScript

Transpiling from ES6 to ES5

Canvas API

Web Storage

Geolocation



**JSON and AJAX with XHR**

Chapter Summary

Bonus: Overview of Promises & Fetch

# What is JSON? (JavaScript Object Notation)

- JSON data is written as name/value pairs
- A name/value pair consists of:
  - field name in **double** quotes, followed by a colon, followed by a value
- Example

```
"city": "Boston"
```

JSON names **require** double quotes

JavaScript names do not

# Sources for JSON

- A common use is to read data from a web server as JSON
- Static text files (can have .json extensions)
- In JS files: Be careful editing! Line breaks such as invisible \n is bad!

```
var bandJSONStringData = '{"name" : "Pink Floyd", ' +  
    '"city" : "London", ' +  
    '"country" : "England", ' +  
    '"yearFormed" : "1965"}';  
  
var bandObj = JSON.parse(bandJSONStringData);
```

- Server calls requesting a JSON response

# JSON Values Can Be:

- A number (integer or decimal)
- A string (in double quotes)
- A Boolean (true or false)
- An array (in square brackets)
- An object (in curly braces)
- null

```
1 | { "bands": [
2 |   {
3 |     "name": "Pink Floyd",
4 |     "city": "London",
5 |     "country": "England",
6 |     "yearFormed": 1965,
7 |     "touring": false,
8 |     "genres": [ "Progressive rock", "psychedelic rock", "art rock" ]
9 |   },
10 |   {
11 |     "name": "Morphine",
12 |     "country": "USA",
13 |     "city": "Boston",
14 |     "yearFormed": null,
15 |     "touring": false,
16 |     "genres": [ "Alternative rock", "experimental rock", "jazz rock" ]
17 |   },
18 |   {
19 |     "name": "Vitamin String Quartet",
20 |     "country": "USA",
21 |     "city": "Los Angeles",
22 |     "touring": true,
23 |     "yearFormed": 1999,
24 |     "genres": [ "Rock", "Pop", "Instrumental" ]
25 |   }
26 | ]
27 | }
```

# Tools

- JSON formatters, validators and converters
  - <https://jsonformatter.curiousconcept.com/>
  - <http://www.utilities-online.info/xmltojson>
  - <http://www.freeformatter.com/json-formatter.html>
- IDE Plugins:
  - Eclipse - [JSON Tools](#)

## DEMO: Tools and JSON Data

10 min

- Visit the URL for itunes <https://itunes.apple.com/lookup?id=909253>
  - View the data returned
- Open the URL: <http://www.utilities-online.info/xmltojson>
- Copy and Paste the JSON data, transform to XML.
- Do you prefer the XML or the JSON?
- Think About it...
  - How could you use this data to create an information page for music?
  - What pieces of data would you use?

# XML versus JSON

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <bands>
3    <band>
4      <name>Pink Floyd</name>
5      <city>London</city>
6      <country>England</country>
7      <yearFormed>1965</yearFormed>
8      <genres>Progressive rock</genres>
9      <genres>psychedelic rock</genres>
10     <genres>art rock</genres>
11   </band>
12   <band>
13     <name>Morphine</name>
14     <country>USA</country>
15     <city>Boston</city>
16     <genres>Alternative rock</genres>
17     <genres>experimental rock</genres>
18     <genres>jazz rock</genres>
19   </band>
20   <band>
21     <name>Vitamin String Quartet</name>
22     <country>USA</country>
23     <city>Los Angeles</city>
24     <genres>Rock</genres>
25     <genres>Pop</genres>
26     <genres>Instrumental</genres>
27   </band>
28 </bands>
```

```
1 { "bands": [
2   {
3     "band": [
4       {
5         "name": "Pink Floyd",
6         "city": "London",
7         "country": "England",
8         "yearFormed": "1965",
9         "genres": [
10           "Progressive rock", "psychedelic rock", "art rock"
11         ]
12       },
13       {
14         "name": "Morphine",
15         "country": "USA",
16         "city": "Boston",
17         "genres": [
18           "Alternative rock", "experimental rock", "jazz rock"
19         ]
20       },
21       {
22         "name": "Vitamin String Quartet",
23         "country": "USA",
24         "city": "Los Angeles",
25         "genres": [
26           "Rock", "Pop", "Instrumental"
27         ]
28       }
29     ]
30   }
31 ]}
```

# Similarities/Differences between JSON & XML

- Both are called "self describing" (human readable)
  - as long as tags and names are descriptive
- Both have a hierarchy
  - JSON can use arrays and translates directly into Javascript objects
- Both can be parsed and used by lots of programming languages
  - JSON support is built in to JavaScript & Python

More discussion at:

<http://www.json.org/xml.html>

# Processing Data from a Server

- We will be making AJAX calls from the browser to Node servers
- When receiving data, we need to parse the results

# JSON Parsers

- JSON is primarily a string of text
- Needs to be converted to an object to make it useful
- Need to utilize a JSON parser, built in to JavaScript

```
let bandJSONStringData = '{"name" : "Pink Floyd", "city" : "London"}';  
  
let bandObj = JSON.parse(bandJSONStringData);  
  
document.getElementById("demo").innerHTML = bandObj.name + "<br>" + bandObj.city;
```

# The JSON Object has two methods

- `JSON.parse(aString[, reviver])`
  - turns JSON strings into JavaScript Objects
- `JSON.stringify(objectJSON[, replacer[, space]])`
  - turns a JS object into a JSON string

**Web Browsers**

**Support**

Firefox 3.5

Internet Explorer 8

Chrome

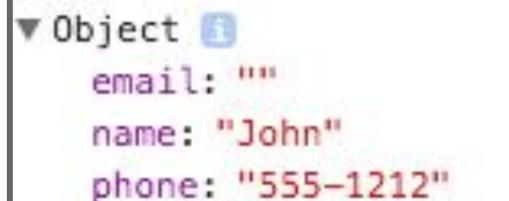
Opera 10

Safari 4

# A simple parse using JSON.parse(aString)

- This is a simple parse

```
let data = '{"name": "John", "email": "", "phone": "555-1212"}';
let o1 = JSON.parse(data);
console.log(o1);
```



## A parse using `JSON.parse(aString[, reviver])`

- Reviver is optional, used to translate values into something more meaningful

```
let o = JSON.parse(data, function(k, v) {  
    if(k === '') { return v; } // if topmost value, return it,  
    console.log(k);  
    if(v === '') {  
        return "unknown";  
    }  
    else return v;  
});  
  
console.log(o);
```

▼ Object ⓘ  
email: "unknown"  
name: "John"  
phone: "555-1212"

- Open the following file in the editor and browser:
  - \Demos\Chapter07-ClientAPIs\JSON\parseReviver.html
- View the Console in Dev Tools
- Set a breakpoint on Line 22
  - watch how the function is called during parsing
  - what could the final iteration, where `k === ''` be used for?

# Review of Ajax

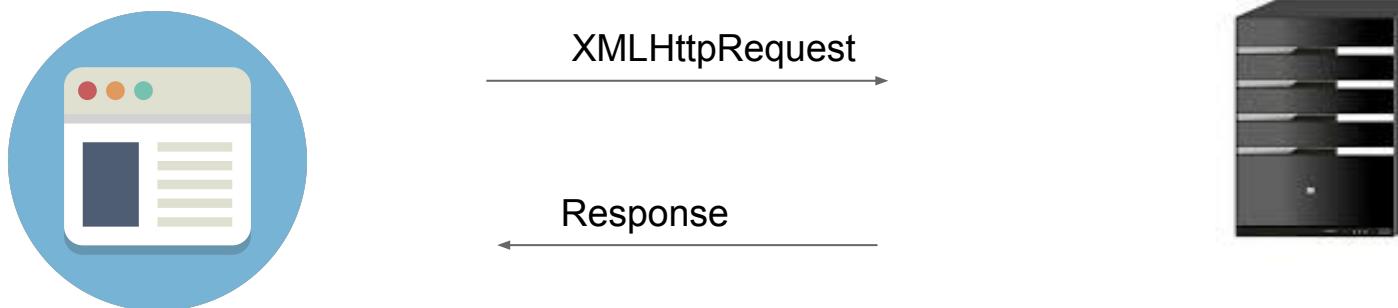
- Asynchronous JavaScript And XML
  - It's not separate code - its JavaScript!
- AJAX is a combination of:
  - A browser built-in XMLHttpRequest object
    - (to request data from a web server)
  - JavaScript and HTML DOM
    - (to display or use the data)

## **With AJAX, you can:**

- Update a portion of a web page without full reload
- Continue to request fresh data after a page has loaded
- Send information in the background to be saved on the server

# How AJAX works

- The client JavaScript makes an XMLHttpRequest and the server sends an HTTP Response
  - The response data is used to update part of the web page rather than replacing the whole page



# How AJAX works

1. An event occurs in a web page (the page is loaded, a button is clicked, etc.)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

# Creating the XMLHttpRequest Object

- Fairly straight-forward, unless need to support old IE browsers

```
16 let xmlhttp;
17 if(window.XMLHttpRequest) {
18     xmlhttp = new XMLHttpRequest();
19 } else {
20     // code for IE6, IE5
21     xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
22 }
```

# Working with the XMLHttpRequest Object

- Setup a function called **onreadystatechange**
  - it will be called when readyState changes

```
29 | xmlhttp.onreadystatechange = function() {  
30 |   if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {  
31 |     let myArr = JSON.parse(xmlhttp.responseText);  
32 |     myFunction(myArr);  
33 |   }  
34 |};
```

# Specify and Send the request

- open: the request type, the URL and a boolean to setup an asynch request
- send: sends the request to the server

```
29  xmlhttp.onreadystatechange = function() {  
30    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {  
31      let myArr = JSON.parse(xmlhttp.responseText);  
32      myFunction(myArr);  
33    }  
34  };  
35  
36  xmlhttp.open("GET", url, true);  
37  xmlhttp.send();
```

# Properties of XHR

Property	Description
<b>onreadystatechange</b>	Defines function to be called when readyState property changes
<b>readyState</b> Holds the status of the XMLHttpRequest.	0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
<b>status</b>	200: "OK" 403: "Forbidden" 404: "Page not found"
<b>statusText</b>	Returns the status-text (e.g. "OK" or "Not Found")

# Setting up the request: `open()` and `send()`

Method	Description
<code>open(<i>method</i>, <i>url</i>, <i>async</i>)  (Specifies the type of request)</code>	<i>method</i> : the type of request: GET or POST  <i>url</i> : the server (file) location  <i>async</i> : true (asynchronous) or false (synchronous)
<code>send()</code>	Sends the request to the server (used for GET)
<code>send(<i>string</i>)</code>	Sends the request to the server (used for POST)

# AJAX Request

- The AJAX request requires several operations
  - Create a handler function to receive the asynchronous response
  - Make the open call with the method and URL
  - Send the request

```
26 let xmlhttp = new XMLHttpRequest();
27 let url = "data/myBand.json";
28
29
30 xmlhttp.onreadystatechange = function() {
31   if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
32     let myArr = JSON.parse(xmlhttp.responseText);
33     myFunction(myArr);
34   }
35 };
36
37 xmlhttp.open("GET", url, true);
38 xmlhttp.send();
```

# AJAX Response

- When the response comes back the function is called
  - It then updates the DOM

```
41  function myFunction(jsonData) {  
42      let out = "";  
43      let bands = jsonData.bands;  
44      let i;  
45      for (i = 0; i < bands.length; i++) {  
46          out += bands[i].name + ' was formed ' + bands[i].yearFormed +  
47          ' in ' + bands[i].city + '<br/>';  
48      }  
49      document.getElementById("showBands").innerHTML = out;  
50  }
```

In this demo:

- start a Node server that allows data to be requested
- view code that requests data asynchronously
- Follow the directions in  
  \Demo\Chapter07-ClientAPIs\AjaxJSON\README.md

In this exercise you will:

- start a Node server that allows data to be requested
- write code to request data asynchronously
- Follow the directions in \Labs\ch07-ClientAPIs\6ajax\README.md

# Client-Side Web APIs

Client-Side (Browser) JavaScript

Transpiling from ES6 to ES5

Canvas API

Web Storage

Geolocation

JSON and AJAX with XHR



## Chapter Summary

Bonus: Overview of Promises & Fetch

# Chapter Summary

In this chapter we have looked at:

- Strategies for adding JS to web pages
- Drawing with the canvas
- Storing data locally with HTML5 Web Storage
- Using Geolocation to get position
- Using AJAX to communicate with the server
- Bonus: Introducing promises & Fetch

# Client-Side Web APIs

Client-Side (Browser) JavaScript

Transpiling from ES6 to ES5

Canvas API

Web Storage

Geolocation

JSON and AJAX with XHR

Chapter Summary



**Bonus: Overview of Promises & Fetch**

# Callback Hell

```
asyncStep1(function (value1) {  
    asyncStep2(value1, function(value2) {  
        asyncStep3(value2, function(value3) {  
            asyncStep4(value3, function(value4) {  
                // Do something with value4  
            });  
        });  
    });  
});  
});
```

# Promises

- Asynchronous communication
  - server, local storage - anything that might take some time
- Better than callbacks
- 3 helpful promise methods to know
  - then - success callback
  - catch - error callback
  - finally - call either way

# A Promise library can flatten the structure

- One benefit is improving multiple nested calls to async functions

```
start()  
.then(promisedStep2)  
.then(promisedStep3)  
.then(promisedStep4)  
.catch() {  
    // handle error  
});
```

With more intuitive API

More at

<http://www.html5rocks.com/en/tutorials/es6/promises/>

# Promises

- Separate libraries were developed

Logos: Q, Bluebird, jQuery



- NOW: a part of EcmaScript

<http://caniuse.com/#feat=promises>

