WEEKLY OVERVIEW

WEEK 9	Final Project Lab / Student Choice
WEEK 10	Final Project Lab / Presentations

STUDENT CHOICE

LEARNING OBJECTIVES

- Implement a fluid layout with Bootstrap.
- Understand how to use ARIA roles to make web content more accessible.
- Use jQuery to make a request for data from a web service API
- Add media to a web page by embedding code for a widget
- Choose the most appropriate file format for an image

AGENDA

- Bootstrap (review)
- Accessibility
- The web development landscape
- REST and APIs
- Audio & video

STUDENT CHOICE

BOOTSTRAP REVIEW

LAYOUT GRIDS

- Composition in any visual art is the placement or arrangement of visual elements — how they are organized on a page.
- Many designers use a grid structure to help them position items on a page.

BENEFITS OF GRID SYSTEM LAYOUTS

While grids might seem like a restriction at first, they have many benefits including:

- Creating continuity between different pages
- Helping users predict where they can find information on various pages
- Making it easier to add new content to the site in a consistent way
- Helping designers/developers collaborate in a consistent way

CSS FRAMEWORKS

CSS FRAMEWORKS PROVIDE CODE FOR COMMON TASKS, SUCH AS:

- Creating responsive layout grids
- Styling forms
- Styling UI elements, etc.

PROS:

- Save developers from repeatedly writing code for the same tasks
- Tested across various browsers
- Useful for rapid website development

CONS:

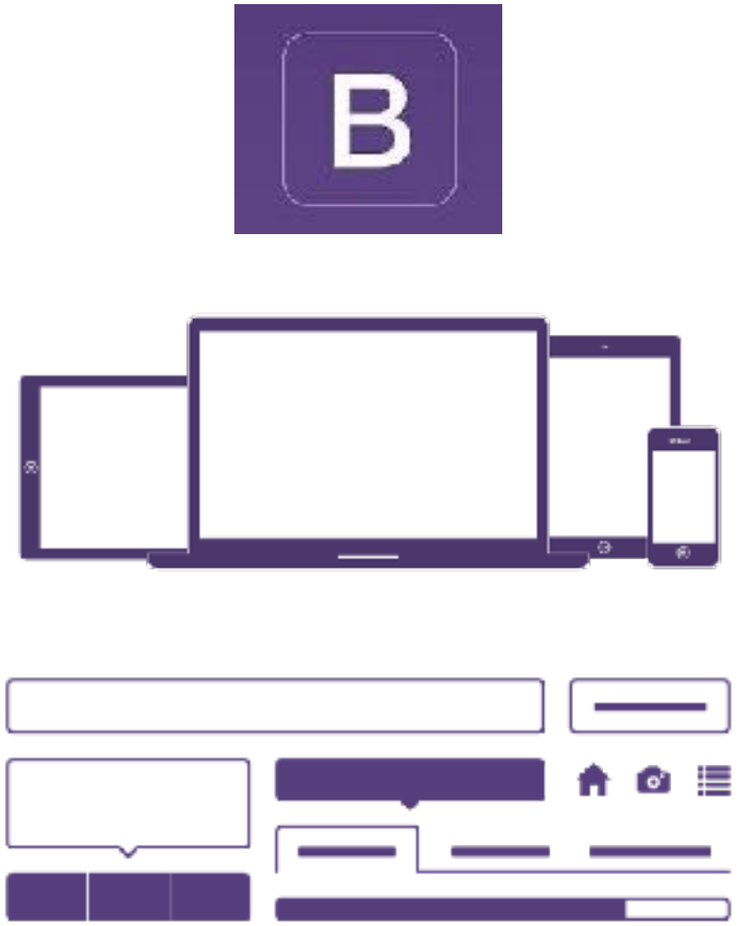
- Often require developers to use class names in HTML that only control the presentation of the page (rather than describe content)
- Often contain more code than you need for your particular page, creating “bloat”

HELLO BOOTSTRAP!

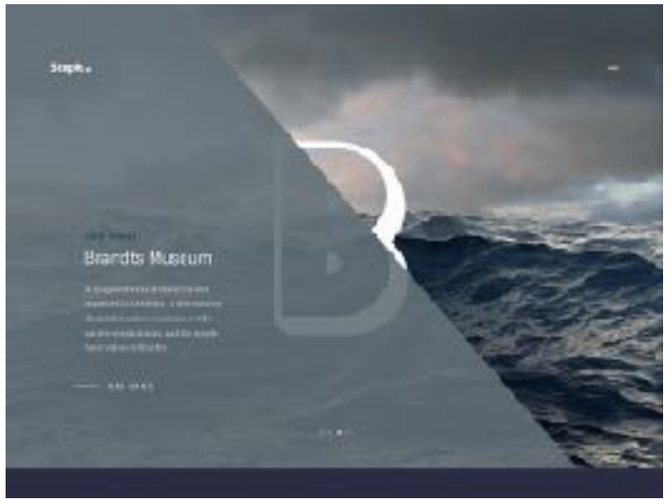
- **Bootstrap** is *"the most popular HTML, CSS and JS framework for developing responsive, mobile first projects on the web."*

FEATURES:

1. Grid system
2. Forms/buttons
3. Navigation
4. Tabs and pills
5. Alerts/error messages
6. Modals
7. And much more



SITES THAT USE BOOTSTRAP



And many, many more: [Bootstrap Expo]

GETTING STARTED

GETTING STARTED

1. Download Bootstrap here
2. Once downloaded, unzip and open folder in Finder. You'll see something like this:

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

3. We'll then need to add the files we want to use to our project and include those files in our HTML.

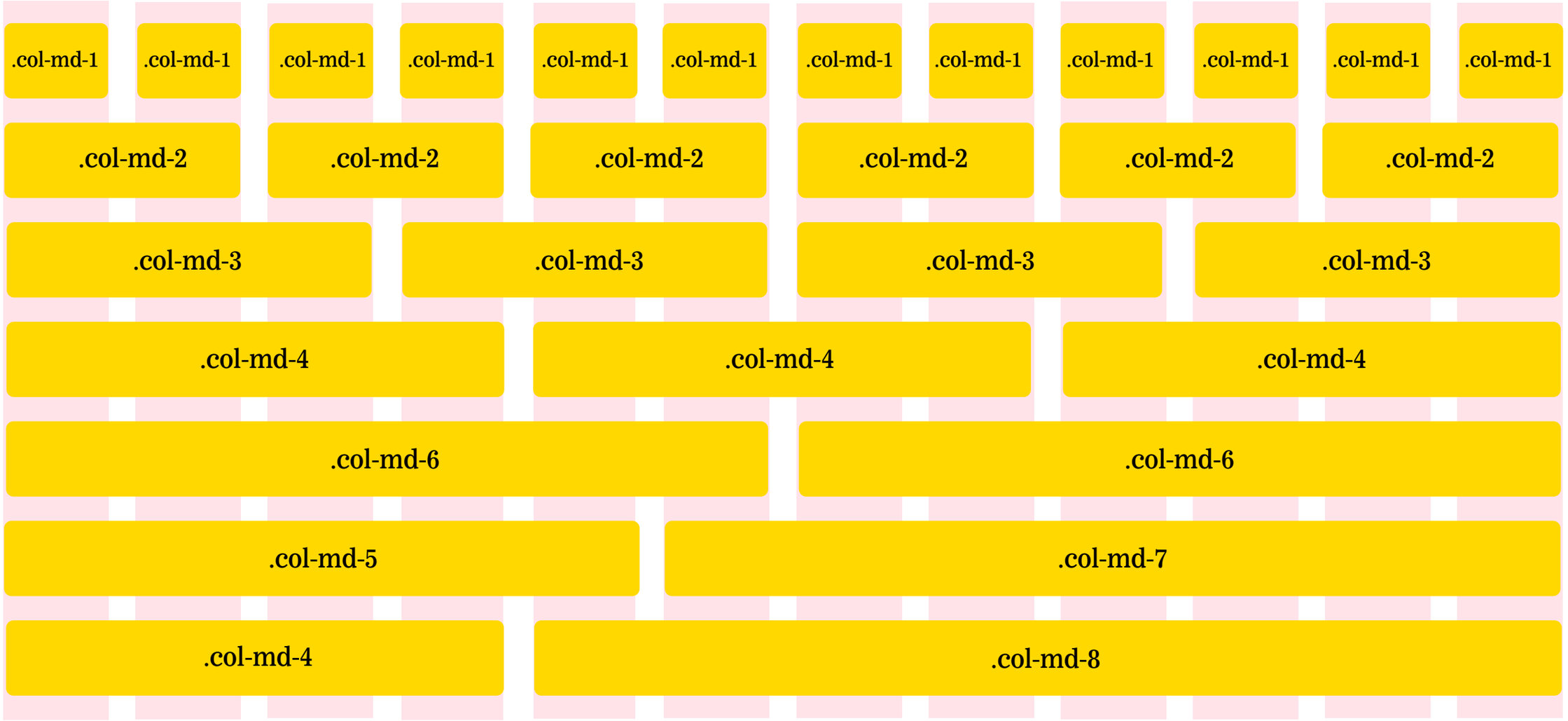
GRID SYSTEM

BOOTSTRAP — GRID SYSTEM

- Bootstrap includes a responsive, mobile-first fluid grid system that appropriately scales up to **12 columns** as the device or viewport size increases.
- Page layouts are created through a series of rows and columns that house your content

BOOTSTRAP — GRID SYSTEM

- It includes predefined classes for easy layout options.



BOOTSTRAP — SETTING UP GRID

1. Add containing element to page
2. Add rows to create horizontal groups of columns
3. Place content within columns, specifying the number of twelve available columns you wish to span.

STEP 1 — CONTAINING ELEMENT

- Bootstrap requires a containing element to wrap site contents and house our grid system. You may choose one of two containers to use in your projects:

Use `.container` for a responsive fixed width container.