```
1 let ajaxService = function() {
2     this.getJSON = function(url) {
3         let xhr = new XMLHttpRequest();
4         xhr.open("GET", url, true);
5         xhr.setRequestHeader("Accept", "application/json");
6         let deferred = Q.defer();
7         xhr.onload = function(e) {
8             if(xhr.readyState === 4) {
9                 if(xhr.status === 200) {
10                     let data = (xhr.responseText) ? JSON.parse(xhr.responseText) : null;
11                     deferred.resolve(data);
12                 } else {
13                     deferred.reject(xhr);
14                 }
15             }
16         };
17         xhr.onerror = function(e) {
18             console.error(xhr.statusText);
19         };
20         xhr.send();
21         return deferred.promise;
22     };
23     return this;
24 }();
```

Example of a function that  
returns a Promise

# Example Using a Promise

- Line 18 returns a Promise
  - Line 19 is defining the success callback for when data is returned

```
53  ajaxService.getJSON('data.json.text')
54    .then(function(data) { //Success
55      let mappedData = data.map(function(dataItem) {
56        for (let prop in dataItem) {
57          dataItem[prop] = capitalizer.capitalize(dataItem[prop]);
58        }
59      }
60      return dataItem;
61   });
```

# Fetch

- Fetch provides an alternative to using XMLHttpRequest
- Provides an API for accessing parts of the HTTP pipeline
  - Such as requests and responses
- It provides a global `fetch()` method with easy, logical way to fetch resources asynchronously across the network
- More on Fetch here:
  - [https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)

# **Modern Web Development**

## **Chapter 8**

Working with HTML5 Forms



# Chapter Objectives

In this chapter we will look at:

- Using new HTML5 Form tags and attributes
- Enforcing constraints using regular expression and patterns



## Form Basics

HTML5 Form Types and attributes

# Creating HTML Form Elements <form>

- Forms are created between <form> </form> tags

```
41  <form action="/api/lookupBand" method="GET">
42    <div>
43      <h1>Search for a band</h1>
44      <label>
45        Band Name
46        <input type="text" name="bandName">
47      </label>
48      <label class="inline-block" for="lname">Genre</label>
49      <select>
50        <option value="rock">Rock</option>
51        <option value="pop">Pop</option>
52        <option value="hiphop">HipHop</option>
53      </select>
54      <input type="submit" value="search">
55    </div>
56  </form>
```

## Search for a band

Band Name

Genre

# The <input> Element: type attributes

- The <input> element is the most used form element
- The type determines the control that is rendered:

type="text"	Defines a one-line text input field
type="radio"	Defines a radio button (selecting one option)
type="checkbox"	Defines checkboxes (1+ option)
type="submit"	Defines a submit button (for submitting form)

# Input type: text

- In traditional web dev, inputs **REQUIRE** a **name** attribute
  - Used to pass and identify the data on the server
  - If using Ajax to send JSON data, not required

```
First Name: <input type="text" name="fname" />
```

First Name:

# Using labels

- Recall that Section 508 requires labels
- Two ways to associate labels with inputs
  - wrapping
  - using for with an id value

```
64      <label>First Name: <input type="text" name="fname" /></label>
65
66      <label for="lname">Last Name</label>
67      <input id="lname" type="text" name="fname" />
```

# Input type: textarea

- Sometimes need more than one line of input
- Can set the number of rows and columns
  - resizable, can all set size using CSS

Enter lots of text here:

Default Text here

```
<label>
    Enter lots of text here:<br />
    <textarea name="textarea" rows="3" cols="30">Default Text here</textarea>
</label>
```

# Radio Buttons

- Group items to the same radio button group
  - by using the same **name** value
  - only **one** value can be checked

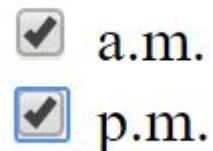
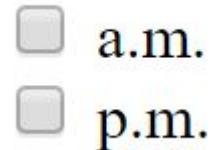
```
<form>
  <input type="radio" name="answer" value="yes"> Yes
  <br />
  <input type="radio" name="answer" value="no"> No
  <br />
  <input type="radio" name="answer" value="other"> Other
</form>
```

- Yes
- No
- Other

- Yes
- No
- Other

# Checkboxes

- Group items to the same checkbox group
  - More than one value can be checked
  - Assign to group by using the same **name** value



```
<form>
  <input type="checkbox" name="preferredTimes" value="am"> a.m.
  <br />
  <input type="checkbox" name="preferredTimes" value="pm"> p.m.
  <br />
</form>
```

# Fieldset & Legend

- The `<fieldset>` element is used to group related data in a form.
- The `<legend>` element defines a caption for the `<fieldset>` element.

```
<form action="/submitform" method="post" target="_blank">
  <fieldset>
    <legend>Personal information:</legend>
    <label>Username:<br/>
      <input type="text"></label>
    <label>First name:<input type="text" name="firstname"></label>
    <label>Last name:<input type="text" name="lastname"></label>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

Personal information:

Username:	<input type="text"/>
First name:	<input type="text"/>
Last name:	<input type="text"/>
<input type="button" value="Submit"/>	

# Boolean Attribute: checked

- To load a page with items already selected, for edit screens:
    - Use **checked="checked"** or just have **checked** present
    - Can be used with radio buttons, checkboxes
- ```
<input type="radio" name="answer" value="yes" checked>
```

- Yes
- No
- Other

# Boolean Attribute: disabled

- Disable buttons if data is incomplete, to prevent a submit
- Display fields that cannot be edited

```
<form action="/submitform" method="post" target="_blank">
  <fieldset>
    <legend>Personal information:</legend>
    <label>Username:<br/>
      <input type="text" disabled="disabled" value="admin"></label>
    <label>First name:<input type="text" name="firstname"></label>
    <label>Last name:<input type="text" name="lastname"></label>
    <input type="submit" value="Submit" disabled>
  </fieldset>
</form>
```

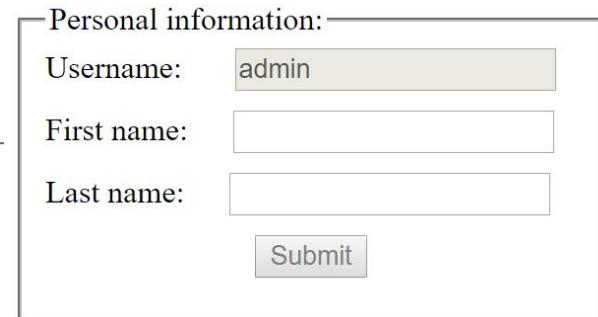
Personal information:

Username: admin

First name:

Last name:

Submit



# Submit Actions

- Input type of submit will allow a form to be submitted to server
  - Specify where via the **action** - *default is the same page*
  - Specify a GET or POST with **method**
  - The server returns response to the **target** - *default is same page*

```
38 <form action="/submitform" method="get" target="_blank">
39     First name:
40         <input type="text" name="firstname">
41         <br /> Last name:
42         <input type="text" name="lastname">
43         <br /> <input type="submit" value="Submit">
44     </form>
```

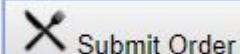
The image shows a screenshot of a web browser displaying a simple HTML form. The form consists of two text input fields, one for 'First name' and one for 'Last name', both with placeholder text. Below the input fields is a single 'Submit' button.

First name:	<input type="text"/>
Last name:	<input type="text"/>
<input type="button" value="Submit"/>	

# Buttons

- Represents a clickable button
  - can have text or images
- The type attribute can change behavior:
  - **submit**: submits form data to the server (default if not specified)
  - **reset**: resets all the controls to their initial values
  - **button**: no default behavior, can use onclick to specify a function
- Outside of a form, JavaScript function needs to be available

```
<button onclick="validateOrder">  
     Submit Order  
</button>
```



Favorite food:

```
<form>  
    <label>  
        Favorite food:  
        <input type="text" name="food" />  
    </label>  
    <br/>  
    <button>Submit</button>  
</form>
```

# LAB: Create a registration form

10 min

- Navigate to the folder \Labs\Ch08-forms\basic
- Open the file README.md and follow steps to create a registration form

# HTML5 Forms

Form Basics



## HTML5 Form Types and attributes

# HTML5 Forms: New Inputs and Attributes

- Previously <input type="text"..> was used for email, number, date etc.
  - Validation was performed with custom JavaScript
- HTML5 now provides built-in data validation types for most common fields
  - Without the need for JavaScript
  - If browser doesn't support new type, it falls back to the text type
  - A quick approach for simple validation

# HTML5 Form Additions

- The new input types and attributes are not supported by all browsers
  - We will review the following:
- New attributes
  - autofocus, autocomplete, placeholder, datalist
  - required, multiple, pattern (regular expression)
  - min, max, step
- New input types
  - email, url, number, tel, range, color, output, date, datetime-local

# The `autofocus` and `required` Attributes

- `autofocus` places cursor in field on page load, only 1 per page

```
<li><label for="name">Name:</label> <input type="text"  
name="name" id="name" autofocus required /></li>
```

- `required` on submit, error if no value

A screenshot of a web browser showing a form field. The field is labeled "Name:" and contains a single character "I". To the right of the input field is a red circular icon with a white exclamation mark. Below the input field is a tooltip-style message box with a yellow exclamation icon and the text "Please fill out this field.".

# autocomplete Attribute

- Prevent automatic completion from the likes of LastPass, Chrome, etc.
  - By using **off** forces them to type, ie. passwords
- Can be set at the form level or the field level
  - Selectively control using **on** where it is desired

Password \*

  
.....

Repeat Password \*

  
.....

```
42 <div>
43     <label>Password *
44         <input name="password" type="password" required autocomplete="off">
45     </label>
46 </div>
```

# placeholder Attribute

- Give users an understanding of what format to enter text

```
<label>URL <input id="url" name="url" type="url"  
placeholder="http://mysite.com">  
</label>
```

URL

# New Input Type: email

- Validates email addresses
  - Only looks for the pattern text@text - not the extension

```
18 <li><label for="email">Email:</label>
19   <input type="email" name="email"
20     placeholder="john_doe@example.com" required />
21   <span class="form_hint">Proper format: name@something.com</span></li>
```

Email:

john\_doe@example.com



Proper format: name@something.com

## New Input Type: tel

- The tel type expects a telephone number
  - No validation, but a mobile device will display the number keypad

```
90 <div>
91     <label>Telephone * <input id="phone" name="phone" type="tel"
92         placeholder="Eg. 412-353-9161" required>
93     </label>
94 </div>
```

# Want regular expressions? Use pattern attribute

- Does not make it required, but if value is given, must match pattern
- Gives more control to developer in what pattern to validate against
  - A phone number with an optional area code might use:

```
91  <label>Telephone * <input id="phone" name="phone" type="tel"
92    placeholder="Eg. 412-353-9161" required
93    pattern="^\D?(\d{3})\D?\D?(\d{3})\D?(\d{4})$">
94  </label>
```



# New Input Type: number

- Appears as a numeric spinner
  - Works with the min, max, and step attributes

```
104  

Shoesize <input id="shoes" name="shoes" type="number" min="5" max="18" step="0.5" value="9">


```

```
105  <div>
106    <label>Shoesize <input id="shoes" name="shoes" type="number"
107      min="5" max="18" step="0.5" value="9">
108    </label>
109  </div>
```

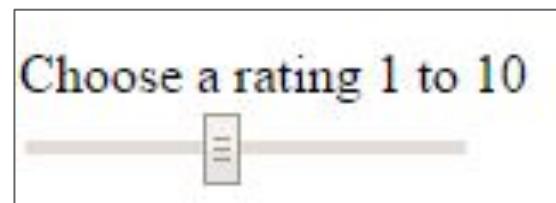
Shoesize

# New Input Type: range

- Can be used to implement sliders
  - Sliders are commonly used to decrease or increase a numerical value

```
17  <div>
18      <label for="input1">Choose a rating 1 to 10</label><br/>
19      <input id="input1" name="rating" type="range"
20          min="1" max="10" step="1" value="5" >
21  </div>
```

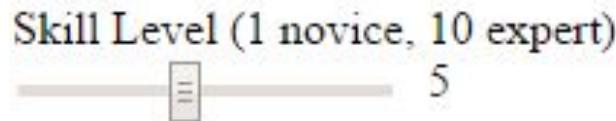


Minimum IE = 10

## New Element: Output

- Used to display the result of a calculation or user action (like sliding)

```
24  <label for="skillInputId">Skill Level (1 novice, 10 expert)</label><br/>
25  <input id="skillInputId" name="skill" type="range"
26  min="1" max="10" step="1" value="5"
27  oninput="skillOutputId.value = skillInputId.value">
28  <output name="skillOutputDisplay" id="skillOutputId">5</output>
```



No support in IE,  
support in Edge

# New Input Type: color

- Allows the user to input a color value via a picker

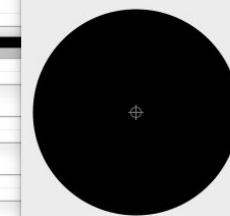
```
<label for="colorid">Select Color</label>
<input type="color" name="favorite-color" id="colorid">
```

- Not supported in Internet Explorer or Safari
- Returns six digit hex value
  - #RRGGBB

No support in IE,  
support in Edge

Contact Us

Name:	<input type="text" value="John Doe"/> *
Email:	<input type="text" value="john_doe@example.com"/> *
Grade:	<input type="text"/>
Select Priority	<input type="radio"/>
Select Color	<input type="color"/>
Enter telephone	<input type="text"/>
Website:	<input type="text" value="http://johndoe.com"/>
Message:	<input type="text"/>



Submit Form

# The HTML <datalist> element

- Contains a set of <option> elements representing values for **other** controls
- Match a **list** attribute to the **datalist**

```
122     <input id="desiredPosition" type="text" list="positionsList" size="30">  
123  
124     <datalist id="positionsList">  
125         <option value="Setter">  
126         <option value="Outside Hitter">  
127         <option value="Right side hitter">  
128     </datalist>
```

No support in Safari



# New Input Type: **date**

No support in Safari, IE, Firefox

- Generates a pop-up window

```
<label for="schedule-date">Select Date</label>
<input type="date" name="scheduleDate" id="schedule-date">
```

- displayed format* is based on the user's operating system locale
- date value* is always formatted yyyy-mm-dd.

Select Date	<input type="text" value="mm/dd/yyyy"/> <span>▼</span>
Select Color	<input type="color"/>
Enter telephone	<input type="text"/>
Website:	<input type="text" value="http://johndoe.com"/>

Chrome

Edge

mm/dd/yyyy	21	2012
January	21	2012
February	22	2013
March	23	2014
April	24	2015
May	25	2016
June	26	2017
July	27	2018
August	28	2019
September	29	2020
October	30	2021
November	1	2022

# LAB: Create an HTML5 Form

10 min

- Examine the file `feedback.html` in `\Labs\ch08-form\newform`
- Replace the input tags with HTML5 tags and attributes
- Make the fields required as marked by \*
- Add autofocus to the name field
- Use placeholders for: name, email, and telephone
- Use a regular expression to validate phone
- Load your work into a browser and test that it works as expected
- Solution is in the `\Form\End` folder

The screenshot shows a web form titled "Your Feedback". It contains the following fields:

- A text input field labeled "Your Name:" with a placeholder "Your Name" and a required indicator (\*) to its right.
- A text input field labeled "E-mail:" with a placeholder "anything@example.com" and a required indicator (\*) to its right.
- A text input field labeled "Phone:" with a placeholder "555-555-5555".
- A text input field labeled "Date:" with no placeholder or required indicator.
- A text input field labeled "Comments:" with a required indicator (\*) to its right.

At the bottom of the form are two buttons: "Submit" and "Reset".

# Chapter Summary

In this chapter we have looked at:

- Using new HTML5 Form tags and attributes
- Enforcing constraints using regular expression and patterns

# **Modern Web Development**

## **Chapter 9**

JavaScript Modules



# Chapter Objectives

In this chapter we will look at:

- Common Pitfalls in classic JavaScript
- Operating in “strict mode” and work-arounds such as IIFEs
- ES6 Modules & Module Loading



## Historical issues

ES6 Modules

Module Loading

# JavaScript can be quite messy, enabling bugs

```
32   <script src="setX.js"></script>
33   <script>
34     var myConstantVars = {
35       xval: x,
36       pi: 3.14
37     }
38
39     xval = 3;
40
41     x = 'Hello';
42
43     console.log('xval is ' + myConstantVars.xval);
44     myConstantVars = { pi: 3.1};
45
46     setTimeout(function () { alert(myConstantVars.pi); }, 3000);
47
48     console.log('now xval is ' + myConstantVars.xval);
49   </script>
```

- JavaScript is extremely flexible
  - This can lead to bugs
- What could go wrong in this code?

- Scripts may depend on one another, the order needs to be preserved
- Typos can overwrite the global namespace

# Dependency management can be difficult & error-prone

Managing dependencies can be confusing

```
9  <body>
10     <!-- Do not change the order of these scripts! -->
11     <script src="jquery.js"></script>
12     <script src="jqueryui.js"></script>
13     <script src="../../someFunctions.js"></script>
14     <script src="../somedir/somedir2/otherStuff.js"></script>
15 </body>
```

- What can go wrong in this example?

Bad ordering of scripts, missing scripts, scripts writing over other variables, functions, etc.

# Scripts (or functions) can be set to use strict mode

```
32  <script src="setX.js"></script>
33  <script>
34      'use strict'
35      var myConstantVars = {
36          xval: x,
37          pi: 3.14
38      }
39
40      xval = 3;
41
42      x = 'Hello';
43
44      console.log('xval is ' + myConstantVars.xval);
45      myConstantVars = { pi: 3.1};
46
47      setTimeout(function () { alert(myConstantVars.pi); }, 3000);
48
49      console.log('now xval is ' + myConstantVars.xval);
50  </script>
```

- Prevents some potential issues and bad practices
- Variables must be declared before use
- Can be used to catch typos

✖ ▾ Uncaught ReferenceError: x is not defined  
at gotchas.html:36  
(anonymous) @ gotchas.html:36

▶ Uncaught ReferenceError: xval is not defined  
at gotchas.html:40

# Difficult to protect values from being changed

- Was the object meant to be replaced or just a value changed?

```
var myConstantVars = {           myConstantVars = { pi: 3.1};  
    xval: x,  
    pi: 3.14  
}
```

- Is there a private modifier in JavaScript?

NO

# Design Patterns such as IIFE can help

- IIFE = Immediately invoked function expression
  - Protects an object by declaring it's usage in an IIFE
  - cannot be accessed outside the expression

```
11  <script>
12    (function (x) {
13
14      var myConstantVars = {
15        xval: x,
16        pi: 3.14
17      }
18
19      setTimeout(function () {
20        alert(myConstantVars.xval); }, 3000);
21
22    }(22.5));
23  </script>
```

the parenthesis causes the code to execute right away in its own code block. Passing data is optional.

We will use newer approaches, but you may encounter other patterns

Other work-arounds:

- module
- revealing module
- closures

# JavaScript Modules

Historical Issues



**ES6 Modules**

Module Loading

# What are Modules?

- A way of organizing JavaScript code
  - Each .js file is a module
  - Everything in files is PRIVATE by default
  - Can indicate what should be publicly available outside of module
- Use modules as dependencies in other files
  - Can require/import modules into other code
  - Simplifies dependency management
  - Code reusability, decoupling, extensibility

# Modules execute in their own scope

- Modules are executed within their own scope, not in the global scope
  - variables, functions, classes, declared in module not visible
    - unless they are explicitly exported
- To use an exported item in a different module, it has to be imported
- Modules prevent polluting the global namespace

# Before ES6 (ES2015): two standards

- **Asynchronous Module Definition (AMD):**
  - Is used in DOJO and RequireJS
  - Loads modules asynchronously
- **CommonJS Modules:**
  - Is used in Node.js
  - Compact syntax
  - Loads modules synchronously

# CommonJS syntax (used in Node)

- Uses **module.exports** & **require**

```
3 module.exports = function arrayToText(array) {  
4     return array.join(", ");  
5 }
```

array-to-text.js

```
5 const arrayToText = require("./array-to-text");  
6  
7 document.addEventListener("DOMContentLoaded", function() {  
8  
9     var text = arrayToText(combinedArray);  
10  
11     document.getElementById("text").innerHTML = text;  
12 });
```

example.js

# ES6 Modules: use **import** and **export** as keywords

JS index.js ×

You, 40 minutes ago | 2 authors (Judy Lipinksi and others)

```
1 import myArrayToText from './array-to-text';
2
3 document.addEventListener("DOMContentLoaded", function() {
4     var myArray = [1, 2, 3, 4, 5, 6, 7];
5     var text = myArrayToText(myArray);
6     document.getElementById("text").innerHTML = text;
7 });
```

ES6 modules are used by  
Ember, Angular, React...

JS array-to-text.js ×

You, a few seconds ago | 2 authors (Judy Lipinksi and others)

```
1 export default function (array) {
2     return array.join(", ");
3 }
```

# ES6 has 2 types of exports: default and named

- Can have multiple named exports per module but only 1 **default** export

```
JS exportExamples.js ×  
1 //private info here  
2 function myFunction() { return "MINE"; }  
3  
4 // ***** NAMED exports  
5 export { myFunction }; // exports above function  
6 export const foo = Math.sqrt(2); // exports a constant  
7  
8 // ***** default export (can only have 1)  
9 export default function() {} //export function  
10  
11 // export default class {} //Example: export class
```

for named imports, it is mandatory to use the same name of the corresponding object

```
1 //Example of Binding  
2 import {foo, myFunction} from "./exportExamples";  
3  
4 import * as eeLib from "./exportExamples";  
5  
6 let message = eeLib.myFunction();  
7 console.log(message);
```

# Classic vs ES2015 functions in modules

```
JS application.js x
1 import Controller from '@ember/controller';
2
3 export default Controller.extend({
4   actions: {
5     raiseAlert: function() {
6       alert('Button was clicked');
7     },
8
9     somethingClicked: function(whatWasClicked) {
10       alert(`#${whatWasClicked} was clicked`);
11     },
12   }
13 })
```

Classic Functions

save: function() {}  
vs  
save( ) {}

```
JS application.js x
1 import Controller from '@ember/controller';
2
3 export default Controller.extend({
4   actions: {
5     raiseAlert() {
6       alert('Button was clicked');
7     },
8
9     somethingClicked(whatWasClicked) {
10       alert(`#${whatWasClicked} was clicked`);
11     },
12   }
13 })
```

ES2015 Functions

# JavaScript Modules

Historical issues

ES6 Modules



**Loading Modules**

# Modules are not supported in browsers

- Need to first have a process scan the code, sort out dependencies and create a bundle
- Examples of popular module bundlers are:
  - **Browserify**: bundler for CommonJS modules
    - And ES6 if you use *Babelify*
  - **Webpack**: bundler for AMD, CommonJS, ES6 modules

# Client-side modules with Browserify

- Allows modules to be compiled for use in the browser
- <http://browserify.org/>



# Using browserify to compile code

- When using ES6 imports, need to transpile using babel before bundling

```
"build": "browserify src/index.js -o dist/bundle.js -t babelify --presets"
```

- The babelify package can be used with the browserify command
  - Works similar to as before, using .babelrc for presets
- The package.json dependencies when using Browserify

```
24  "devDependencies": {  
25    "babel-core": "^6.26.0",  
26    "babelify": "^8.0.0",  
27    "babel-cli": "^6.26.0",  
28    "babel-preset-env": "^1.6.1",  
29    "browserify": "^16.1.1"  
30  },
```

## DO NOW: Review use of Browserify

10 min

1. In VSCode, navigate to /Demos/Chapter09-ModernJS/modulesES6
2. Look over the code in the /src directory and the **build** script in package.json
3. Use **npm install** to get the dependencies
4. Use **npm run build** to execute the build script
  - a. notice the file created in the dist directory
5. Open the test.html in Chrome and IE

# Chapter Summary

In this chapter we have looked at:

- Common Pitfalls in classic JavaScript
- Operating in “strict mode” and work-arounds such as IIFEs
- ES6 Modules & Module Loading

# Modern Web Development

## Bonus: Chapter 10

Unit Testing JavaScript



# Chapter Objectives

In this chapter we will:

- Learn about unit testing JavaScript code
- Demonstrate assertion frameworks and testrunners
- Karma & Jasmine for front-end testing

# Unit Testing JavaScript



## Intro to Unit Testing

Introducing Mocha

Assertions with Chai

Code coverage tools

# Testing Applications

- Testing provide safeguards
- Tests document systems
- Types of tests
  - **Unit:** Small tests covering individual functions
  - **Integration:** Integrated modules (combined functionality)
  - **Functional:** Focused on end result of larger actions
  - **Acceptance:** Typically client-side, end-to-end testing

# Unit Tests

- Unit tests to see if a given module works
- All the dependencies are stubbed
  - we are providing fake dependencies for a module
- Test the API
  - write the test for the exposed methods, not for the internal workings of the given module

# **Unit tests have the following structure:**

- Test setup
- Calling the tested method
- Assertions that expected behavior happened

# **What tech is available to help test?**

- Jasmine
- Karma
- Chai
- Mocha
- QUnit
- many others...

# Unit Testing JavaScript

Intro to Unit Testing

## ➤ **Introducing Mocha**

Assertions with Chai

Code coverage tools

# Mocha is a feature-rich JavaScript test framework

- Mocha is free and open source, licensed under the MIT license
  - <https://mochajs.org/>
- Runs on Node.js and in the browser
- Makes asynchronous testing simple
- Mocha tests run serially
  - for flexible and accurate reporting
  - maps uncaught exceptions to the correct test cases
- Hosted on GitHub



simple, flexible, fun

# Getting Started with Mocha

- Mocha is easy to get up and running

```
npm install mocha --save-dev
```

```
"scripts": {  
  "test": "mocha"  
},
```

- Create a script in package.json to refer to mocha

- You can then run mocha testing a single file or a directory of files:

**npm test [file-to-test.js]**

**npm test [directory-to-test]**

# Sample Test File

```
const assert = require('chai').assert;
const Band = require('../app/band');
describe('Band', () => {

  it('should return a member count', () => {
    // setup
    const band = new Band('The Beatles', ['John', 'Paul', 'George', 'Ringo']);

    //execute
    const count = band.memberCount();

    //verify
    assert.equal(4, count);
  });
});
```

bandTest.js

# Unit Testing JavaScript

Intro to Unit Testing

Introducing Mocha

➤ **Assertions with Chai**

Code coverage tools

# Using chai as the assertion library

- Chai is a TDD and BDD\* assertion library.
- Supports traditional TDD-style assertions
- Also support BDD-style “should” and “expect” syntax
- Works with any JavaScript testing framework
- Add to your project with `$ npm install chai --save-dev`

\* BDD means Behavior-driven development

# Three assertion styles

- Pick whichever style you like, but be consistent on your team and project

```
it('is 42', () => {
  const value = 42;
  assert.equal(42, value);
});
```

```
it('is 42', () => {
  const theAnswer = 42;
  theAnswer.should.equal(42);
});
```

```
it('is 42', () => {
  const theAnswer = 42;
  expect(theAnswer).to.equal(42);
});
```

# Writing test cases

- All the tests for a module or class are grouped into a test file
- The test file serves as a regression suite and documentation for that module under test
- With mocha, tests are grouped using `describe()` block
- Each test goes in an `it()` block

# Describing modules

- mocha allows nested describe blocks to logically group tests

```
describe('Band', () => {
  describe('composing', () => {
    // tests go here...
  });

  describe('touring', () => {
    // tests go here...
  });
});
```

# Writing tests

- Each test goes in an `it` block

```
describe('Band', () => {
  it('should return a member count', () => {
    const band = new Band('The Beatles', ['John', 'Paul', 'George', 'Ringo']);
    assert.equal(4, band.memberCount());
  });

  it('contains a member', () => {
    const band = new Band('The Beatles', ['John', 'Paul', 'George', 'Ringo']);
    assert(band.hasMember('John'));
  });
});
```