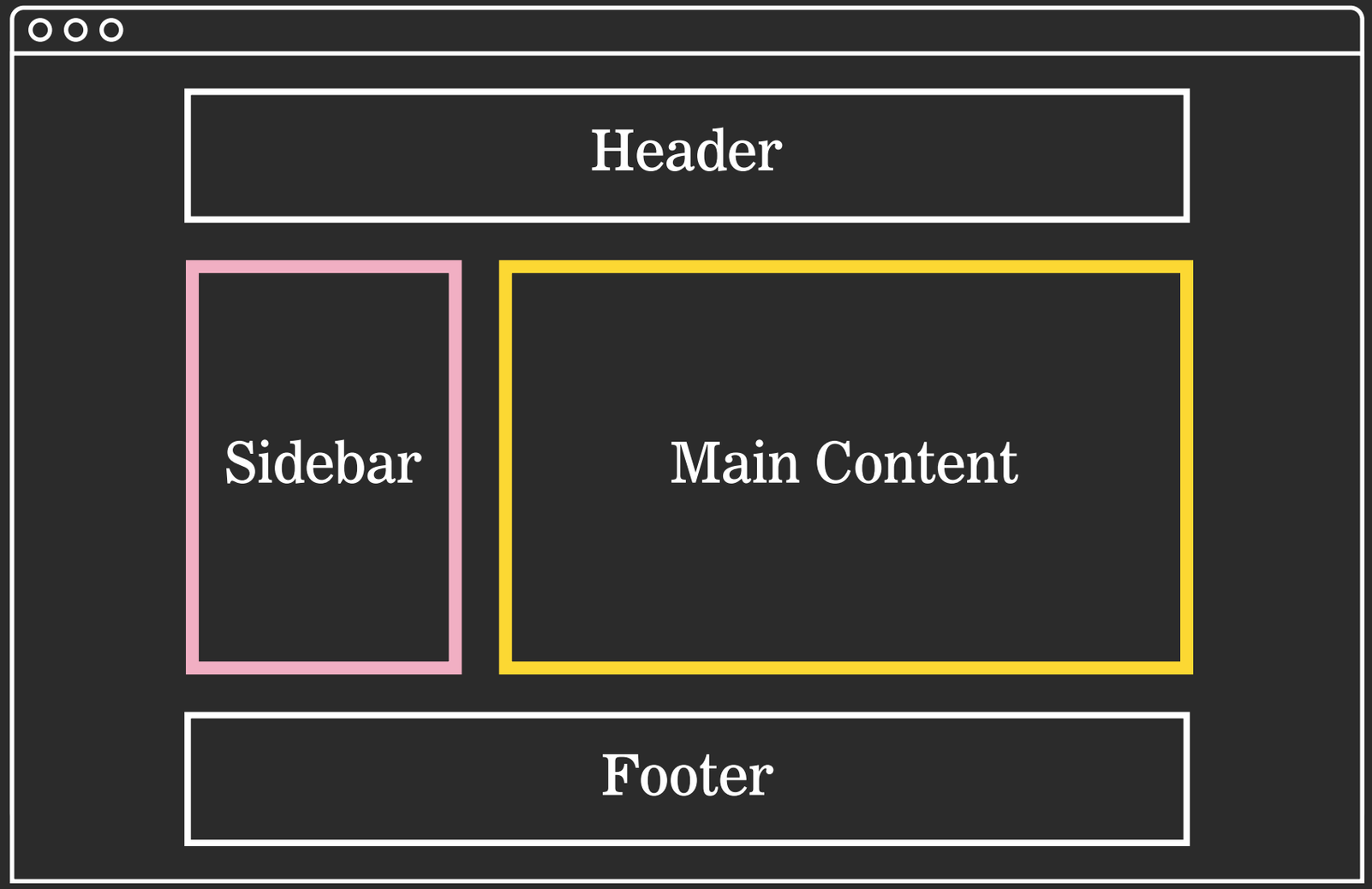
```
<div class="container">
  ...
</div>
```

Use `.container-fluid` for a full width container, spanning the entire width of your viewport.

```
<div class="container-fluid">
  ...
</div>
```

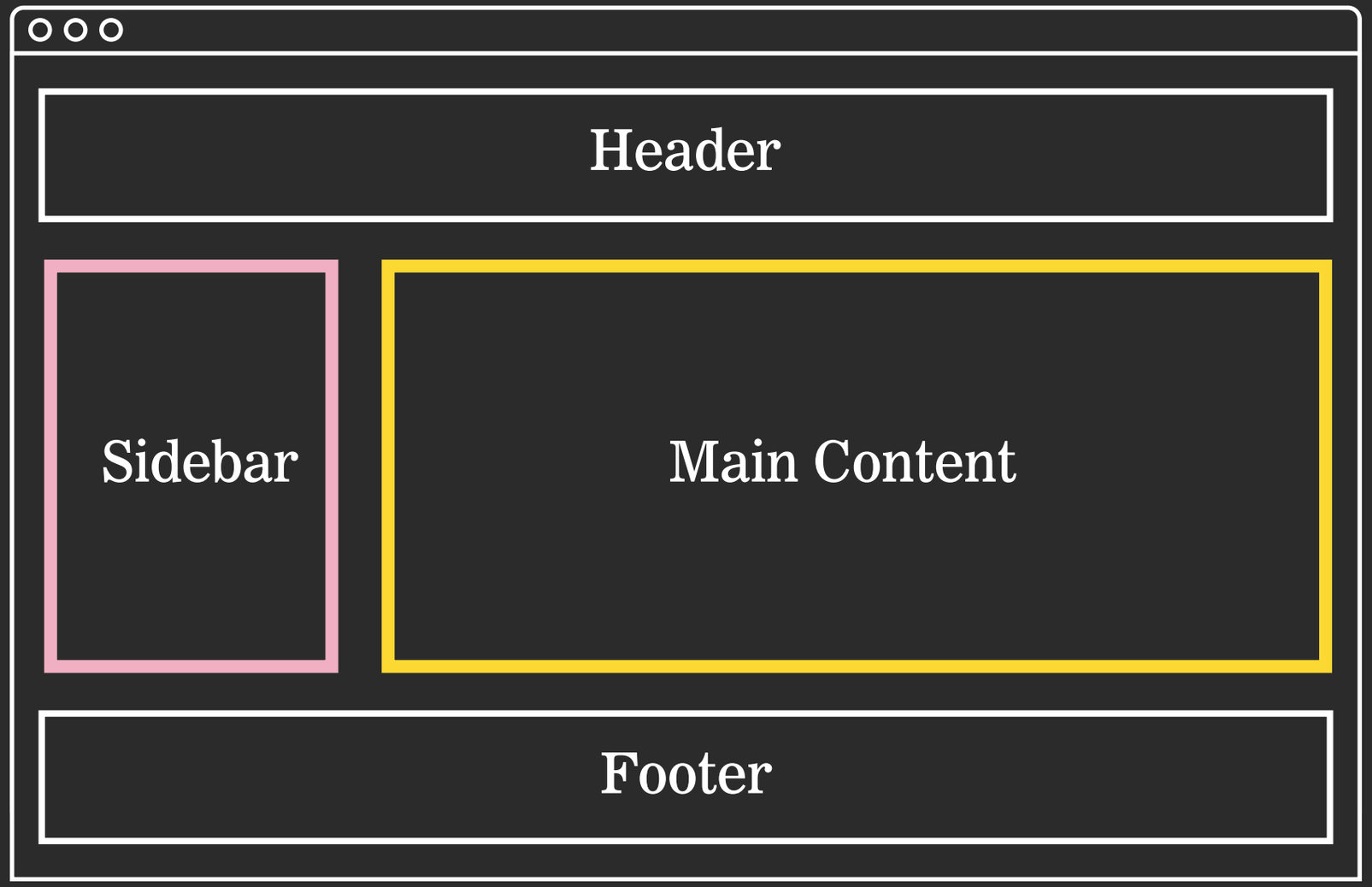
FIXED WIDTH LAYOUT

Fixed width layouts do not change size as the user increases/ decreases width of browser window



FLUID LAYOUT

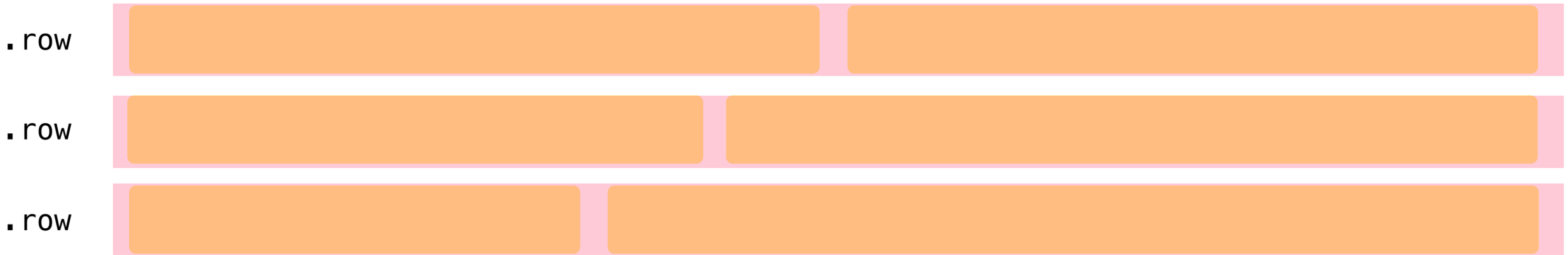
Fluid layouts stretch and contract as the user increases/ decreases the size of their browser window



STEP 2 — ADD ROWS

- Use rows to create horizontal groups of columns.
- Only columns may be immediate children of rows.

```
<div class="row">
    (columns)
</div>
```



STEP 3 — ADD COLUMNS — STACKED TO HORIZONTAL

- Content should be placed within columns.

```
<div class="row">
  <div class="col-md-5">.col-md-5</div>
  <div class="col-md-7">.col-md-7</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
```

ACTIVITY

KEY OBJECTIVE

- Implement a fluid layout with Bootstrap.

TYPE OF EXERCISE

- Individual/Partner

LOCATION

- Starter Code > rows_columns

TIMING

10 min

1. Take a look at grid.png
2. Recreate the grid in your HTML
3. BONUS: Add Bootstrap code to create a responsive layout for smaller screens (Hint: Use the `.col-sm-#` classes)

ACTIVITY

KEY OBJECTIVE

- Understand how Bootstrap works

TYPE OF EXERCISE

- Partner

TIMING

4 min

1. With a partner, use Chrome's inspect element to look at the different styles that Bootstrap is adding. See if you can find them in the Bootstrap stylesheet!