# Testing asynchronous code

- To test asynchronous code with Mocha, invoke the `done()` callback, passed to `it()`, when your test is complete. Mocha will wait for this function to be called to complete the test.

```
const asyncFunc = require('../app/asyncFunc');
const assert = require('chai').assert;

describe('async', () => {
  it('run without an error', (done) => {
    asyncFunc(function() {
      done();
    })
  });
});
```

# Testing promises

- When testing code that returns promised promises the `done` `callback()` is not necessary. Using the Chai as Promised plugin we can test like this.

```
describe('promise', () => {
  it('run without an error', () => {
    assert.eventually.equal(promiseFunc(), 100);
  });
});
```

# Unit Testing JavaScript

Intro to Unit Testing

Introducing Mocha

Assertions with Chai

➤ **Code coverage tools**

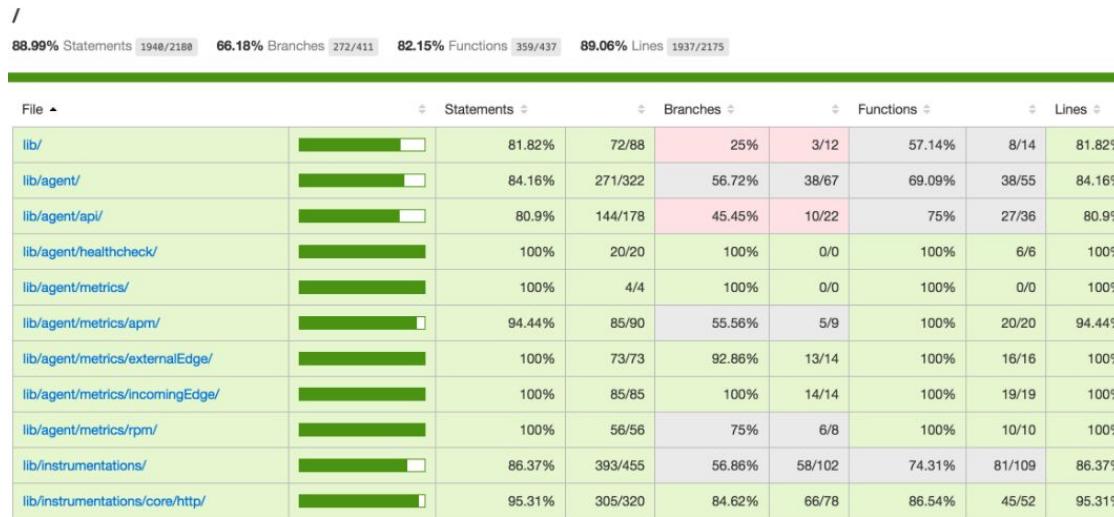
# Code Coverage

- To get a better idea of how well your codebase is covered with tests you can generate a coverage report
- This report will include metrics on:
  - line coverage
  - statement coverage
  - branch coverage
  - function coverage

# Istanbul is a code coverage tool

A script can be added to package.json to use istanbul with mocha:

```
istanbul cover _mocha $(find ./lib -name \"*.spec.js\"  
-not -path \"./node_modules/*\")
```



# **DO NOW: Testing**

**10 min**

- Open /Demos/Chapter10-Testing/README.md
- Follow the directions in README.md to practice with testing

# Chapter Summary

In this chapter we have looked at:

- Learn about unit testing JavaScript code
- Demonstrate assertion frameworks and testrunners

# Modern Web Development

## Chapter 11

jQuery Intro & Selectors



# Chapter Objectives

In this chapter we will look at:

- The differences between jQuery versions and minified
- Where to place jQuery code to execute
- What is the jQuery object?
- Using jQuery Selectors with html tags, id and classes
- Avoiding conflicts with more than one framework

# **jQuery Overview & Selectors**



## **Overview of jQuery**

Using a CDN

Introduction to jQuery Selectors

Bonus: Avoiding Conflicts

# Overview of jQuery

- A single JavaScript Library file
  - Provides cross-browser support
- Powerful way to select HTML elements in a page
- Animate HTML elements
- Easily make Asynchronous JavaScript and XML (AJAX) calls
- Modify CSS styles easily
- Add event behavior to HTML without bloating markup

# jQuery provides different script options:

- jQuery 1.x – 2006 Supports IE 6/7/8 + modern browsers *last 1.12*
- jQuery 2.x – 2013 Supports modern browsers (no IE 6/7/8 support) 2.2
- jQuery 3.x – Released 2016
  - jQuery-Compat 3.0.0 package supports IE8
  - jQuery 3.2.1 Mar 2017 bug fixes, improvements, some deprecations
  - jQuery 3.3.0 and 3.3.1 Jan 2018 - many deprecations
    - Run all tests from command line with grunt and karma
- Versions with **min** in the name have been minified
- Versions with **slim** in the name have no Ajax, effects, or deprecated code

# Where do we put jQuery code?

- Download the jQuery script from <http://jquery.com> or use CDN
- Reference script within the <head> or at the bottom of the <body>
- The \$(document).ready is often used to wait until page is loaded
  - Must include jQuery library first!

```
<script src="http://code.jquery.com/jquery.js"></script>

<script>
    $(document).ready(function() {
        //jQuery code located here will execute once the page loads
        console.log("Hello");
    });
</script>
```

# Other ways to attach jQuery

- `$( handler )`
- `$( document ).ready( handler )`
- `$( "document" ).ready( handler )`
- `$( "img" ).ready( handler )`
- `$( ).ready( handler )`

As of jQuery 3, which is the only 1 that is recommended?

- the rest have been deprecated



**the first is now the  
recommended practice**

# The recommended way to include jQuery

```
5 | <script src="http://code.jquery.com/jquery-3.3.1-min.js"></script>
6 | <script>
7 |   $(function () {
8 |     // jQuery code located here will
9 |     // execute once the page loads
10 |     console.log("Hello");
11 |   });
12 | </script>
```

# **jQuery Overview & Selectors**

Overview of jQuery



## **Using a CDN**

Introduction to jQuery Selectors

Avoiding Conflicts

# Examples of jQuery Content Delivery Network (CDN)

- These are examples of some of the CDNs available for jQuery:

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
```

```
<script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
```

```
<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

# What if the CDN is Down?

- If there was a problem with the CDN you can load the script locally:

```
<script src="http://code.jquery.com/jquery.js"></script>
<script>    window.jQuery || 
              document.write('<script src="../../js/jquery-2.2.2.js"></script>');
</script>
```

# **jQuery Overview & Selectors**

Overview of jQuery

Using a CDN



**Introduction to jQuery Selectors**

Avoiding Conflicts

# jQuery Selectors are based upon CSS3

- Return a single element or multiple elements
- Can select based upon HTML, class, id, child nodes, and more
- The syntax uses a \$ or jQuery, parenthesis, and **double quotes**

`jQuery("h3") selects all <h3> elements`

`$ ("p") selects all <p> elements`

# jQuery selectors return jQuery objects

- You can access properties that contain info
  - `$.().length` – gives the number of selected items
  - `$.().jquery` – gives the version of jQuery

```
11 $(document).ready(function() {  
12     console.log("there are " + $("div").length + " divs on this page");  
13  
14     var nonJQ = { someProp: "A regular JS object" },  
15         isJQ = $( "body" );  
16  
17     if ( nonJQ.jquery ) { // Falsy, since version String is undefined  
18         console.log( "a is a jQuery object!" );  
19     }  
20  
21     if ( isJQ.jquery ) { // Truthy, since string is defined  
22         console.log( "b is a jQuery object - using " + isJQ.jquery );  
23     }
```

there are 4 divs on this page  
b is a jQuery object - using 2.2.2

# Selecting by HTML Element Name

- To select all div tags and give them a 1px solid blue border

```
$("div").css('border', '1px solid blue');
```

- Once selected, can call functions on selection
  - **css** can change styling
  - **click** can add a click event
  - we will start with these functions, then explore more

# Selecting by #ID

- Use the # character with the id value



How to select <button id="changeBlue"> element by id?

```
$("#changeBlue").click(function() {  
    $(this).css('background-color','lightblue');  
});
```

This also adds a click event, so that when the button is clicked, it's (this) background-color is set to blue

Why is \$ needed around this? \$(this)

to wrap this in a jQuery object and be able to call css()

# Selecting by .className

- Use the . character with the class value



How to select <button class="colorPink"> element by class and set the font color to pink?

```
$(".colorPink").css('color','pink');
```

This changes the color to pink for any marked items

# Selecting Multiple Elements by using comma

- Use the comma to separate elements:

```
$("p, a, span")
```

selects all paragraphs, anchors, and span elements

```
$(".important, .notification")
```

selects all elements containing the class `.important` or `.notification`

# Selector Review

- Syntax for a selector = `$( "selector" )`

# in front = element by **ID**

```
$( "#myID" )
```

. in front = element by **Class**

```
$( ".myClass" )
```

'name' = tag/element by **Name**

```
$( "div" )
```

# Exercise: Use jQuery Selectors

20 min

- Open /Labs/Ch11-jQuerySelectors/
- Follow the directions in README.md to practice with jQuery Selectors

# Chapter Summary

In this chapter we have looked at:

- The differences between jQuery versions and minified
- Where to place jQuery code to execute
- What is the jQuery object?
- Using jQuery Selectors with html tags, id and classes
- Bonus: Avoiding conflicts with more than one framework

# **jQuery Overview & Selectors**

Overview of jQuery

Using a CDN

Introduction to jQuery Selectors



**Bonus: Avoiding Conflicts**

# **4 Ways to avoid conflicts in jQuery**

1. Create a new alias
2. Use an Immediately Invoked Function Expression (IIFE)
3. Use argument passed to the `jQuery( document ).ready()` function
4. Use the more concise syntax for the DOM ready function

# Create a new alias

- Use `jQuery.noConflict()` to create an alias
  - Can mix prototype and jQuery

```
12      <script src="prototype.js"></script>
13      <script src="jquery.js"></script>
14      <script>
15          // Give $ back to prototype.js; create new alias to jQuery.
16          var $jq = jQuery.noConflict();
17      </script>
```

# Use an Immediately Invoked Function Expression (IIFE)

- Keep using \$ by wrapping your code in an IIFE
  - This is also a pattern for jQuery plugin authoring
  - Can **NOT** mix prototype and jQuery

```
20      <!-- Using the $ inside an immediately-invoked function expression. -->
21      <script src="prototype.js"></script>
22      <script src="jquery.js"></script>
23      <script>
24
25          jQuery.noConflict();
26
27          (function ($) {
28              // Your jQuery code here, using the $
29          })(jQuery);
30
31      </script>
```

# Use argument passed to the ready function

- Use argument passed to the `jQuery( document ).ready()` function

```
<script src="jquery.js"></script>
<script src="prototype.js"></script>
<script>

    jQuery(document).ready(function ($) {
        // Your jQuery code here, using $ to refer to jQuery.
    });

</script>
```

# Use the concise syntax for the DOM ready function

```
43    <script src="jquery.js"></script>
44    <script src="prototype.js"></script>
45    <script>
46
47        jQuery(function ($) {
48            // Your jQuery code here, using the $
49        });
50
51    </script>
```

# Modern Web Development

## Chapter 12

CSS manipulation and more jQuery Selectors



# Chapter Objectives

In this chapter we will start:

- Further manipulating styles with css()
  - css()/addClass()/hasClass()/removeClass()/toggleClass()
- Utilizing more jQuery selectors
- Traversing selections
- Using method chaining
- Using jQuery.each() and .each()

# CSS manipulation and more jQuery Selectors



## Manipulating styles with css()

More selectors and filters

Traversing

Using method chaining

Using jQuery.each() and .each()

# Setting css style using .css(propertyName,value)

- As we have practiced, you can set a single property and its value
- You can also set multiple properties using an object

```
62      $("#button1").css('background-color', 'yellow');  
63  
64      $("#button2").css({  
65          "width": 100,  
66          "height": 50,  
67          "color": "black",  
68          "background-color": "yellow",  
69          "border": "2px solid black",  
70          "border-radius":'5px'  
71      });
```

# Getting computed style with .css()

- Can pass 1 or more property names and get back the computed style

```
73  let styleProps1 = $("#button1").css("background-color");
74  console.log(styleProps1);
75
76  let stylePropsMany = $("#button2").css([
77    "width", "height", "color", "background-color"
78  ]);
79  console.log(stylePropsMany);
```

```
rgb(255, 255, 0)
▼ {width: "100px", height: "50px", color: "rgb(0, 0, 0)", background-color: "rgb(255, 255, 0)"}
  background-color: "rgb(255, 255, 0)"
  color: "rgb(0, 0, 0)"
  height: "50px"
  width: "100px"
▶ __proto__: Object
```

# Modifying CSS Classes

- We can also add or remove class attribute values
  - `addClass()`
  - `hasClass()`
  - `removeClass()`
  - `toggleClass()`

## Adding one or more class names

- `addClass()` adds one or more class names to the class attribute of each matched element:

```
$("div").addClass('box')
```

More than one class:

```
$("div").addClass('box shadow')
```

# Checking class values with hasClass()

- Returns true if the selected element has a matching class that is specified:

```
14  if ($("#div").hasClass('box')) {  
15    |   console.log('it is a box');  
16  } else {  
17    |   console.log('it is NOT a box');  
18  }
```

## Removing CSS Classes with `removeClass()`

- Removes class names from the class attribute of each matched element
- You can remove one or more specific classes:

```
$("div").removeClass('box important');
```

# Using toggleClass to alternate

- Checks if present
  - if it is - removes
  - if it is not, it adds

```
24  <p class="blue">Click to toggle</p>
25  <p class="blue highlight">highlight</p>
26  <p class="blue">on these</p>
27  <p class="blue">paragraphs</p>
28  <button>Toggle Highlight</button>
29  <script>
30  $( "button" ).click(function() {
31    $("p" ).toggleClass( "highlight" );
32  });
33  </script>
```

Click to toggle  
highlight  
on these  
paragraphs  
[Toggle Highlight](#)

Click to toggle  
highlight  
on these  
paragraphs  
[Toggle Highlight](#)

# CSS manipulation and more jQuery Selectors

Manipulating styles with `css()`



**More selectors and filters**

Traversing

Using method chaining

Using `jQuery.each()` and `.each()`

# Select for Attribute Values

- As with CSS3, use brackets `[attribute]`

```
$ ("a[title]")
```

selects all anchor elements that have a title attribute

```
$ ("a[title='MyTitle'])")
```

selects all anchor elements that have a "MyTitle", title attribute value

# Using Starts Within Selectors

- `[attribute^='attributeValue']`
  - selects all elements with an attribute that starts with a value:  
`$( "input[value^='Submit']" )`
- Selects any input element whose value attribute begins with "Submit"

```
<input type="button" value ="Submit Events"/> ←  
<input type="button" value ="Form Submit"/>  
<input type="button" value ="Submit Report"/> ←
```



# Using Contains in Selectors

- `:contains()` will select elements that match the contents within the contains exception:

```
$( "div:contains('Inc')")
```

Selects `divs` that contain the text "Inc" (case-sensitive)



```
<div>Food, INC.</div>
```

→ 

```
<div>Astoria, Illinois is Incorporated</div>
```

```
<div>Chrysler LLC</div>
```

→ 

```
<div>Apple, Inc.</div>
```

# Selecting Input Elements

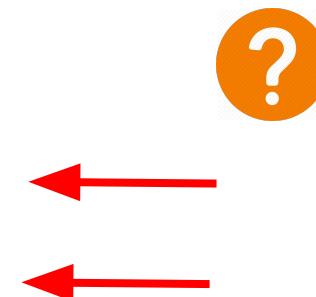
- `$(":input")` will select all input elements including:  
input, select, textarea, button, radio
- Can target specific attribute values:  
`$("input[type='radio'])")`  
Targets all radio buttons on the page

# Selecting Descendants

- The use of space indicates a descendant:
  - child, grandchild, great-great-great-grandchild etc.

`$( "div span" )` selects all `span` that are inside of a `div`

```
<div>
    <p><span>1st span</span></p>
    <span>2nd span</span>
</div>
```



# Selecting Immediate Children

- To reference immediate children of a parent use the > character:

```
$("div > span")
```

Selects all span elements that are **immediate** children of a div

```
<div>
  <p><span>1st span</span></p>
  <span>2nd span</span>
</div>
```



# Selecting the Adjacent Sibling

- `$( "span + input" )` selects an `input` element that immediately follows a `span`



```
<span>Name:</span>  
<input type="text" id="first"/> ←  
<input type="text" id="middle" />  
<p><input type="text" id="last" /></p>
```

# Selecting General Siblings

`$("#container ~ div").css('color', 'red');` selects all div sibling elements that follow the element with the id of #container



20

```
<div id="container">Some cool stuff</div>
```

21

```
<div>Twenty-one</div>
```

22

```
<p>Twenty-two</p>
```

23

```
<div>Twenty-three</div>
```

Some cool stuff  
Twenty-one

Twenty-two

Twenty-three

# Selecting Even or Odd Rows in a Table

- `$( "tr:odd" )` and `$( "tr:even" )` is the jQuery syntax for selecting odd or even rows
- The index is "0" based
  - Odd would return 1, 3, 5, 7, 9, ...
  - Even would return 0, 2, 4, 6, 8, ...

# Selecting the First Child, \$("element:first-child")

- Selects when matches the first child in an element group
- `$( "span:first-child" ).css( 'color', 'blue' )`



```
19①<body>
20②  <div>
21③    <p>
22④      <span>1st span in paragrpah</span>
23⑤    </p>
24⑥    <span>span that is 2nd child of div</span>
25⑦  </div>
26
27  <span>First span child of body, that is 2nd child<br/></span>
28  <span>Second span child of body, that is 3rd child</span>
29⑧  <div>
30    <span>Div Group 1, 1st child span</span>
31    <span>Div Group 1, 2nd child span</span>
32  </div>
33⑨  <div>
34    <span>Div Group 2, 1st child span</span>
35    <span>Div Group 2, 2nd child span</span>
36  </div>
37  <span>Sibling to divs and spans of body</span>
38  <p><span>First span in paragraph</span></p>
```

1st span in paragrpah

span that is 2nd child of div

First span child of body, that is 2nd child

Second span child of body, that is 3rd child

Div Group 1, 1st child span Div Group 1, 2nd child span

Div Group 2, 1st child span Div Group 2, 2nd child span

Sibling to divs and spans of body

First span in paragraph

# Using has() in a Selector `$( "parent:has(element) " )`

- Checks to see if a parent contains a certain descendant:

```
$( "li:has(a) " )
```

- Selects all li elements that have a specified element as a descendant:

```
<ul>
```

→ <li>first Child with <a href="#">Link</a></span>

```
<li>a second Child</li>
```

```
</ul>
```



# CSS manipulation and more jQuery Selectors

Manipulating styles with `css()`

More selectors and filters



**Traversing**

Using method chaining

Using `jQuery.each()` and `.each()`

# Traversing

- After an initial selection with jQuery, you can traverse the DOM
- jQuery has methods for traversing parents, children, and siblings
- Each method can optionally be passed string selectors
  - Some can take another jQuery object in order to filter the selection

# Finding the parents from a selection

- `$(".span.subchild").parent()` - returns element defined at line 13
- `$(".span.subchild").parents()` - lines 11,12,13
- `$(".span.subchild").parentsUntil( "div.grandparent" )` - lines 12,13

```
11      <div class="grandparent">
12          <div class="parent">
13              <div class="child">
14                  <span class="subchild"></span>
15          </div>
16      </div>
17      <div class="surrogateParent1"></div>
18      <div class="surrogateParent2"></div>
19  </div>
```

# Children selects direct > children, find goes deeper

- `$( "div.grandparent" ).children( "div" )` - lines 12, 17, 18
- `$( "div.grandparent" ).find( "div" );` lines 12,13,17,18

```
11    <div class="grandparent">
12        <div class="parent">
13            <div class="child">
14                <span class="subchild"></span>
15            </div>
16        </div>
17        <div class="surrogateParent1"></div>
18        <div class="surrogateParent2"></div>
19    </div>
```

# Siblings selects

- `$( "div.parent" ).next()` - line 17
- `$( "div.parent" ).prev()` - returns empty array [ ]
- `$( "div.parent" ).nextAll()` - lines 17,18
- `$( "div.surrogateParent1" ).siblings()` - lines 12, 18 (both directions)

```
11      <div class="grandparent">
12          <div class="parent">
13              <div class="child">
14                  <span class="subchild"></span>
15              </div>
16          </div>
17          <div class="surrogateParent1"></div>
18          <div class="surrogateParent2"></div>
19      </div>
```

# Selector Test Page

<http://codylindley.com/jqueryselectors>

The screenshot shows a web browser window titled "Selectors" displaying the URL <http://codylindley.com/jqueryselectors>. The page content is titled "jQuery Selectors".

On the left, there is a section for "Enter Your Own Selectors" with an example: `li:nth-child(2)`. Below it is a text input field containing `li:nth-child(2)` and a "toggle" button.

Under "Basics", there are several examples with "toggle" and "documentation" buttons:

- `$(‘code’)`
- `$(‘#myid’)`
- `$(‘.myclass’)`
- `$(‘*’)`
- `$(‘code’, #myid, .myclass’)`

Below these, under "Hierarchy", are examples:

- `$(‘div code’)`
- `$(‘li > ul’)`

On the right side of the page, there is a "Toggle Button" example with the following code:

```
Toggle Button = $(‘selector’).addClass(‘highlight’).animate({ marginLeft: 10 }, ‘fast’);  
Download This Lab
```

Below the code, there is a note: "This sentence is in <div id="myid">. It is followed by a horizontal rule." and a paragraph: "This is a paragraph, which means it is wrapped in <p> and </p>. Those "p" tags in the previous sentence are formatted as <code>.

A list follows:

- This is the first list item (<b>) in an unordered list (<ul>).
- This is the second list item. It has a [link](#) in it.
- This is the third list item. It has a class of "myclass otherclass"
- This is the fourth list item. It has **strong** text and **emphasized** text.
  - second-level list item 1
  - second-level list item 2

Below the list, there is another paragraph: "<p class="myclass">This is another paragraph. It has class="myclass" Otherwise, nothing **special** about it at all."

At the bottom, there are examples of form inputs:

- text input
- disabled text input
- select list w/ options
  - another option
  - a third option

# CSS manipulation and more jQuery Selectors

Manipulating styles with `css()`

More selectors and filters

Traversing



**Using method chaining**

Using `jQuery.each()` and `.each()`

# Understanding Control Chaining and Commands

- Commands can be chained together using the dot notation
- Browsers do not have to find the same element(s) more than once

```
$("p.answer").css('background', 'red').click(function () {  
    alert("added event");  
});
```

1. find paragraphs with class="answer"
2. change the background color to red
3. add a click event

# **CSS manipulation and more jQuery Selectors**

Manipulating styles with `css()`

More selectors and filters

Traversing

Using method chaining

➤ **Using jQuery.each() and .each()**

# The `.each()` command executes against all matches

- Iterate executing a function for each matched element

```
30  <div>Click a div</div>
31  <div style="color:■blue">to iterate through</div>
32  <div>swapping the current style</div>
33
34  <script>
35      $("div").click(function () {
36          $("div").each(function (i) {
37              if (this.style.color !== "blue") {
38                  this.style.color = "blue";
39              } else {
40                  this.style.color = "";
41              }
42          });
43      });
44  </script>
```

Click a div  
to iterate through  
swapping the current style

# The `$.each()` command works on arrays and objects

- jQuery provides a convenient way to work with arrays and objects

```
17  $.each([22, 10, 42, 19], function (index, value) {  
18      console.log(index + ": " + value);  
19  });  
20  
21  var band = {  
22      "name": "U2",  
23      "Country": "Ireland"  
24  };  
25  
26  $.each(band, function (key, value) {  
27      console.log(key + ": " + value);  
28  });
```



What will the output be?

0: 22
1: 10
2: 42
3: 19
name: U2
Country: Ireland

# Exercise: JQuery CSS Manipulation & Selectors

20 min

- In this lab, you will:
  - Use multiple CSS manipulations
  - Use various selection techniques
- Open \Labs\ch12-CSS\_jQuerySelectors\README.md
- Follow the directions in README.md to practice with selectors

# Chapter Summary

In this chapter we have begun:

- Further manipulating styles with `css()`
  - `css()/addClass()/hasClass()/removeClass()/toggleClass()`
- Utilizing more jQuery selectors
- Traversing selections
- Using method chaining
- Using `jQuery.each()` and `.each()`

# **Modern Web Development**

## **Chapter 13**

jQuery Event Model

# Chapter Objectives

In this chapter we will look at:

- How events play a critical role in client-side applications
- Working with the jQuery Event Model and event object properties
- Responding to browser resize and scroll events
- Handling Keyboard Events
- Event Handler and delegation of events with on, off, one and trigger
- Responding to form events using jQuery
- Working with mouse events

# jQuery Event Model



## **Attaching and manually invoking event handlers**

Working with Event Objects

Browser Resize Events

Handling Keyboard Events

On, Off, and Delegating Events

Working with forms

Handling Mouse Events

# Events without jQuery

- Events like button clicks are handled differently in different browsers

```
12  function assignEventListeners() {  
13      var button = document.getElementById ("testButton");  
14      if (button.addEventListener) { // all browsers except IE before version 9  
15          button.addEventListener ("mousedown", function () {buttonDownHandler(button)}, false);  
16          button.addEventListener ("mouseup", function () {buttonUpHandler (button)}, false);  
17      }  
18      else {  
19          if (button.attachEvent) { // IE before version 9  
20              button.attachEvent ("onmousedown", function () {buttonDownHandler(button)});  
21              button.attachEvent ("onmouseup", function () {buttonUpHandler (button)});  
22          }  
23      }  
24  }
```

- jQuery provides cross-browser support

# Basic Event Handling

- Events are assigned to elements
  - Using a jQuery select, a standard function name, and a callback
- Example:

```
$("#button1").click(function() {  
    alert('The id button1 was clicked');  
});
```

```
<button id="button1">Button1 - click </button>
```

# Common Events

- Browser events are assigned to items
  - Using a jQuery select, a standard function name, and a callback
  - See more at <http://api.jquery.com/category/events>

Mouse	Keyboard	Form	Browser
click	keypress	submit	resize
dblclick	keydown	change	scroll
mouseenter / mouseleave	keyup	focus	
mousedown / mouseup		blur	
contextmenu		select	

# Example of simple events

- Add handlers for each type of event

```
14  $("#button1").click(function() {  
15  |     alert('The id button1 was clicked');  
16  });  
17  
18  $("#button2").dblclick(function() {  
19  |     alert('The id button1 was double clicked');  
20  });  
21  
22  $("#button3").mouseenter(function() {  
23  |     alert('The id button3 was hovered');  
24  });
```

# Manually invoking events

- You can dynamically trigger an event by calling it without a handler

```
$ (“#target”) .event-name ()
```

- This is a shortcut for calling .trigger(“event-name”)

```
$ (“#target”) .trigger (“event-name”);
```



How to programmatically click a button with an id of submit?

```
$(“#submit”).click()
```

# jQuery Event Model

Attaching and manually invoking event handlers



## Working with Event Objects

Browser Resize Events

Handling Keyboard Events

On, Off, and Delegating Events

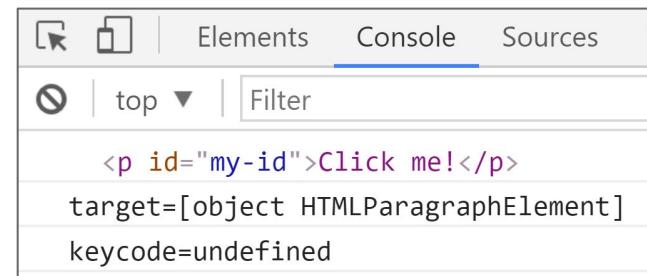
Working with forms

Handling Mouse Events

# Events have Properties

- Some events have properties associated with them
  - Access them via the function input parameter
- Some values may be undefined, depending on the event

```
73 $("p").click(function(e) {  
74     console.log(e.currentTarget);  
75     console.log('target=' + e.currentTarget);  
76     console.log('keyCode=' + e.keyCode);  
77 });
```



# Common Event Properties

- Properties are copied over and normalized from JavaScript event object
  - jQuery normalizes these properties for cross-browser consistency:
    - target, relatedTarget
    - pageX, pageY
    - which
    - metaKey
- Other properties:
  - altKey, bubbles, button, buttons, cancelable, char, charCode, clientX, clientY, ctrlKey, currentTarget, data, detail, eventPhase, key, keyCode, metaKey, offsetX, offsetY, originalTarget, pageX, pageY, relatedTarget, screenX, screenY, shiftKey, target, toElement, view, which*

Can access other properties not copied over:  
`var files = event.originalEvent.dataTransfer.files;`

# Some events carry location information

```
27  $("*").click(function(event){  
28      console.log("X=" + event.pageX +  
29          " Y=" + event.pageY +  
30          " target=" + event.currentTarget);
```

```
35 <body>  
36     <div id="test">  
37         click me  
38     </div>  
39     <div id="test2">  
40         test2  
41     </div>  
42     <div id="test3">  
43         This is test 3  
44     </div>  
45 </body>
```



What happens  
when line 37 is  
clicked?