ACCESSIBILITY

WAI

- Web Accessibility Initiative
- Task force that creates standards for making websites accessible to the whole range of users

WCAG

- Web Content Accessibility Guidelines
- “the international standard for making web content more accessible to people with disabilities” (from https://www.w3.org/WAI/intro/wcag.php)

WCAG GUIDELINES

Perceivable

Operable

Understandable

Robust

WCAG GUIDELINES

Perceivable

Operable

Understandable

Robust

- Provide text alternatives for non-text content.
- Provide captions and other alternatives for multimedia.
- Create content that can be presented in different ways, including by assistive technologies, without losing meaning.
- Make it easier for users to see and hear content.

WCAG GUIDELINES

Perceivable

Operable

Understandable

Robust

- Make all functionality available from a keyboard.
- Give users enough time to read and use content.
- Do not use content that causes seizures.
- Help users navigate and find content.

WCAG GUIDELINES

Perceivable

Operable

Understandable

Robust

- Make text readable and understandable.
- Make content appear and operate in predictable ways.
- Help users avoid and correct mistakes.

WCAG GUIDELINES

Perceivable

Operable

Understandable

Robust

- Maximize compatibility with current and future user tools.

WCAG GUIDELINES

Perceivable | **Operable** | Understandable | Robust

- Make all functionality available from a keyboard.
- Give users enough time to read and use content.
- Do not use content that causes seizures.
- Help users navigate and find content.

Through correct use of standard HTML code, developers can do everything but this

ARIA

- Accessible Rich Internet Applications suite
- ARIA **roles** identify the purpose of an element
- ARIA **attributes** (`aria-`) describe **properties** and **states** of an element

ARIA


FIRST RULE OF ARIA USAGE
WE DO NOT USE ARIA
memegenerator.net

ARIA ROLES

- Abstract roles
- Widget roles
- Document structure roles
- Landmark roles (most commonly used)

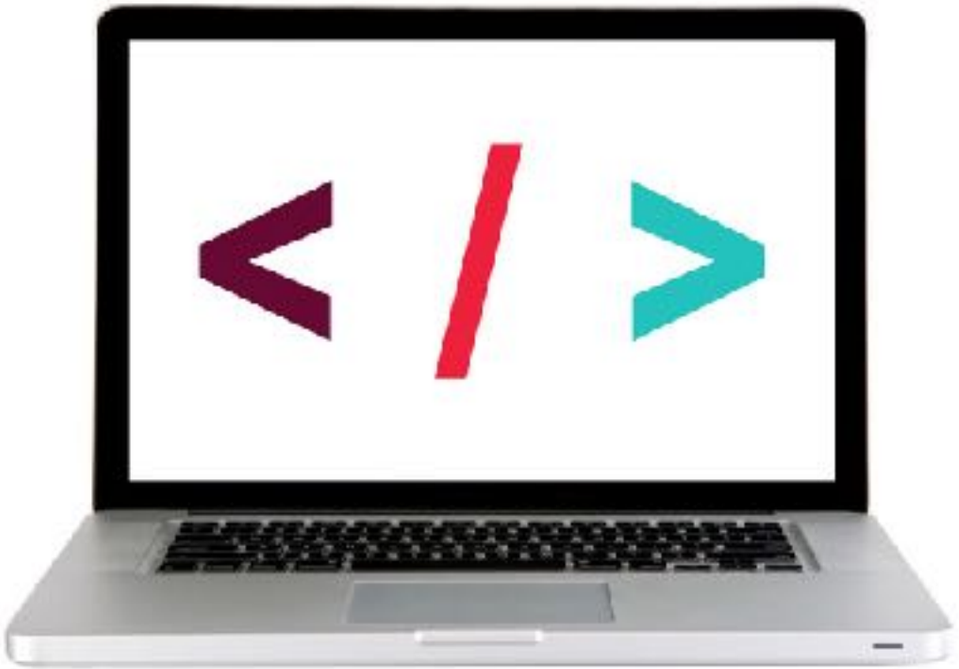
assigned with the `role` attribute in HTML

```
<nav class="global-nav deluxe" id-"main-nav" role="navigation">
…
</nav>
```

ARIA ROLES

Landmark Roles	
`application`	web app (rather than web document)
`banner`	site-oriented content (logo, company info, site search)
`complementary`	content complementary to main content but still meaningful on its own
`contentinfo`	info about document (copyrights, privacy statements)
`form`	collection of items and objects that create a form
`main`	content directly related to central topic of document
`navigation`	navigational elements for navigating the document and related documents
`search`	collection of items and objects that create a search facility

LET'S TAKE A LOOK



https://www.w3.org/TR/wai-aria/roles

EXERCISE — ARIA ROLES

KEY OBJECTIVE

- Understand the use of ARIA roles in web accessibility

TYPE OF EXERCISE

- Individual/paired

EXECUTION

6 min

1. Open https://alistapart.com/column/wai-finding-with-aria-landmark-roles in your browser
2. Examine the web page elements and identify the ARIA roles in use
3. Sketch the structure of the document, then label the sections of the page with their roles
4. BONUS: Identify other roles you would add to the page, and point out what sections you would add them to

STUDENT CHOICE

THE WEB DEVELOPMENT LANDSCAPE

FRONT END DEVELOPMENT

- client/browser code (HTML, CSS, JS)
- what the user sees and interacts with

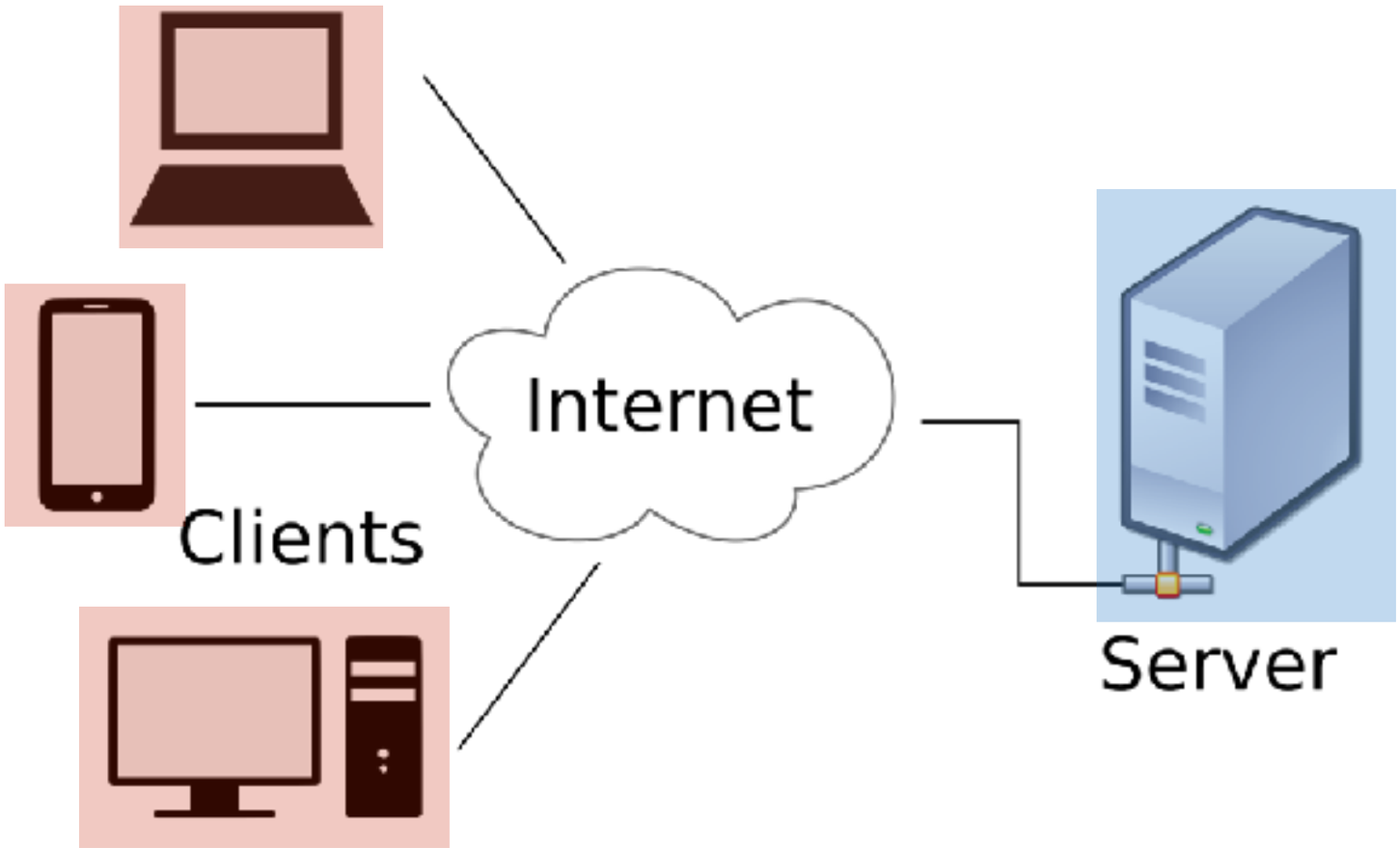
BACK END DEVELOPMENT

- server-side code
- handles such functions as routing, data handling, and databases
- stuff behind the scenes that makes web applications work