# Event Bubbling

- In the previous example, the handler will be called 3 times
  - Bubbles up the hierarchy

X=60 Y=20 target=[object HTMLDivElement]

---

X=60 Y=20 target=[object HTMLBodyElement]

---

X=60 Y=20 target=[object HTMLHtmlElement]

---

# jQuery Event Model

Attaching and manually invoking event handlers

Working with Event Objects



## Browser Resize Events

Handling Keyboard Events

On, Off, and Delegating Events

Working with forms

Handling Mouse Events

# Browser events

- The two browser events you can respond to are:
  - `resize()`
  - `scroll()`
- Resize can be helpful to respond when the user resizes the window

```
15  <script>
16
17      $(window).resize(function () {
18          $("body").html("<div>" + $(window).width() + "px</div>");
19      );
20
21  </script>
```

# Using scroll()

- You can respond to scroll events on the window or other objects
  - window
  - elements with overflow: scroll

```
<div id="target" style="overflow: scroll; width: 400px; height: 100px;">
    Rich in mystery. Concept of the number one tesseract rings of Uranus rea
    Tesseract are creatures of the cosmos colonies bits of moving fluff. Con
</div>
```

28	\$(window).scroll(function () {
29	//Do something here, such as load more data
30	});
31	
32	\$( "#target" ).scroll(function() {
33	console.log( "<div>Handler for .scroll() called.</div>" );
34	});

# jQuery Event Model

Attaching and manually invoking event handlers

Working with Event Objects

Browser Resize Events



## Handling Keyboard Events

On, Off, and Delegating Events

Working with forms

Handling Mouse Events

# Handling Keyup / Keydown Events

- Keyup/Keydown events are sent when the user interacts with the keyboard
- Can be attached to any element
- Event is sent to the element that has the focus - usually a form element

Name: <input id="name-text-box" type="text" /> <br />

- This logs for each key pressed in the input box. which gives the keycode
  - <http://www.asciiitable.com/>

```
21 //keyup
22 $("#name-text-box").keyup(function(e) {
23     console.log('You pressed ' + e.which);
24});
```

# Using keypress()

- Keypress event is sent to element when browser registers keyboard input
  - Shift, Esc, and delete trigger keydown but **not** keypress
- Keydown and keyup provide a code indicating which **key** is pressed
  - **keypress** indicates which **character** was entered

KeyDown Event typing:  .keydown() called event.which=65

KeyPress Event typing:  .keypress() called event.which=97

# Implementing global shortcuts

- Can attach key interactions to the document object
- Because of event bubbling, all key presses will make their way up the DOM to the document object unless explicitly stopped

# jQuery Event Model

Attaching and manually invoking event handlers

Working with Event Objects

Browser Resize Events

Handling Keyboard Events

## ➤ **On, Off, and Delegating Events**

Working with forms

Handling Mouse Events

## Using on()

- Using on gives more flexibility when assigning events to elements
- Syntax: `on(eventType, handler(eventObject))`

```
$('#email-text-box').on('click', function() {  
    console.log('You clicked in the email textfield');  
});
```

- This example of on() is now the default behavior of events
  - Ex. `$(“button”).click(handler) == $(“button”).on(‘click’,handler)`

# Disable all or some events using off()

- off() is used to remove a handler previously bound to an element:
- Specific events can also be targeted using off('event-name'):

```
20  $("#buttonOn").click(function() {  
21      alert('adding a click and a mouseover');  
22      $("p").on("click",function() {  
23          alert('The button added this click');  
24      })  
25      $("p").on("mouseover",function() {  
26          alert('The button added this hover');  
27      })  
28  });
```

```
30  $("#buttonOff").click(function() {  
31      $("p").off("mouseover");  
32  })
```

# Using on() to Bind Multiple Events

- on() allows multiple events to be bound to one or more elements
- Use a space to separate events

```
$("#email-text-box").on('cut copy paste', function(e) {  
    e.preventDefault();  
    alert('Cut/Copy/Paste is disabled for this textbox.');//  
});
```

**preventDefault()** allows you to stop the normal behavior from happening. contextmenu for example can allow you to override what happens with right-click

# Using on() allows delegation

- Events are ONLY assigned to existing elements
- If planning on adding elements later, can use **delegation**
  - Pass an argument which is a descendant selector

```
28  $(document).ready(function() {  
29      $("table").on("click","tr", function(e){  
30          alert(e.target.innerHTML);  
31      });  
32  });
```

# DEMO: Using Delegation

5 min

- Open \Demos\Chapter13-jQueryEvents\table-add-row.html
  - In browser
  - In editor
- Use the buttons to add rows to the tables
- Click the first rows and newly added rows
- Examine the code to see the use of on, to allow click behavior for future rows

# jQuery Event Model

Attaching and manually invoking event handlers

Working with Event Objects

Browser Resize Events

Handling Keyboard Events

On, Off, and Delegating Events



**Working with forms**

Handling Mouse Events

# Using val() to get the current values

- You can use val() to :
  - get the **current** value of the first element in the set
  - set the value of every matched element
- When called on an empty collection, it returns undefined

```
26 <div>
27   Enter Text: <input type ="text">
28   <output id="textOutput"></output>
29 </div>
```

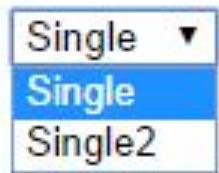


how to get val and display in output as the user types?

```
63 $("input[type=text]").keyup(function() {
64   var textMessage =
65     $("input[type=text]").val() ;
66   $(this).next().html(textMessage);
67 });
```

# Using val() with single select

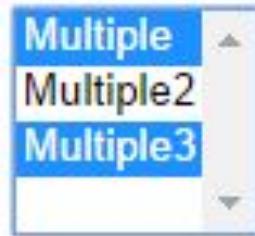
- You can use :checked to select the right elements  

```
69  $("#singleSelect").change(function() {  
70    var singleMessage =  
71      // $("select#singleSelect option:checked").val() ;  
72    $(this).val();  
73    $("#singleOutput").html(singleMessage);  
74  });
```

# Using val() with a multiple select

- On HTML multiple selects, an array is returned



```
37 <div>
38   <select id="multipleSelect" multiple="multiple">
39     <option selected="selected">Multiple</option>
40     <option>Multiple2</option>
41     <option selected="selected">Multiple3</option>
42   </select>
43   <output id="multipleOutput"></output>
44 </div>

78 $("#multipleSelect").change(function() {
79   var multipleValues = $( "#multipleSelect" ).val() || [];
80   var multipleMessage = multipleValues.join( ", " ) ;
81   $("#multipleOutput").html(multipleMessage);
82 } );
```

# Using val with radio buttons

- We can use \$(this) or :checked to get the val for radio buttons

```
53 <fieldset>
54   <input type="radio" name="yesnoradio" value="Yes"> Yes
55   <input type="radio" name="yesnoradio" value="No"> No
56 </fieldset>
57 <output id="radioOutput"></output>
```



```
84 $('input[type=radio][name=yesnoradio]').change(function() {
85   var radioMessage = $(this).val() ;
86   // $("input[type=radio][name=yesnoradio]:checked").val() ;
87   $("#radioOutput").html(radioMessage);
88 });
```

Preferred Contact time(s)  
 A.M.  P.M.

# Using val() with checkboxes

- Checkbox values are arrays
- We can use a map() function to work on the returned array

```
46  <fieldset><label>Preferred Contact time(s)<br/>
47      <input type="checkbox" name="checkboxname" value="A.M."> A.M.
48      <input type="checkbox" name="checkboxname" value="P.M."> P.M.
49      </label>
50  </fieldset>
92  $("input[type=checkbox][name=checkboxname]").change(function() {
93      var checkedValues = $('input:checkbox:checked').map(function() {
94          return this.value;
95      }).get().join();
96
97      $("#checkOutput").html(checkedValues);
98  });
```

# Setting Form values

- We can make use of val() to set up form data to be edited

```
62 <form name="editForm">
63   Full Name: <input type ="text" name="fullname"> <br />
64
65   <select id="multipleTimesSelect" multiple="multiple">
66     <option>10am - 11am</option>
67     <option>11:30am - 12:00pm</option>
68     <option>1pm - 3pm</option>
69   </select>
70 </form>
```



What will the jQuery code look like to populate with existing data?

# Using val() to populate a form

- Both single and multiple values can be passed in

```
110 $("input[name='fullname']").val('Cody Ocean');  
111  
112 $("#multipleTimesSelect").val([ "10am - 11am", "1pm - 3pm" ]);
```

# jQuery Event Model

Attaching and manually invoking event handlers

Working with Event Objects

Browser Resize Events

Handling Keyboard Events

On, Off, and Delegating Events

Working with forms



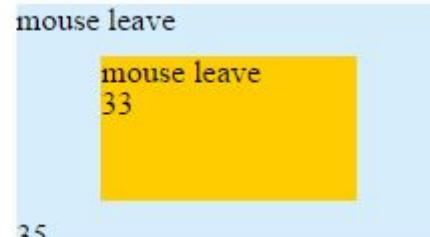
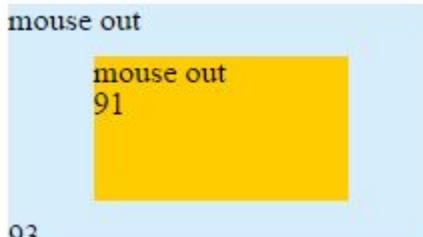
**Handling Mouse Events**

# Using `mouseover()` and `mouseenter()`

- **`mouseover`** fires when the mouse pointer enters the element
  - bubbles
- **`mouseenter`** JS event is proprietary to IE but jQuery simulates this event for all browsers
  - does not bubble, not triggered by descendants

# Using mouseout() and mouseleave()

- **mouseout** fires when the pointer moves out of the child element as well
- **mouseleave** fires only when the pointer moves out of the bound element
- **mouseleave JS event is proprietary to IE but jQuery simulates this event for all browsers**



```
42 <script>
43 var i = 0;
44 $( "div.overout" )
45   .mouseout(function() {
46     $( "p:first", this ).text( "mouse out" );
47     $( "p:last", this ).text( ++i );
48   })
49   .mouseover(function() {
50     $( "p:first", this ).text( "mouse over" );
51   });
52
53 var n = 0;
54 $( "div.enterleave" )
55   .on( "mouseenter", function() {
56     $( "p:first", this ).text( "mouse enter" );
57   })
58   .on( "mouseleave", function() {
59     $( "p:first", this ).text( "mouse leave" );
60     $( "p:last", this ).text( ++n );
61   });
62 </script>
```

# Using hover()

- You can pass 1 or 2 functions to hover
- Two functions can be passed for the **mouseenter** and the **mouseleave**
- If one function is passed, it fires for both events

```
57  $("#hover-here").hover(  
58      |     function(){  
59          |         $(this).css("background-color", "#00FF99");  
60          |     },  
61          |     function(){  
62              |         $(this).css("background-color", "#FFFFFF");  
63          |     }  
64      );
```

# Using a toggleClass with one hover function

- jQuery's has functions to toggle state
  - This is similar to the hover example with two functions

```
7 | .highlight { background-color: yellow }
```

```
66 | $("#toggle-hover").hover(function() {  
67 |   | $(this).toggleClass('highlight');  
68 | });
```

# Right-click events

- You can look for a right click event with contextmenu
- To disable normal behavior, use preventDefault()

```
15    $("p.question1").contextmenu(function (e) {  
16        e.preventDefault();  
17        $("p.answer1").hide();  
18    });
```

# Exercise: JQuery Event Properties

10 min

- In this lab, you will:
  - Handle browser events using jQuery
- Open \Labs\ch13-EventModel\README.md
- Follow the directions to practice with events

# Chapter Summary

In this chapter we have looked at:

- How events play a critical role in client-side applications
- Working with the jQuery Event Model and event object properties
- Responding to browser resize and scroll events
- Handling Keyboard Events
- Event Handler and delegation of events with on, off, one and trigger
- Responding to form events using jQuery
- Working with mouse events

# **Modern Web Development**

## **Chapter 14**

jQuery Manipulating the DOM

# Chapter Objectives

In this chapter we will see that:

- jQuery provides multiple functions for DOM manipulation:
  - `html()/text()`
  - `attr()`
  - `append()/prepend()`
  - `appendTo()/prependTo()`

# **jQuery Manipulating the DOM**



## **Using html() and text()**

Modifying Attributes and Properties

Adding and Removing Nodes

# Getting Content with html()

- Using html() returns the HTML contents of the first element it matches

```
console.log($(".table").html());
```