Front end

- HTML
- CSS
- JS

Back end

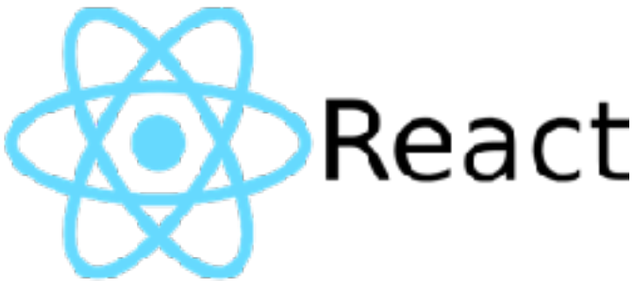


- JS
- Python
- Ruby
- PHP
- …

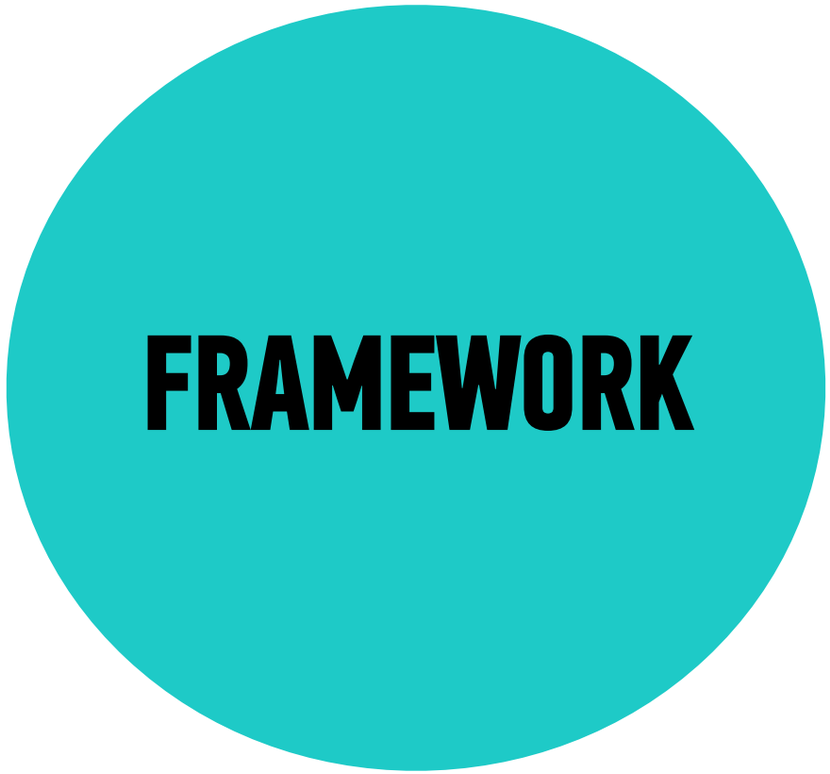
THE WEB DEVELOPMENT LANDSCAPE

combine basic
front end building
blocks

to create libraries and frameworks



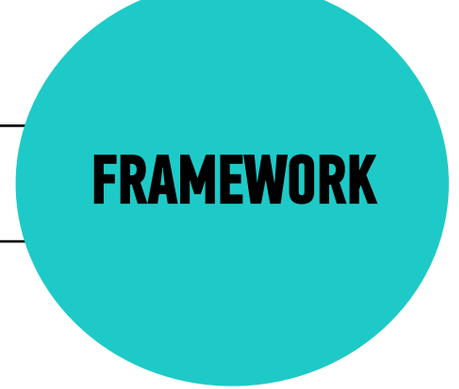
LIBRARIES VS FRAMEWORKS



LIBRARY

- Set of predefined functions that your code calls
- Each call performs work and returns a result (and control) to your code
- Specific, well-defined operations
- Example: jQuery

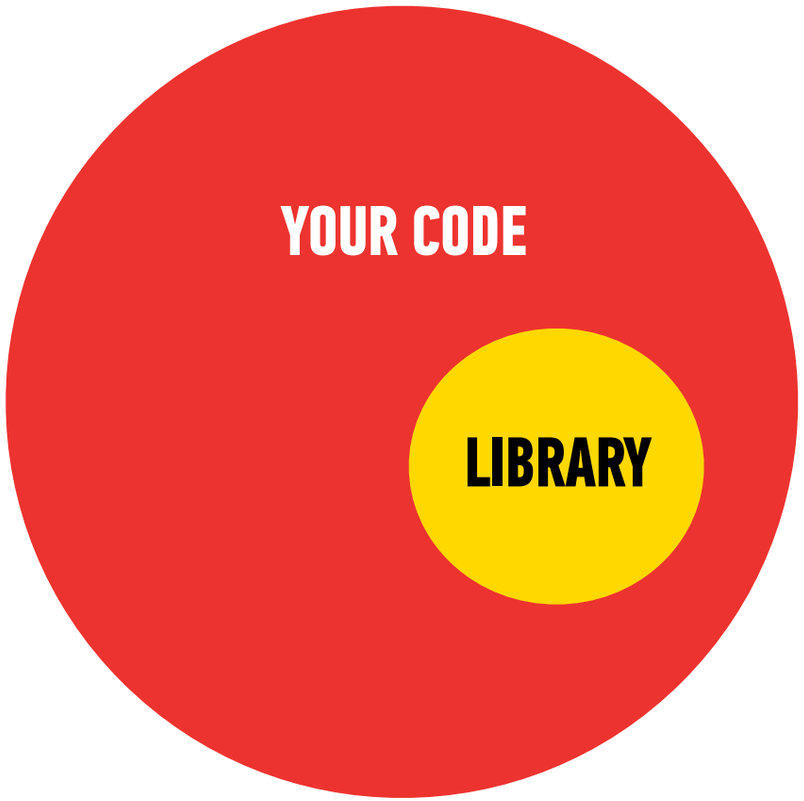
FRAMEWORK

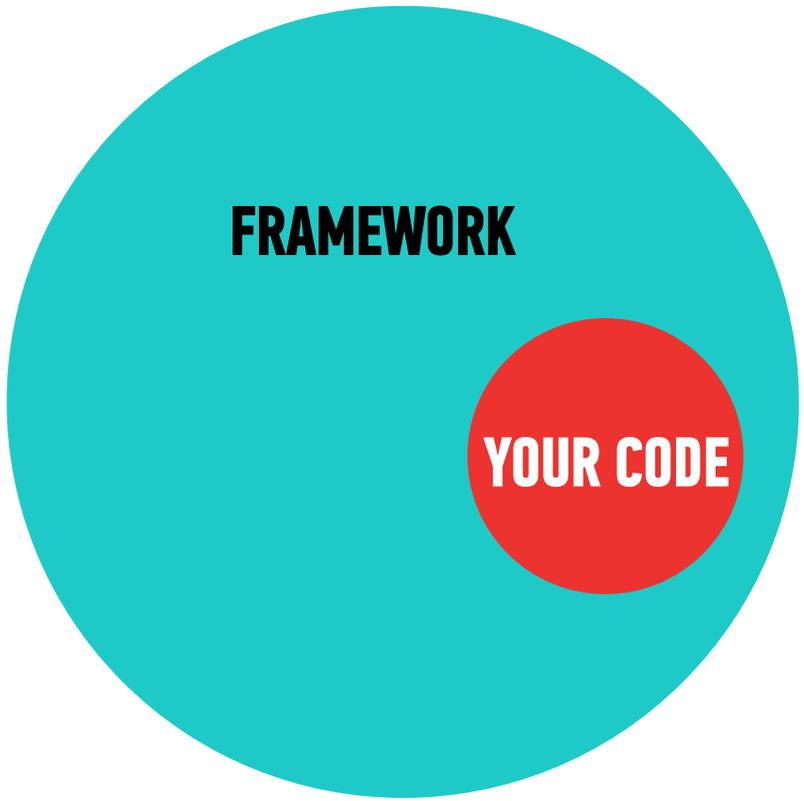


- Opinionated architecture for building software
- Control-flow exists, you fill in with your code
- Calls your code; is always in control
- Examples: Angular, React, Vue

LIBRARIES VS FRAMEWORKS

"Your code calls a library, but a framework calls your code."

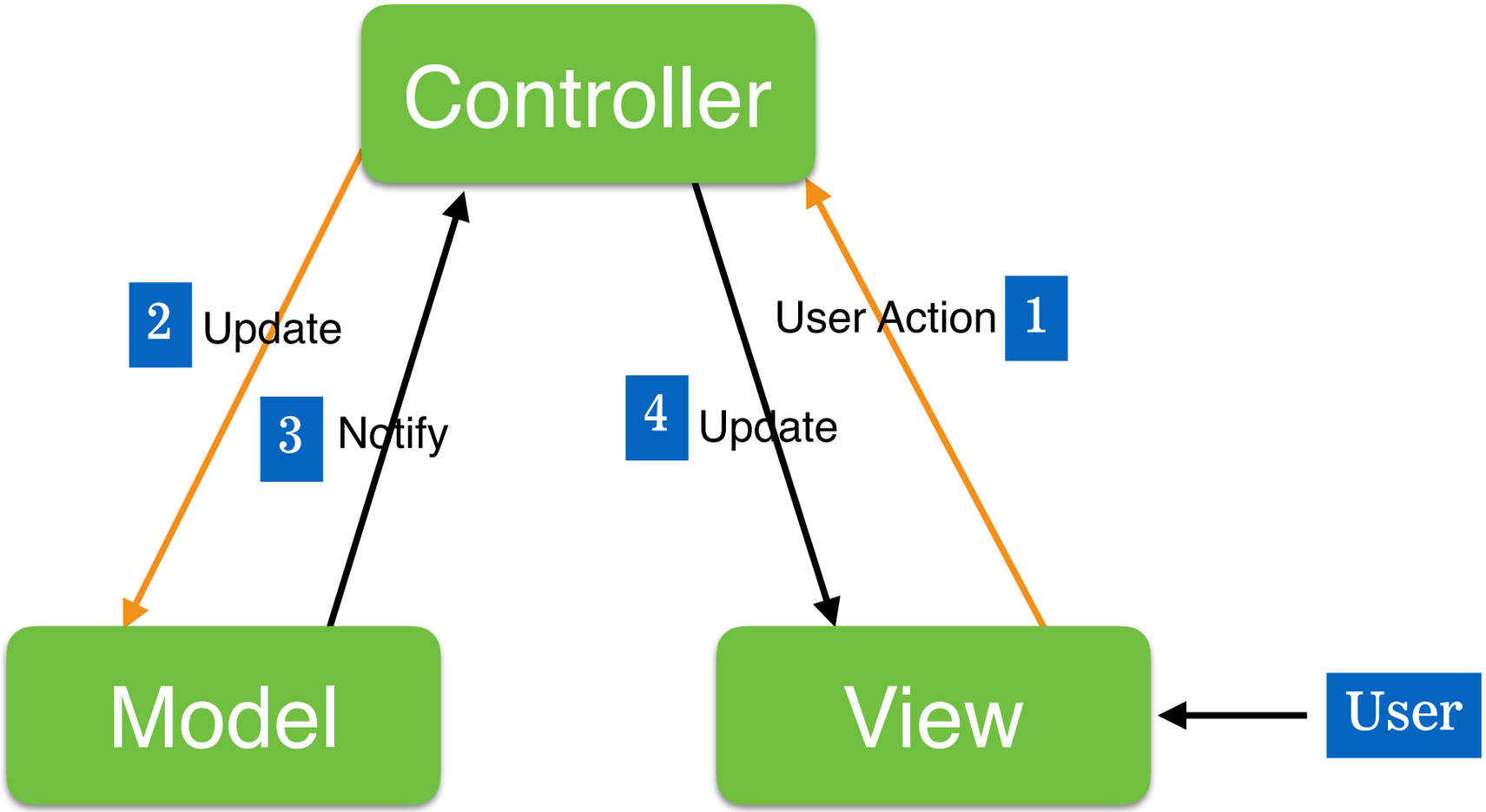




MODEL-VIEW-CONTROLLER (MVC)

- **Model**: handles data and business logic
- **View**: presents data to user in any supported format and layout
- **Controller**: receives user inputs and calls appropriate resources to carry them out

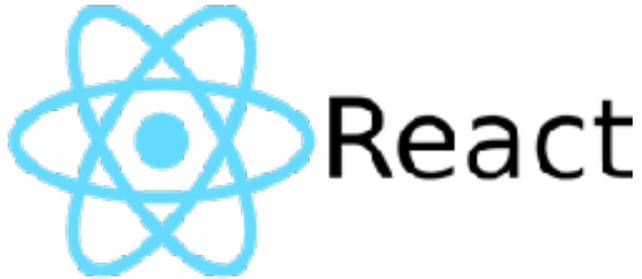
MODEL-VIEW-CONTROLLER (MVC)



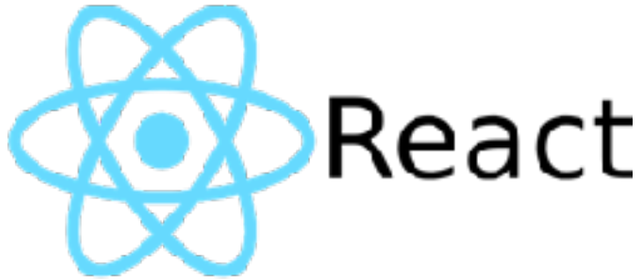
WHY USE FRAMEWORKS?

- Standard / well known
  - Dictates a method that cannot be (easily) ignored
- Common problems already solved
  - Cross Browser
  - Accessibility
  - Complexity of state
  - etc

- somewhere between a framework and a library
  - “a framework that feels like a library”
- It only cares about your views (V from MVC)
- BUT you must do your views the React way



- Define a small view template
- Small, reusable, and independent



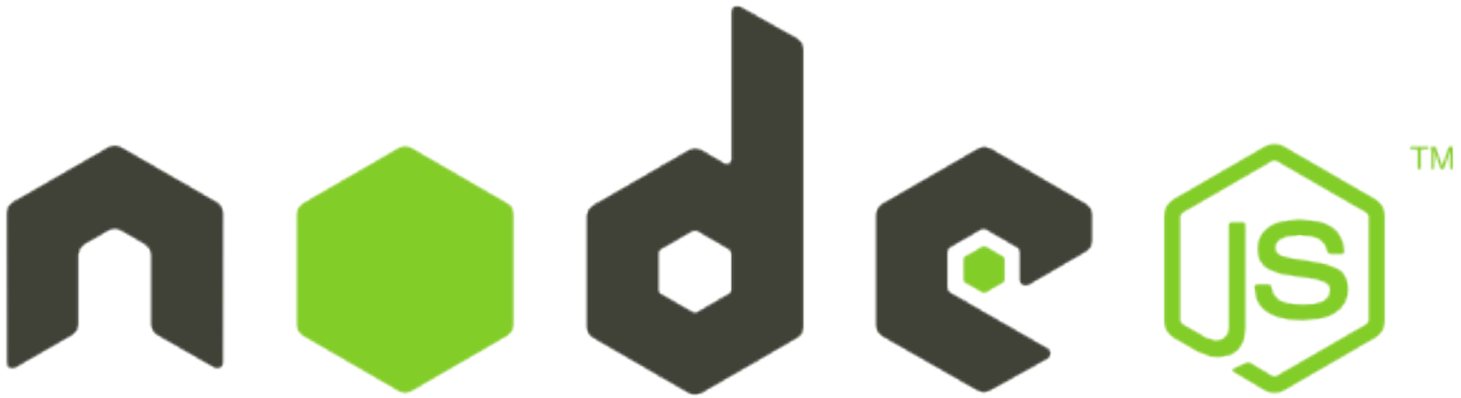
- Angular 1.x (angularjs.org) released in 2010
  - Widely deployed
  - Still being supported
- Angular 2.x (angular.io) released in September 2016
  - Complete rewrite
  - Based on TypeScript, which is a superset of JavaScript

VUE

- more flexible than Angular and React
- considered easier to learn/use than Angular

NODE

- Back-end environment
- Runs on server
- Allows you to write your back end code in JavaScript, rather than learning a separate back-end language.
- Supports server-side frameworks like Express, Meteor, and Socket.IO



In the Slack channel for today (`18-student-choice`), share your answers to one or both of the following questions:

- **Significant thing**:
  "The most significant thing I learned today about the web development landscape is _____."
- **Outstanding question**:
  "My biggest outstanding question on the web development landscape is _____."

STUDENT CHOICE

REST AND APIS

WEB SERVICE

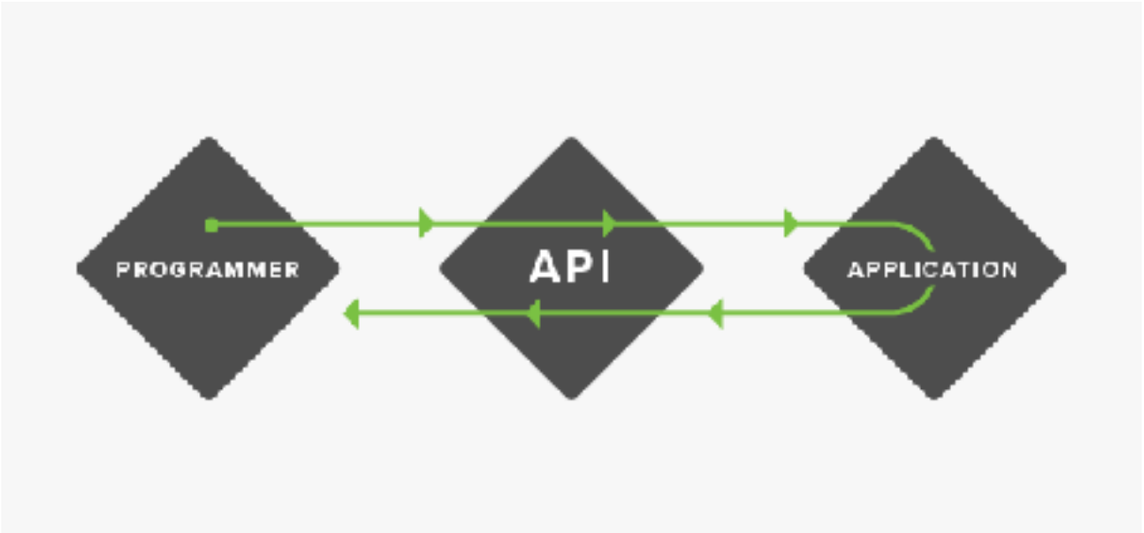
- An online source of data
- Communicate using HTTP, but instead of markup we receive data
- We can use multiple services in a single app

API = application programming interface

- Each service has an API, which is a predefined set of objects, properties, and methods anyone can use to access that service
- Any service we access online through our apps will have an API
- Intermediary; allows different pieces of software to communicate

APIS IN THE REAL WORLD

- Most APIs are unique, like separate languages
- API for devices (iPhone); for operating systems (macOS); for JavaScript libraries (jQuery API)



HOW FRONT END DEVELOPERS USE APIS

- Use HTTP to request/receive structured data from endpoints on a server
- **Endpoints** are addresses (URLs) that will return data (JSON) instead of markup (HTML)

WHAT WE NEED TO KNOW TO USE AN API

- Its terms of service (paid service? limit on usage?)
- How to make a request (URL and parameters)
- What kind of data is returned and how to parse it

HOW MIGHT A SERVICE REQUEST BE DIFFERENT THAN USING OUR OWN DATA?

- May need to authenticate when requesting data
- May be a lag, requiring user notification
- Request may result in an error

REST (representational state transfer)

- architectural style of web applications
- transfers a representation of the state of a server resource to the client

RESTful API

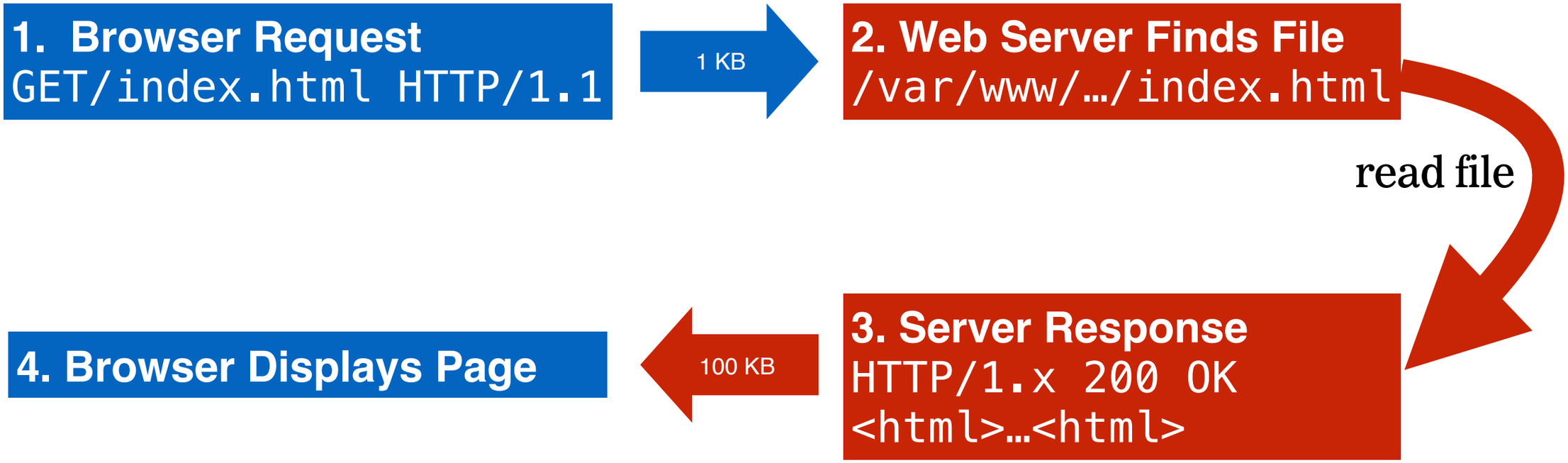
- adheres to REST architecture
- uses
  - a base URL
  - an Internet media type (such as JSON)
  - standard HTTP methods

HTTP

HTTP (hypertext transfer protocol)

- System of rules for how web pages are transmitted between computers
- Defines the format of messages passed between HTTP clients and HTTP servers
- A client sends a **request** to a server.
- A server sends a **response** back to a client.