If more than one table, return their table contents, including html tags

```
88① <table id="table1">
89②   <tr>
90     <td>Table 1 - Cell One</td>
91     <td>Table 1 - Cell Two</td>
92   </tr>
93 </table>
94
95① <table id="table2">
96②   <tr>
97     <td>Table 2 - Cell One</td>
98     <td>Table 2 - Cell Two</td>
99   </tr>
100 </table>
```



```
<tbody><tr>
  <td>Table 1 - Cell One</td>
  <td>Table 1 - Cell Two</td>
</tr>
</tbody>
```

# Setting Content with html()

- Alternative to .innerHTML in JavaScript
- Can be used to load a target area of screen
  - Such as with receiving AJAX data

```
$("div#output").html("<p>Inserted Paragraph</p>");
```

# Passing Functions into html()

- By passing a function, a custom message could be set for each match
  - using index
  - using information for the current match \$(this)

```
$("div.button-count").html(  
    function(i) {  
        var buttonCountMessage = '<span>'  
            + $("button").length  
            + ' buttons</span>';  
  
        return '<p>There are ' + buttonCountMessage  
            + ' on this page - at index ' + i  
            + ' </p>';  
    });
```

# Using `text()` instead of `html()`

- `text()` function gathers the combined text contents of **each** element in the set of matched elements, **including** descendants
- `text(textString)` sets the content of each element in the set of matched elements to the specified text
- `text` can be used in XML docs

# Using text() to get DOM contents

- The example from before changes when we use text()

```
console.log($(".table").text());
```

```
88<table id="table1">
89    <tr>
90        <td>Table 1 - Cell One</td>
91        <td>Table 1 - Cell Two</td>
92    </tr>
93</table>
94
95<table id="table2">
96    <tr>
97        <td>Table 2 - Cell One</td>
98        <td>Table 2 - Cell Two</td>
99    </tr>
100</table>
```



```
Table 1 - Cell One
Table 1 - Cell Two
```

```
Table 2 - Cell One
Table 2 - Cell Two
```

# Setting Text with the `text()` Function

- Update the text between the chosen tags:

```
<div class="album-info">  
    Buckethead has released <span id="album-count">0</span> albums  
</div>
```

- You can update this count like so:

```
$("#album-count").text('264');
```

Buckethead has released 264 albums

# jQuery Manipulating the DOM

Using html() and text()



**Modifying Attributes and Properties**

Adding and Removing Nodes

# Using Dot Notation to Modify Properties

- Each button is changed to have a pop up

```
$('button').each(function(i) {  
    this.title= 'This button is at index = ' + i;  
});
```

Turn off All for button 1

This button is at index = 2

# Object attributes can be accessed using attr()

- The `.attr()` method gets the first matched element's attribute value
  - To get all matched elements individually, use a looping construct such as jQuery's `.each()`

```
var val = $('#email-text-box').attr('value');
console.log(val);
```

# Setting Attributes

- Object attributes can be updated using attr()
  - Use a map\object literal to set multiple items

```
$('input').attr('title', 'please complete this field');

$('input').attr({
    title: 'please complete this',
    value: 'incomplete'
});
```

# Why use attr()?

- **Convenience:** can call directly on jQuery object and use chaining
- **Cross-browser consistency:**
  - some attribute values are reported inconsistently across browsers
  - even across versions of a single browser
  - The `.attr()` method reduces inconsistencies

## When not to use .attr()

- The .attr() method returns undefined for attributes that have not been set
- To retrieve and change properties such as the checked, selected, or disabled state of form elements, use the .prop() method

# Difference between attributes and properties

- attr() just looks for the existence of the checked attribute
- to determine if a checkbox is checked - use the property:

```
27 $("input")
28     .change(function () {
29         var $input = $(this);
30         $("p").html(".attr( 'checked' ): <b>" + $input.attr("checked") + "</b><br>" +
31             ".prop( 'checked' ): <b>" + $input.prop("checked") + "</b><br>" +
32             ".is( ':checked' ): <b>" + $input.is(":checked") + "</b>");
33     })
34     .change();
```



Check me

.attr( 'checked' ): **checked**  
.prop( 'checked' ): **false**  
.is( ':checked' ): **false**

# **jQuery Manipulating the DOM**

Using `html()` and `text()`

Modifying Attributes and Properties



**Adding and Removing Nodes**

# Adding and Removing Nodes

- You can insert nodes into elements using  
`append()` and `appendTo()`  
`prepend()` and `prependTo()`
- The pairs have similar results, but are used differently
- To remove nodes from an element use `remove()`

# Appending to Nodes

- Appending inserts just before the closing tag of the matching element:

```
1 <h2>Greetings</h2>
2 <div class="container">
3   <div class="inner">Hello</div>
4   <div class="inner">Goodbye</div>
5 </div>
```

```
$( ".inner" ).append( "<p>Test</p>" );
```

OR

```
$( "<p>Test</p>" ).appendTo( ".inner" );
```

If using appendTo - content must be enclosed in html tags such as <p> or <span>

```
1 <h2>Greetings</h2>
2 <div class="container">
3   <div class="inner">
4     Hello
5     <p>Test</p>
6   </div>
7   <div class="inner">
8     Goodbye
9     <p>Test</p>
10    </div>
11 </div>
```

# AppendTo will “move” content”

- Can use a select on left with appendTo

```
15  <div id="source">
16      <p>This is some source code</p>
17  </div>
18
19  <div id="target">
20      <p>Hello</p>
21      <p>Goodbye</p>
22  </div>
23
24  <script>
25      $("body").click( function() {
26          $("#source").appendTo("#target");
27      });
28  </script>
```



how will this render  
after a click event?

Hello

Goodbye

This is some source code

```
<div id="target">
<p>Hello</p>
<p>Goodbye</p>
<div id="source">
<p>This is some source code</p>
</div>
</div>
```

# Prepending to Nodes

- Prepending inserts just after the opening tag of the matching element:

```
1 <h2>Greetings</h2>
2 <div class="container">
3   <div class="inner">Hello</div>
4   <div class="inner">Goodbye</div>
5 </div>
```

```
$( ".inner" ).prepend( "<p>Test</p>" );
OR
$( "<p>Test</p>" ).prependTo( ".inner" );
```

If using prependTo - content must be enclosed in html tags such as <p> or <span>

```
1 <h2>Greetings</h2>
2 <div class="container">
3   <div class="inner">
4     <p>Test</p>
5     Hello
6   </div>
7   <div class="inner">
8     <p>Test</p>
9     Goodbye
10  </div>
11 </div>
```

# Removing Nodes

- `remove()` will remove any matched element from the DOM

```
<div class="container">
  <div class="hello">Hello</div>
  <div class="goodbye">Goodbye</div>
</div>
```

```
$( ".hello" ).remove();
```

```
<div class="container">
  <div class="goodbye">Goodbye</div>
</div>
```

# Exercise: Manipulating the DOM

15 min

- In this exercise, you will practice manipulating the DOM with jQuery
- Follow the directions in \Labs\ch14-jQueryDOM\README.md

# Chapter Summary

In this chapter we have seen that:

- jQuery provides multiple functions for DOM manipulation:
  - `html()/text()`
  - `attr()`
  - `append()/prepend()`
  - `appendTo()/prependTo()`

# **Modern Web Development**

## **Chapter 15**

jQuery Animations and Effects

# Chapter Objectives

In this chapter we will:

- Explore several built-in animations in jQuery:
  - Show/Hide different elements
  - Fade-in/Fade-out
  - Slide up/Slide down
  - Bonus: Custom Animations

# jQuery Effects



## **Showing and Hiding elements**

Toggling elements

Fading in and out

Sliding up and down

Bonus: Custom Animations

# jQuery Effects

- The jQuery library provides techniques for adding animation
  - Animations and effects work across multiple browsers
- Simple, standard animations
  - Show/Hide/Toggle
  - SlideUp/SlideDown/SlideToggle
  - FadeIn/FadeOut/FadeTo
- The ability to create custom effects

# Showing Elements

- Here the paragraph is hidden with CSS

```
<button id="q1">What is 42 multiplied by 42?</button>
<p id="answer1" style="display:none"></p>
```

- Using `show()` we can bring the answer in to view:

```
$("#button#q1").click(function() {
    $("#p#answer1").text(42*42).show();
});
```

If using !important in your styles, such as `display: none !important`, it is necessary to override the style using `.css("display", "block !important")` should you wish for `.show()` to function correctly

# Hiding Elements

- The `hide()` function allows objects to be hidden

```
$("button#q1").dblclick(function() {  
    $("p#answer1").hide();  
});
```

- By default, immediately hidden, with no animation

```
$("p#answer2").hide("slow");
```

- When providing a parameter, the hide is an animation
- Using `slow` animates a fade over 600 milliseconds
- Duration can be "fast", "slow", or a number (milliseconds)

# Toggling elements with toggle()

- Using `toggle()` alternates between show/hide

```
$("button#toggle-button").click(function() {  
    $("div#toggle-div").toggle();  
});
```

- Can have multiple functions defined for the toggle

# jQuery Effects

Showing and Hiding elements



**Toggling elements**

Fading in and out

Sliding up and down

Bonus: Custom Animations

## toggle() Function

- The `toggle()` function alternates between show and hide

```
<div id="toggle-album-cover">Toggle Album Cover</div>

```

```
$( "#toggle-album-cover" ).click(function() {
    $( "#album-cover" ).toggle( "slow", function() {
        // Animation complete.
    });
});
```

# jQuery Effects

Showing and Hiding elements

Toggling elements



**Fading in and out**

Sliding up and down

Bonus: Custom Animations

# FadeIn / FadeOut uses opacity level

- The fade in/out functions unhides/hides the element using opacity

```
<div id="fade-in-cover">Fade In Album Cover</div>
<div style="display:none" id="fade-out-cover">Fade Out Album Cover</div>

```

```
$( "#fade-in-cover" ).click(function() {
    $( "#album-cover" ).fadeIn( "slow", function() {
        $( "#fade-in-cover" ).hide();
        $( "#fade-out-cover" ).show();
    });
});

$( "#fade-out-cover" ).click(function() {
    $( "#album-cover" ).fadeOut( "slow", function() {
        $( "#fade-out-cover" ).hide();
        $( "#fade-in-cover" ).show();
    });
});
```

# jQuery Effects

Showing and Hiding elements

Toggling elements

Fading in and out



**Sliding up and down**

Bonus: Custom Animations

# The sliding functions

- These hide and show by animating the height of elements in a page
- They accept a duration and optional callback

```
$(".slideDownbox").click(function () {  
    $(this).hide().slideDown('slow');  
});  
  
$(".slideUpbox").click(function () {  
    $(this).slideUp(2000);  
});  
  
$("#slideToggle").click(function () {  
    $('.slideTogglebox').slideToggle();  
});
```

# Exercise: JQuery Animations

20 min

- In this exercise you will create a quiz page with jQuery animation effects
- Follow the directions in \Labs\ch15-jQueryAnimationsEffects\README.md

## Example Quiz Reveal Using jQuery Animations and Effects

Question 1

In what year did the Beatles first appear on the Ed Sullivan Show?

February 9th, 1964

Question 2

Who is the fastest guitar player?

# Chapter Summary

In this chapter we have:

- Explored several built-in animations in jQuery:
  - Show/Hide different elements
  - Fade-in/Fade-out
  - Slide up/Slide down
  - Bonus: Custom Animations

# jQuery Effects

Showing and Hiding elements

Toggling elements

Fading in and out

Sliding up and down



**Bonus: Custom Animations**

# Custom Animations

- Custom methods allow animations to be run
  - `animate()`  
create animation effects on any numeric CSS property
  - `queue()`  
Set or show a list of animation functions to be executed
  - `dequeue()`  
Fire the next function in queue for matched elements
  - `delay()`  
Fire a timer to delay execution of subsequent items in the queue
  - `jQuery.fx.off`  
Globally disable all animations
  - `stop()`  
Stop the currently-running animation of matched elements

# Custom Animations

- Animate takes element from current state to a new state based upon properties

```
$("button").click(function(){
    $("div").animate({
        height: '50px',
        width: '70px',
        left: '250px',
        opacity: '0.75'
    });
});
```

# Modern Web Development

## Chapter 16

jQuery AJAX Features



# Chapter Objectives

In this chapter we will see that:

- jQuery simplifies the process of making Ajax calls
- jQuery provides convenient functions of `load()`, `get()`, and `post()`
- The `ajax()` function provides complete control over Ajax calls

# jQuery AJAX Features



## Simplifying AJAX with jQuery

- Using load() function

- Making GET Requests

- Making POST Requests

- Setting default options for ajax()

# jQuery AJAX Features

- jQuery simplifies working with AJAX

```
var xmlhttp = new XMLHttpRequest();
var url = "data/myBands.json";

xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
    var myArr = JSON.parse(xmlhttp.responseText);
    myFunction(myArr);
}
};

xmlhttp.open("GET", url, true);
xmlhttp.send();
```

- Default type = GET
- parses data

```
15  $.ajax({
16      url: "data/myBands.json",
17      dataType: 'JSON',
18  })
19  .done(function (data) { doStuff(data); })
20  .fail(function (err) { console.log(err); })
21});
```

# jQuery AJAX Features

- Load JSON, XML, HTML or even scripts
- Cross-Browser Support
- Simple API
  - `ajax()` provides coarse control
  - `get()`, `post()`, `getJSON()` provide conveniences

# Using AJAX with jQuery utility functions

- A jQuery utility function simplifies common tasks
  - cross browser support
- Get a JSON file with `$.getJSON()`
- Get a script file with `$.getScript()`
- Process any type of file with `$.get()` or `$.ajax()`

## Example: `$.getJSON()`

- `$.getJSON()` retrieves a JSON file
- Automatically parses the JSON data

```
$.getJSON("data/myBands.json",
          function(parsedData) {
            myFunction(parsedData);
          }
);
```

## Example: `$.getScript()`

- `$.getScript()` loads and executes a script file

```
$("#execute-script").click(function(){
    $.getScript("getScriptExample.js");
});
```