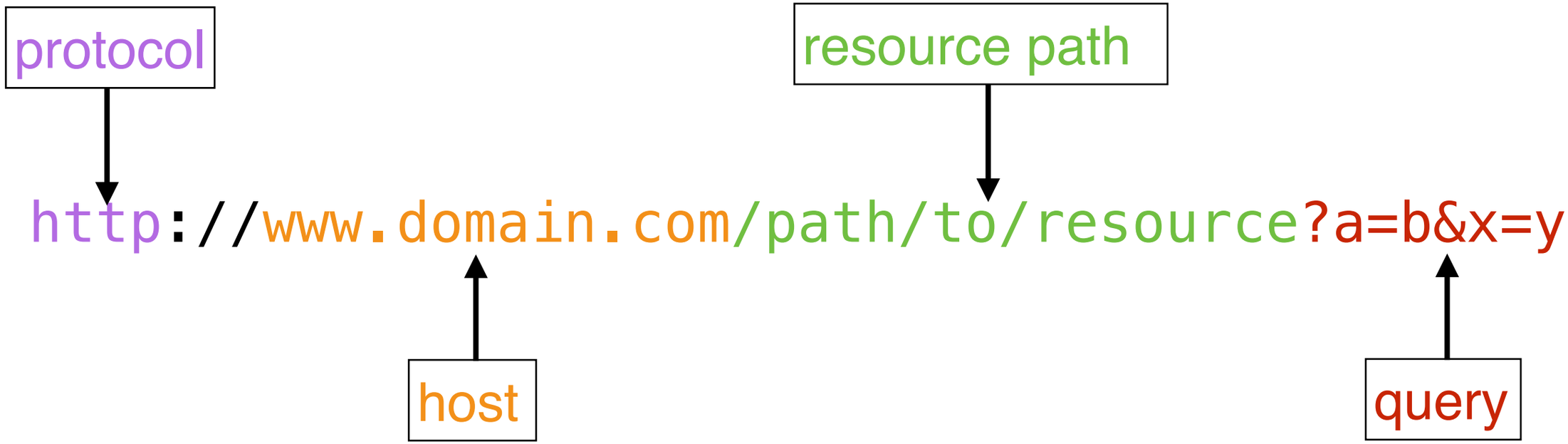
HTTP REQUEST AND RESPONSE



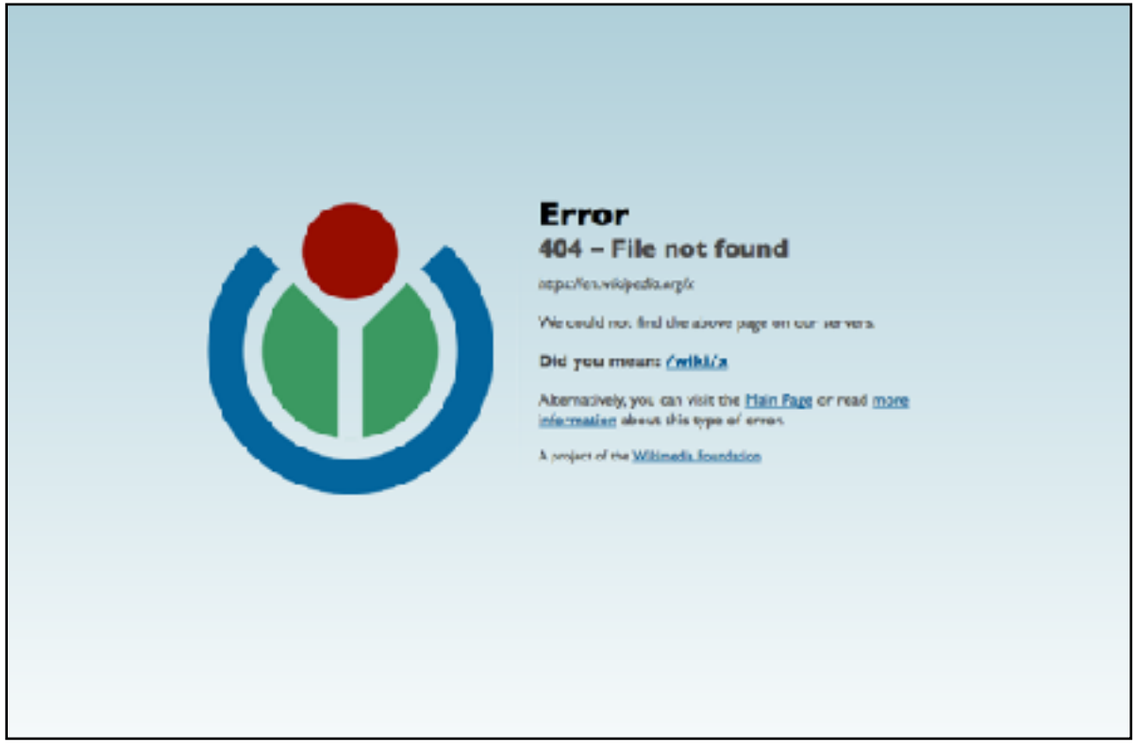
HTTP (hypertext transfer protocol)

- HTTP clients are generally web browsers (Chrome, Firefox, Safari, Edge, etc.)
- HTTP servers are web servers (Apache, Nginx, etc.)
- **Web applications** are programs that plug into a web server, process the HTTP requests that the server receives, and generate HTTP responses

HTTP REQUESTS IN EVERYDAY LIFE



HTTP STATUS CODES



HTTP STATUS CODES

200	Okay
301	Moved permanently
302	Moved temporarily
400	Bad request
403	Forbidden
404	Not found
500	Internal server error

Ajax

Ajax

- Originally AJAX (Asynchronous JavaScript and XML)
- XML is a format for data interchange that's derived from the same markup language that HTML comes from.
- Since JSON was codified, it's become the standard for data interchange on the web, so Ajax is no longer functionally an acronym.

What does Ajax let us do?

- Communicate with servers from within our apps
- Make the communication asynchronous (in the background)
- We can update interfaces and content without refreshing the page

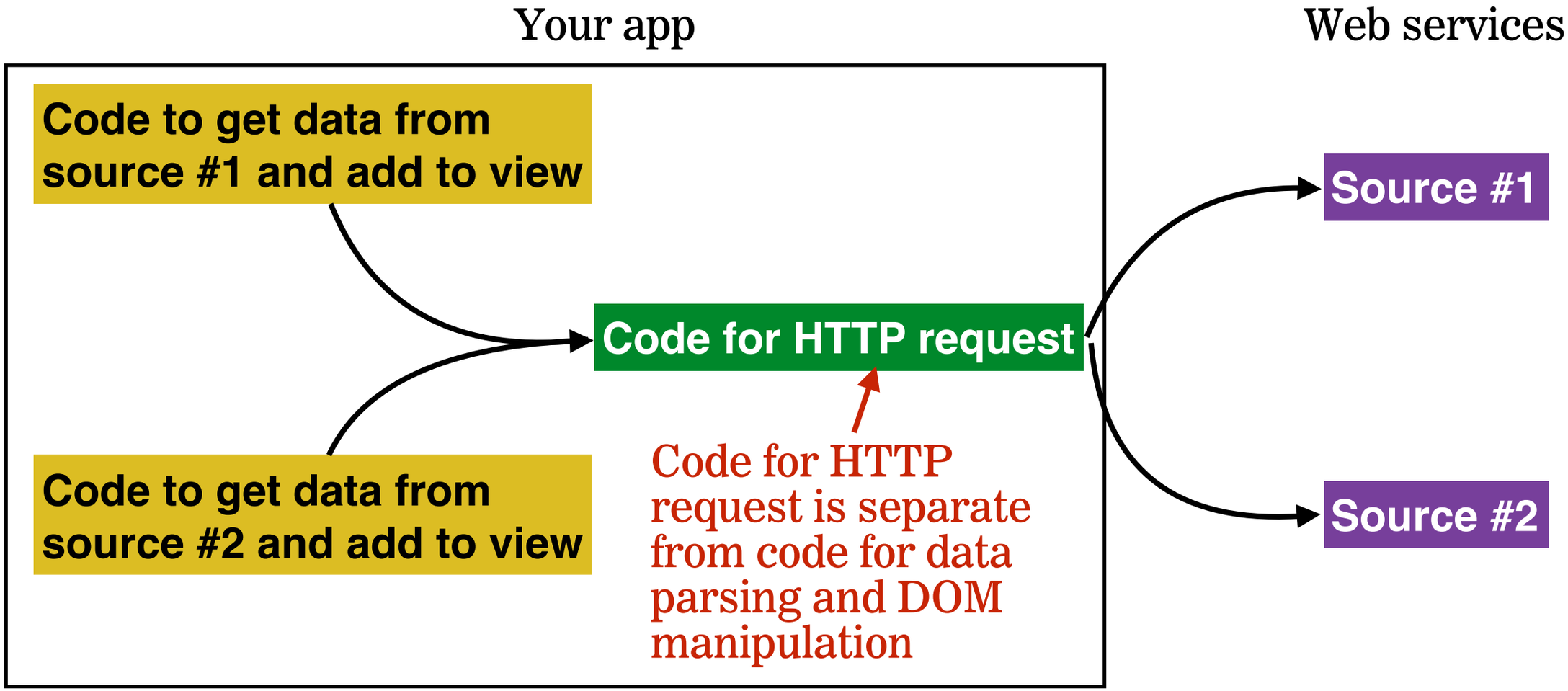
SEPARATION OF CONCERNS

- Programming principle of keeping different aspects (or **concerns**) of an application separate
- Many ways to do this
- One common separation is between data (the information we're presenting) and view (the code that determines how data is presented)
- We should be able to change the code for one concern without affecting the code for the other

SEPARATION OF CONCERNS - HTTP

- For HTTP code, the code for the client should be abstracted from the code for the HTTP request
- You should be able to reuse your code for multiple APIs/services, rather than making custom code for each one

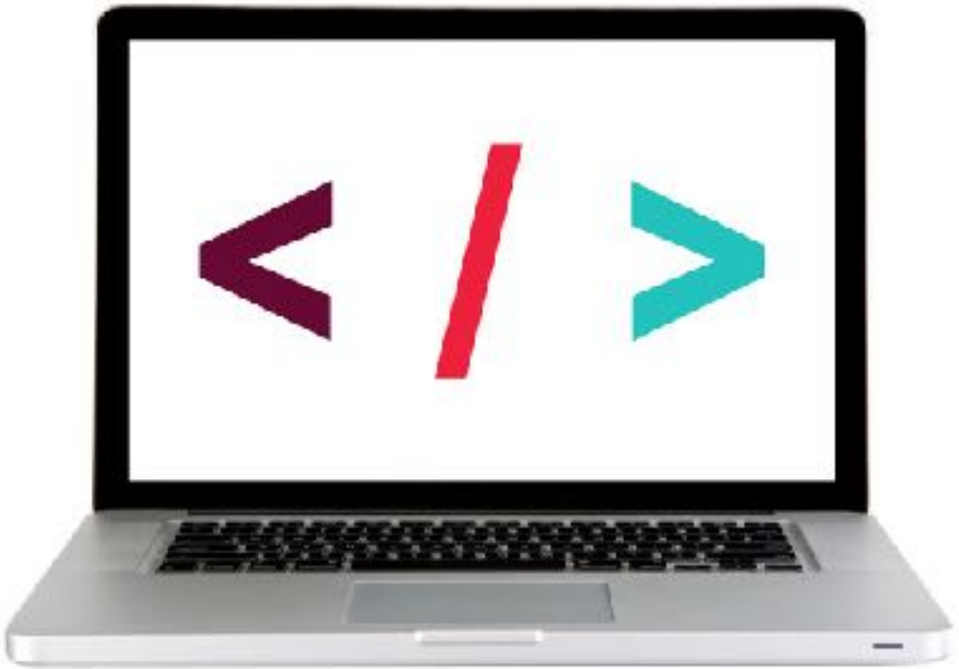
SEPARATION OF CONCERNS - HTTP



Creating an Ajax request with jQuery

method	description
`$.get()`	loads data from a server using an HTTP GET request
`$.ajax()`	performs an Ajax request based on parameters you specify

LET'S TAKE A LOOK



EXERCISE — GETTING DATA FROM AN API

KEY OBJECTIVE

- Use jQuery to make a request for data from a web service API

TYPE OF EXERCISE

- Individual/partner

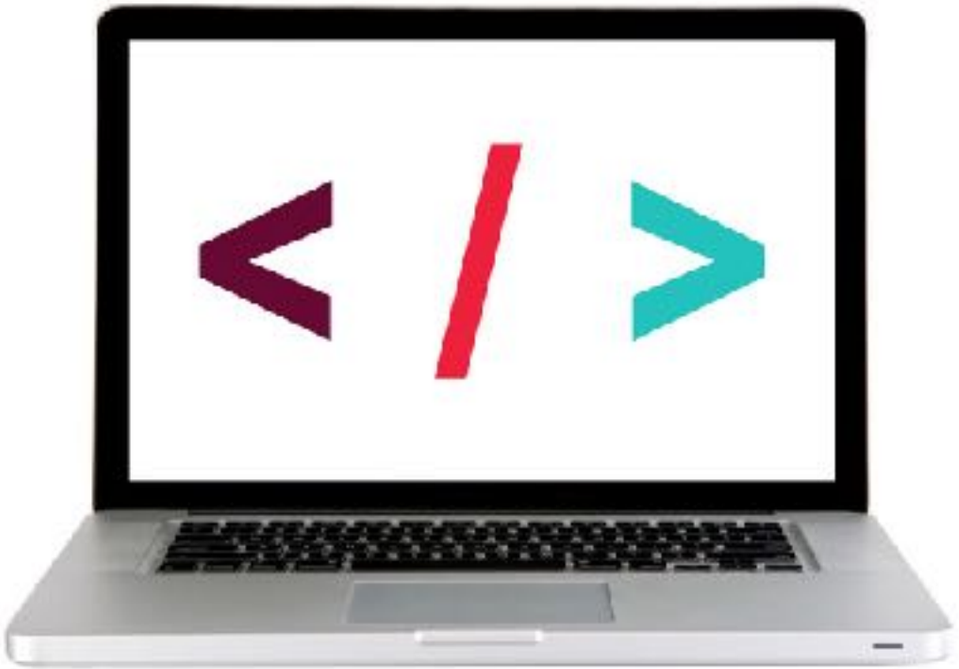
EXECUTION

5 min

1. Open start files > ajax in your editor
2. Use jQuery to request data from the following URL and log the result to the console: http://data.consumerfinance.gov/api/views.json

- A unique string assigned to you by a web service
- Associated with your account
- You include the key in your Ajax request
- The service can then identify which user is making each request, and log the volume of data used by each account

LET'S TAKE A LOOK



STUDENT CHOICE

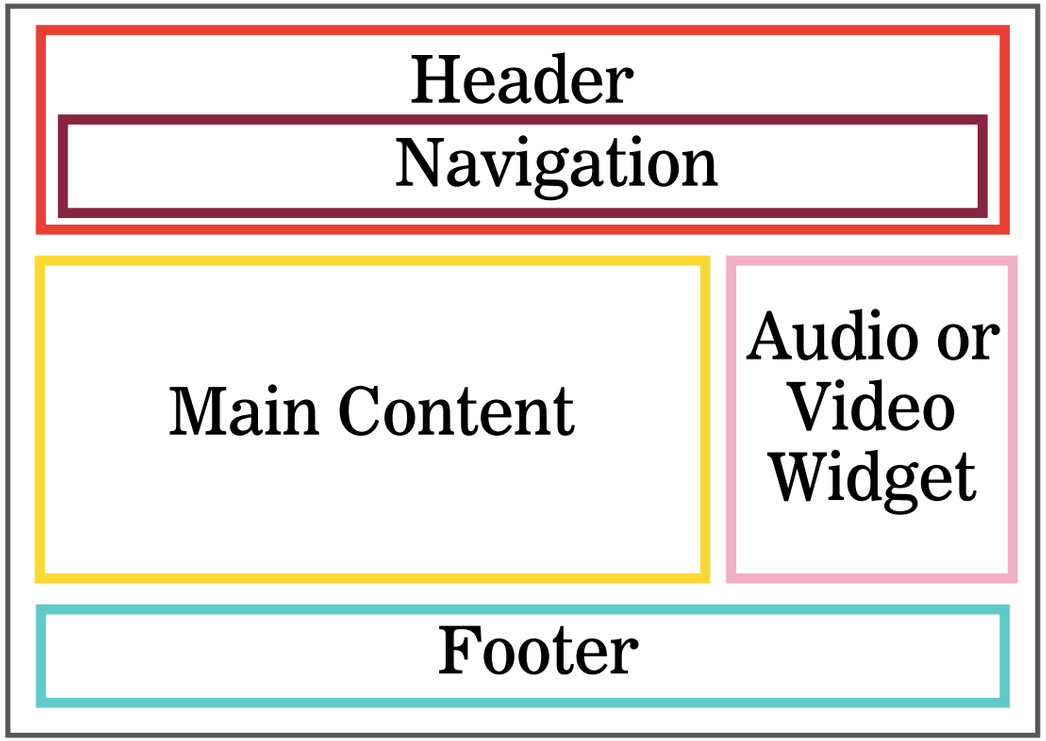
AUDIO AND VIDEO

AUDIO & VIDEO

```
<html>
<head>…</head>
<body>
  <header>
    <nav>…</nav>
  </header>
  <div id="main">…</div>
  <aside>
    …
  </aside>
  <footer>…</footer
</body>
</html>
```

`iframe` element

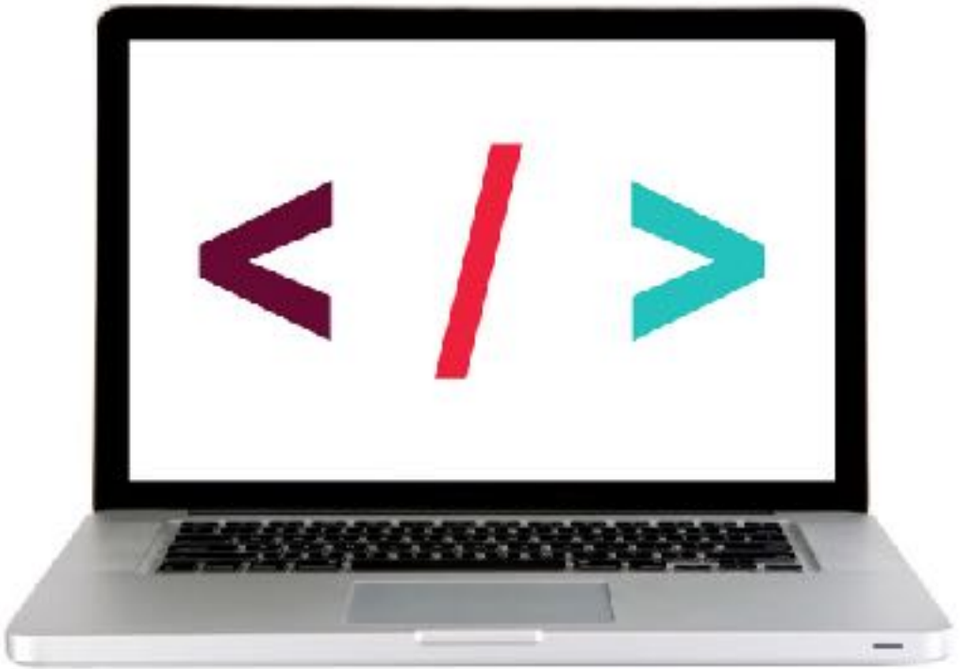
```
<iframe src="https://…"></iframe>
```



IFRAME ELEMENT

- Essentially embeds a web page within another web page
- Allows a provider to give you code for formatted content that you can put on your site, but without your site's styling affecting the content
- Offered by YouTube, Soundcloud, and many others in the code provided by the "embed" option

LET'S TAKE A LOOK



http://bit.ly/2qFiNDo

EXERCISE — EMBEDDING MEDIA WIDGETS

KEY OBJECTIVE

- Add media to a web page by embedding code for a widget

TYPE OF EXERCISE

- Individual

EXECUTION

10 min

1. Open start files > media in your editor
2. Embed https://youtu.be/iCvmsMzlF7o (or another YouTube video of your choice) in the first section element
3. Add media from at least 2 other websites of your choice to the other section elements, so all section elements have content

STUDENT CHOICE

IMAGE FORMATS

OPTIMIZING IMAGES FOR THE WEB

INSTRUCTIONS FOR PHOTOSHOP:

- Photoshop - Paid option for professionals in visual design. It's now a part of the Creative cloud and is available for $9.99/month or $119/year
- Step by step: optimizing images for the web in Photoshop

INSTRUCTIONS FOR GIMP:

- Download Gimp for Mac - Free program that’s similar to Photoshop. A little more challenging to learn, but every bit as powerful.
- Step by step: optimizing images for the web in Gimp

IMAGE FORMATS: JPEG

- JPEGs should be used for images with many different colors in the picture.