## Example: `$.get()`

- `$.get()` takes a file and a callback function as arguments
  - Works like `$.getJSON()` but with any type of file

```
$.get( "snippet.html", function( data ) {
    $( "#result" ).html( data );
    alert( "Load was performed." );
});
```

# jQuery AJAX Features

Simplifying AJAX with jQuery



**Using load() function**

Making GET Requests

Making POST Requests

Setting default options for ajax()

## Using load()

- Load directly into an element using load()
  - All or a portion of the document

```
$("#all").load( "snippet2.html");
```

```
$("#just-one-id").load( "snippet2.html #paragraph2");
```

# jQuery AJAX Features

Simplifying AJAX with jQuery

Using load() function



**Making GET Requests**

Making POST Requests

Setting default options for ajax()

# Making GET requests

- Settings include: url, data, callback, datatype
- This example retrieves JSON data

```
var jqxhr = $.get( "http://your-website.come/api/bands/1", function() {
  alert( "success" );
})
.done(function() {
  alert( "second message after success" );
})
.fail(function() {
  alert( "error" );
})
.always(function() {
  alert( "Im always called, failure or not" );
});
```

.done and .fail replace success and error  
(which have been deprecated)

# jQuery AJAX Features

Simplifying AJAX with jQuery

Using load() function

Making GET Requests



**Making POST Requests**

Setting default options for ajax()

# Making POST calls

- Settings include: url, data, callback, datatype
- This example adds a band to server

```
var jqxhr = $.post( "http://localhost:3000/api/bands", bandData)
    .done(function(data) {
        alert( "Data Loaded: " + data );
    })
    .fail(function() {
        console.log( "error, data= " + data );
    })
    .always(function() {
        alert( "finished" );
    });
});
```

# jQuery AJAX Features

Simplifying AJAX with jQuery

Using load() function

Making GET Requests

Making POST Requests



**Setting default options for ajax()**

# Setting options with `$.ajax()`

- When making multiple ajax calls, can set reusable options
- Can be set in `$.ajaxSetup()` for reusability
- Options are used, and can be changed when making calls

```
71  $.ajaxSetup({
72    url: '/band',
73    type: 'POST',
74    dataType: 'json'
75  });
76
77
78  $.ajax({
79    type: 'GET', //can reset options in here
80    success: function (data) {
81      $('div').html(data);
82    }
83  });
```

# **Exercise: JQuery AJAX**

- In this exercise you will:
  - start a Node server that allows data to be requested
  - write jQuery code to request data asynchronously
- Follow the directions in \Labs\ch16-jQueryAJAX\README.md

# Chapter Summary

In this chapter we have seen that:

- jQuery simplifies the process of making Ajax calls
- jQuery provides convenient functions of `load()`, `get()`, and `post()`
- The `ajax()` function provides complete control over Ajax calls

# Modern Web Development

## Chapter 17

### jQuery Plugins

Show 10 entries Search:

Name	Position	Office	Age	Start date	Salary
Airi Satou	Accountant	Tokyo	33	2008/11/28	\$162,700
Angelica Ramos	Chief Executive Officer (CEO)	London	47	2009/10/09	\$1,200,000
Ashton Cox	Junior Technical Author	San Francisco	66	2009/01/12	\$86,000
Bradley Greer	Software Engineer	London	41	2012/10/13	\$132,000
Brenden Wagner	Software Engineer	San Francisco	28	2011/06/07	\$206,850
Brielle Williamson	Integration Specialist	New York	61	2012/12/02	\$372,000
Bruno Nash	Software Engineer	London	38	2011/05/03	\$163,500
Caesar Vance	Pre-Sales Support	New York	21	2011/12/12	\$106,450
Cara Stevens	Sales Assistant	New York	46	2011/12/06	\$145,600
Cedric Kelly	Senior Javascript Developer	Edinburgh	22	2012/03/29	\$433,060

Showing 1 to 10 of 57 entries Previous [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) Next

1 - Delivery Location    2 - Order Items    3 - Payment Info

**Payment Information**

Please enter payment information.

Card Type: American Express  
Card Number: 123456789011234  
Expiration Date: Month: 12 Year: 2014  
Card Information  
Name on Card: Wesley Snipes  
Security Code: 123  
Zip Code: 90210

**Processing your order...**

**Place Order**

# Chapter Objectives

In this chapter we will see that:

- Productivity can be enhanced through plugins
- Plugins such as DataTables make working with tabular data simpler
- Bonus: Custom plugins can be written using `$.fn.yourPlugin`

# jQuery Plugins



## **Introduction to jQuery Plugins**

Chapter Summary

Bonus: Custom Plugin Concepts

# jQuery Plugins

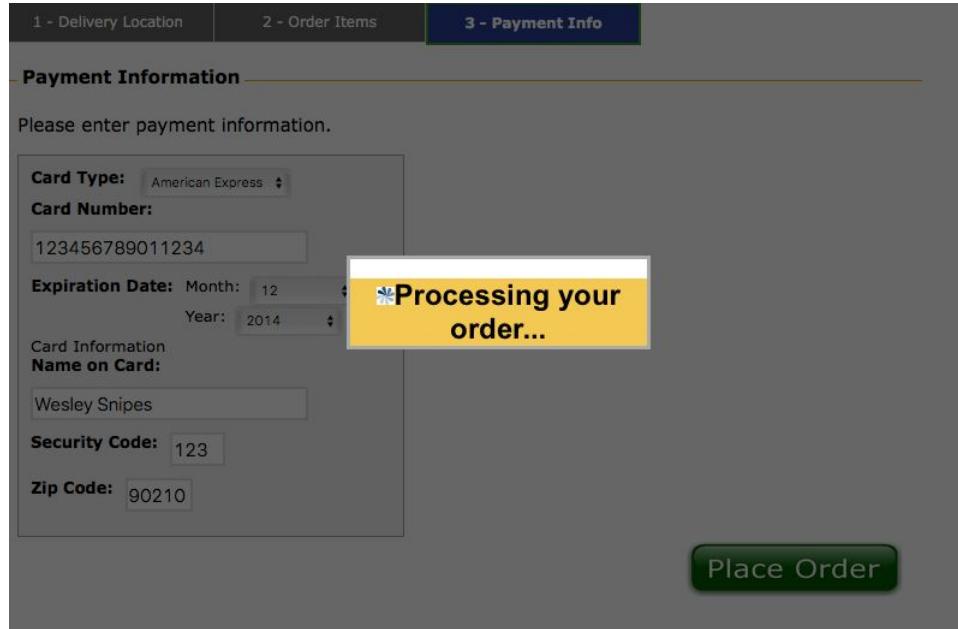
- There is a large community based around developing jQuery plugins
- The site plugin.jquery.com now recommends using node to find packages  
<https://www.npmjs.com/browse/keyword/jquery-plugin>
- Many categories are available, and most have good online demos/tutorials

# Examples of jQuery Plugins

- **BlockUI** – Prevent users from clicking during an operation
- **MaskedInput** – Provides a mask for textboxes
- **DataTables** – Sorting, filtering, and paging functionality for tabular data

# The BlockUI Plugin

- BlockUI creates a modal window



# Using the BlockUI Plugin

- Can create a custom message using the BlockUI plugin

```
$('#PlaceOrderButton').click(function () {  
  
    //Add block UI code here  
    $.blockUI({  
        message: '<h1>' +  
                 'Processing your order...</h1>'  
    });  
  
  
    var params = $('form').serialize();  
    //Simulate the order being placed  
    window.setTimeout(function () {  
        placeOrderComplete({ "Status": true, "Errors": [] });  
    }, 2000);  
  
});
```

# Using the Masked Input Plugin

- The Masked Input plugin restricts input to textboxes
- Custom masks can be supplied:

Date

99/99/9999

Phone

(999) 999-9999

Tax ID

99-99999999

SSN

999-99-9999

# Using the Masked Input Plugin

- After including the script:

```
<script src="jquery.maskedinput.js" ></script>
```

- The mask function can be called on an input using a pattern

```
$ ("#ssn") .mask("999-99-9999") ;  
$ ("#date") .mask("99/99/9999") ;  
$ ("#phone") .mask("(999) 999-9999") ;  
$ ("#ein") .mask("99-9999999") ;
```

# jQuery Data Plugins

- Several plugins exist for working with tabular data:
  - DataTables
  - FlexiGrid
  - InGrid
  - jqGrid
  - TableSorter

Show 10 entries						Search:
Rendering engine	Browser	Platform(s)	Engine version	CSS grade		
Gecko	Firefox 1.0	Win 98+ / OSX.2+	1.7	A		
Gecko	Firefox 1.5	Win 98+ / OSX.2+	1.8	A		
Gecko	Firefox 2.0	Win 98+ / OSX.2+	1.8	A		
Gecko	Firefox 3.0	Win 2k+ / OSX.3+	1.9	A		
Gecko	Camino 1.0	OSX.2+	1.8	A		
Gecko	Camino 1.5	OSX.3+	1.8	A		
Gecko	Netscape 7.2	Win 95+ / Mac OS 8.6-9.2	1.7	A		
Gecko	Netscape Browser 8	Win 98SE+	1.7	A		
Gecko	Netscape Navigator 9	Win 98+ / OSX.2+	1.8	A		
Gecko	Mozilla 1.0	Win 95+ / OSX.1+	1	A		
Showing 1 to 10 of 58 entries						<a href="#">First</a> <a href="#">Previous</a> <a href="#">1</a> <a href="#">2</a> <a href="#">3</a> <a href="#">4</a> <a href="#">5</a> <a href="#">Next</a> <a href="#">Last</a>

# Using the DataTables Plugin

- Info for this plugin is found here: <https://datatables.net/>
- Data support for DOM, Javascript, Ajax
- Easy to use

```
$ (document) .ready(function () {  
    $ ('#myTable') .DataTable ();  
} );
```

# Exercise: JQuery Plugins

15 min

- In this lab you will work with jQuery Plugins
- Follow the directions in /Labs/ch17-jQueryUI\_Plugins/Plugins/README.md

# jQuery Plugins

Introduction to jQuery Plugins



## Chapter Summary

Bonus: Custom Plugin Concepts

# Chapter Summary

In this chapter we have seen that:

- Productivity can be enhanced through plugins
- Plugins such as DataTables make working with tabular data simpler
- Bonus: Custom plugins can be written using `$.fn.yourPlugin`

# jQuery Plugins

Introduction to jQuery Plugins

Chapter Summary



**Bonus: Custom Plugin Concepts**

# What does plugin code look like?

- Most plugins still follow older conventions
  - Code is contained within a closure, and IIFE
    - Immediately invoked function expression
  - Takes care of namespace issues
    - functions you declare will only exist for the plugin

```
24      (function ($) {  
25          'use strict';  
26  
27          //code goes here  
28  
29      })(jQuery);
```

# Naming a Plugin

- Avoid random names that don't explain the plugin's function
  - Made up fancy names are reserved for things such as frameworks
- Avoiding vagueness
  - If building a tooltip plugin, call it tooltip rather than tip
- Employ unique names to avoid collisions
  - Rather than tooltip, use a name like wTooltip

# Create the plugin function

- Extend the jQuery \$.fn object
- This will make our plugin available to any jQuery elements



Why is **return this** important?

- initialize the plugin on any element using the plugin name

```
23 <script>
24   (function ($) {
25     'use strict';
26
27     // Plugin class and prototype will go here.
28
29     $.fn.wTooltip = function () {
30       $('body').append('hello tooltip');
31
32       return this;
33     };
34
35     })(jQuery);
36   </script>
37 </head>
38 <body>
39   <script>
40     $('body').wTooltip().css('color', 'red');
41   </script>
42 
```

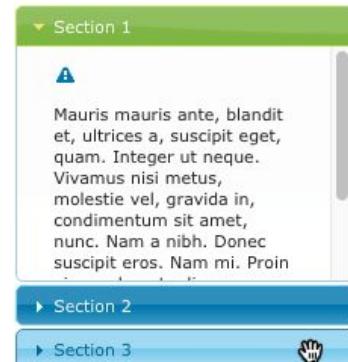
# Three approaches to theming jQuery UI plugins

- **Use the downloaded ThemeRoller theme:** (easiest way)
  - Generate and download a theme
  - App creates a new jquery-ui.theme.css file and images directory
- **Modify the CSS files:**
  - Start with the default theme or ThemeRoller-generated theme
  - Adjust the jquery-ui.theme.css file or individual plugin stylesheets
  - Recommended to only change jquery-ui.theme.css file and images
- **Write completely custom CSS: (most control, not recommended)**
  - Custom CSS for each plugin without using the framework classes or plugin-specific stylesheet
  - Achieve complex look & feel, highly customized markup
  - Require manual updates for future plugins.



# The jQuery UI CSS Framework

- The CSS Framework provides semantic presentation classes
  - indicate the role of an element within a widget
  - such as a header, content area, or clickable region
- Applied consistently across all widgets for similar look & feel
  - Accordion or buttons have the same ui-state-default class
  - On hover, class is ui-state-hover, then ui-state-active when selected
- Framework styles only include attributes that affect the look and feel
  - Primarily color, background images, and icons
  - These are "safe" and will not affect functionality of individual plugins



# Plugin specific stylesheets

- Plugin specific stylesheets are separated in jquery-ui.structure.css
- Additional structural style rules
  - Dimensions, padding, margins, positioning, and floats
- Developers of jQuery plugins should leverage the jQuery UI CSS Framework because it will make it much easier for end users to theme and use your plugin

# Modern Web Development Summary

with HTML5, CSS3, & JavaScript libraries  
(includes jQuery) (5 days)



# Course Objectives

- What is **Modern Web Development**?
  - Set up a local **Node** server for development and make **AJAX** calls
- What are **HTML5** semantic tags, forms, and features?
  - Work with **HTML5** APIS: Geolocation, Media, Canvas, and Web Storage
- How can **CSS3** separate design from content and address accessibility?
  - Add styling to pages and save money in print
- What is **Bootstrap**?
- What is Modern **JavaScript** (ES6)?
- How can **jQuery** enhance productivity and allow cross-browser compatibility?