IMAGE FORMATS: GIF AND PNG

- Use GIF or PNG with saving images with few colors (a.k.a flat color)

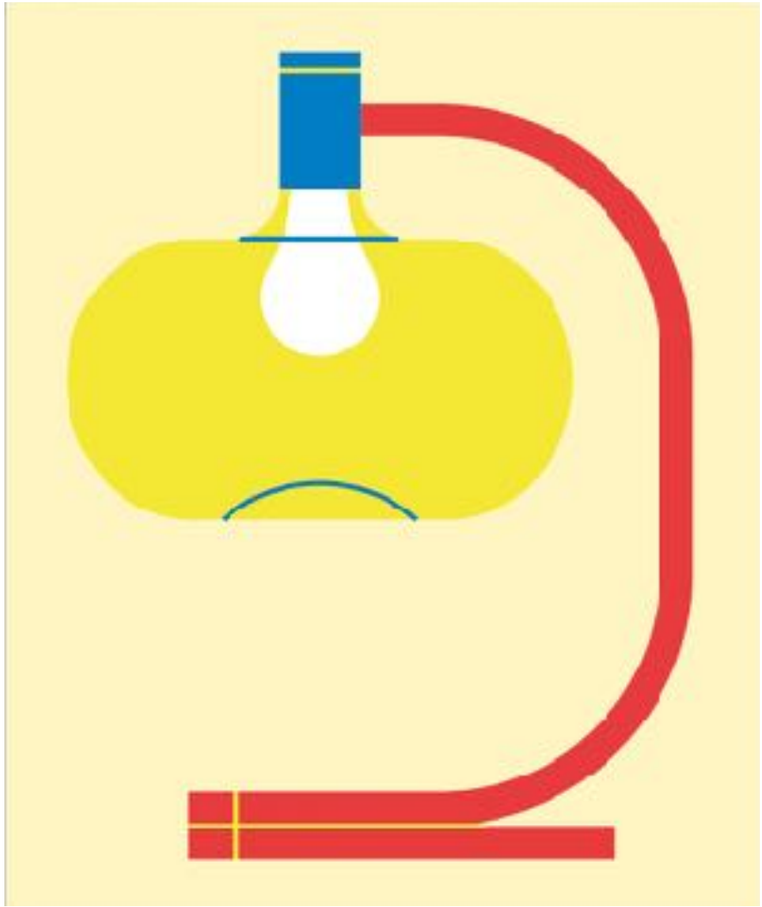


IMAGE FORMATS: PNG

- Supports transparency *and semi-transparency*, great for logos, icons, and repeating background tiles.
- Almost always preferable to a gif, unless semi-transparency is not needed, and the gif format is significantly smaller.

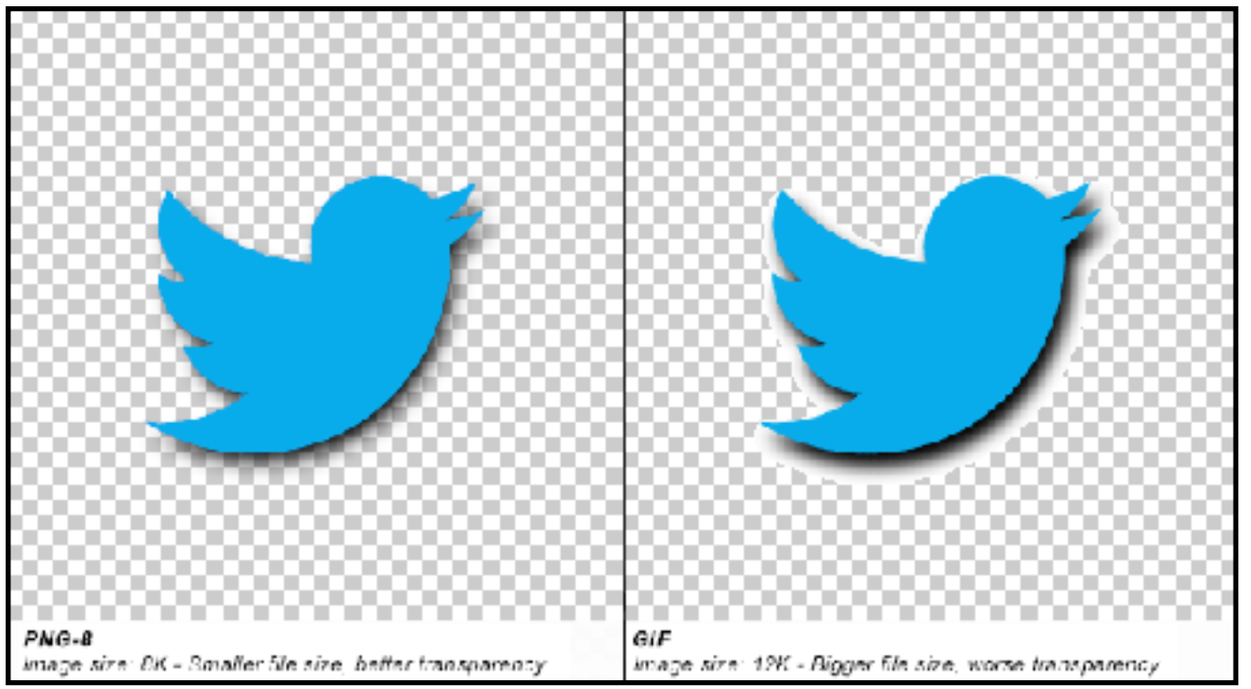


image credit: http://stackoverflow.com/questions/2336522/png-vs-gif-vs-jpeg-when-best-to-use

IMAGE FORMATS: GIF

- Can have basic transparency but does not support semi-transparency
- Png is preferable unless the image meets **both** these conditions:
    1. Semi-transparency is not needed
    2. The gif format is *significantly* smaller than the png format

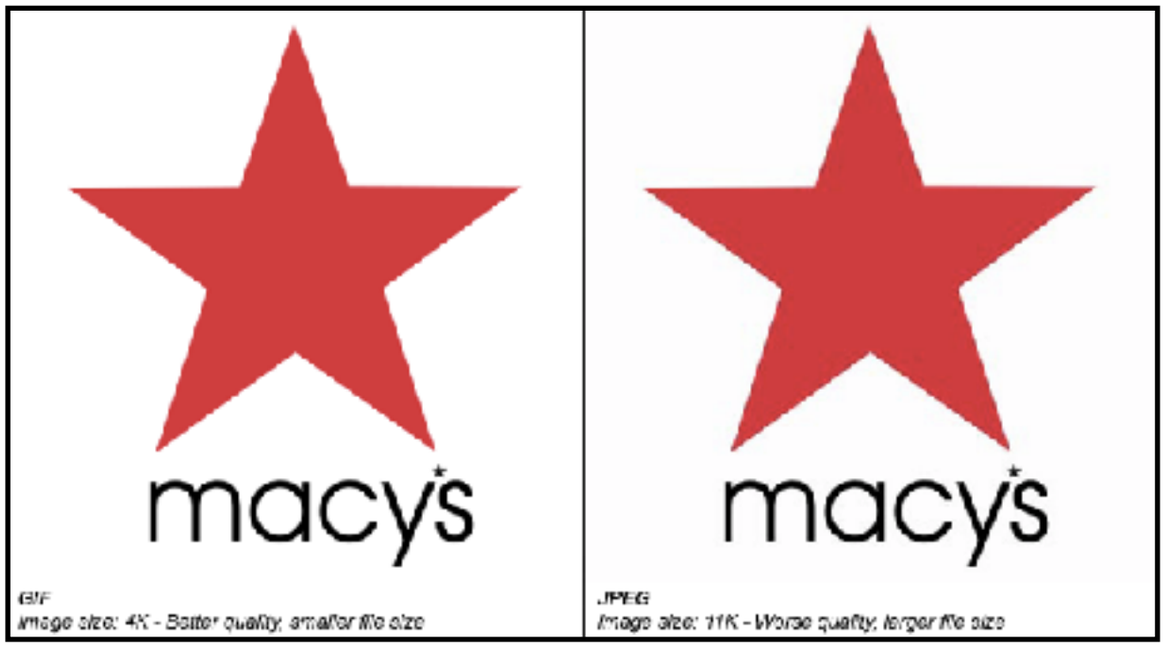


image credit: http://stackoverflow.com/questions/2336522/png-vs-gif-vs-jpeg-when-best-to-use

ACTIVITY

EXERCISE

KEY OBJECTIVE

- Identify which file format should be used for selected images

ACTIVITY

1. 

2. 

3. 

4. 

5. 

6. 

LEARNING OBJECTIVES

- Implement a fluid layout with Bootstrap.
- Understand how to use ARIA roles to make web content more accessible.
- Use jQuery to make a request for data from a web service API
- Add media to a web page by embedding code for a widget
- Choose the most appropriate file format for an image

WEEKLY OVERVIEW

WEEK 9	Final Project Lab / Student Choice
WEEK 10	Final Project Lab / Presentations

EXIT TICKETS!