Building User Interfaces

Javascript
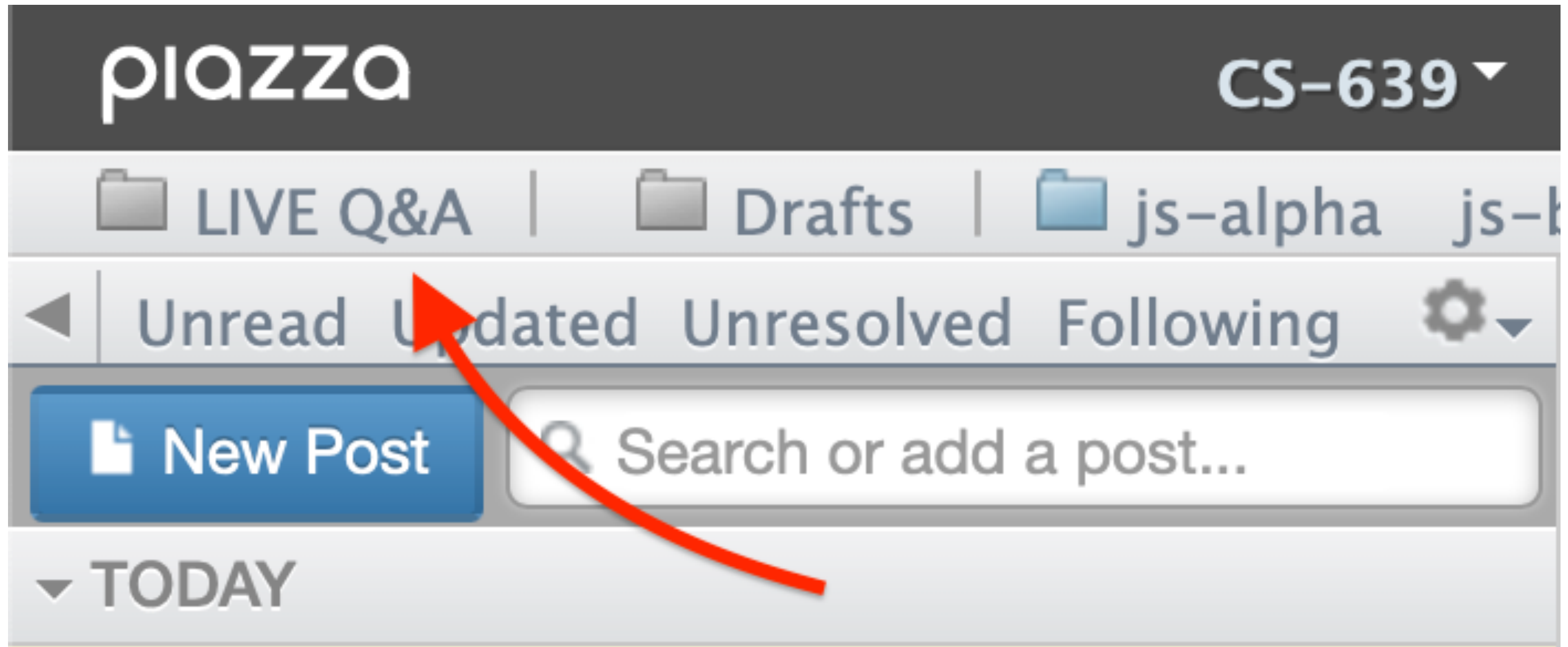
Intermediate Concepts

Professor Bilge Mutlu

What we will learn today?

— Working with JSON data

— \<div\>, CSS/No-CSS

— Working with APIs

— Working with component libraries

Live Q&A Reminder

Working with JSON data

What is JSON?

Definition: JavaScript Object Notation (JSON) is a structured way to represent text-based data based on JS object syntax.

JSON can include any JS data type. Do you remember how many types there are? (7 types: number, string, boolean, null, undefined, symbol, object)

```
{ string : value, .......}
```

- JavaScript object in text form, useful for saving and sending data in a simple format

Refresher: JS Objects

Definition: Objects are unordered collection of related data of primitive or reference types.

Object elements are defined using `key: value` statements.

```
var teachingAssistant = {
    firstName: "Cole",
    lastName: "Nelson",
    age: 26
}
teachingAssistant;
> {firstName: "Cole", lastName: "Nelson", age: 26}
```

JSON Objects:

```
{ "firstName": "Cole",
  "lastName": "Nelson",
  "role": "TA",
  "email": "ctnelson2@wisc.edu" }
```

JSON Arrays:

```
{ "TAs" : [
 (0){ "Name": "Cole Nelson" , "Year": "First" },
 (1){ "Name": "John Balis" , "Year": "First" },
 (2){ "Name": "Derek Manning" , "Year": "First" }]}
```

How to use JSON data[1]

Text, defined in JavaScript

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

var text = '{ "TAs": [' +
  '{ "Name":"Cole Nelson" , "Year":"First" },' +
  '{ "Name":"John Balis" , "Year":"First" },' +
   '{ "Name":"Derek Manning" , "Year":"First" }]}';

obj = JSON.parse(text);

document.getElementById("TANames").innerHTML =
  "Our TAs are " + obj.TAs[0].Name +
  " and " + obj.TAs[1].Name + ".";
```

String concatenation

[1] See a working example in CodePen

How to request JSON from a server[2]

- Most of the time you will request data in this way

— Requests can be synchronous or asynchronous.

— asynchronous requests are recommended as they produce a *callback* when the data is received and lets the browser continue its work while the request is made.

[2] More on Synchronous/asynchronous Requests

Slight Detour: Callback Functions[3]

Definition: A *callback function* is passed into another function as an argument, which is then invoked inside the outer function to complete a routine or action.

```javascript
function greeting(name) {
    alert('Hello ' + name);
}

function processUserInput(callback) {
    var name = prompt('Please enter your name.');
    callback(name);
}

processUserInput(greeting);
```

greeting() is passed into processUserInput() to then be called within that function
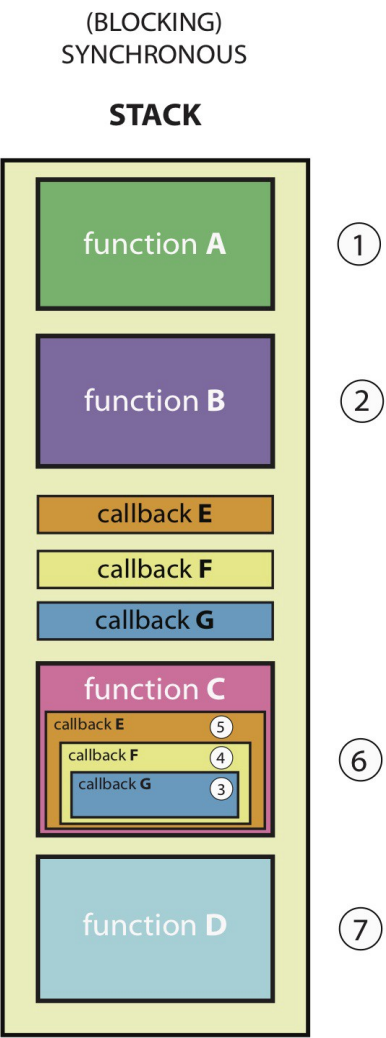
[3] More on callback Functions

© Building User Interfaces | Professor Mutlu | Lecture 03 — *Javascript 2: Intermediate Concepts*   10

Methods for Asynchronous Requests

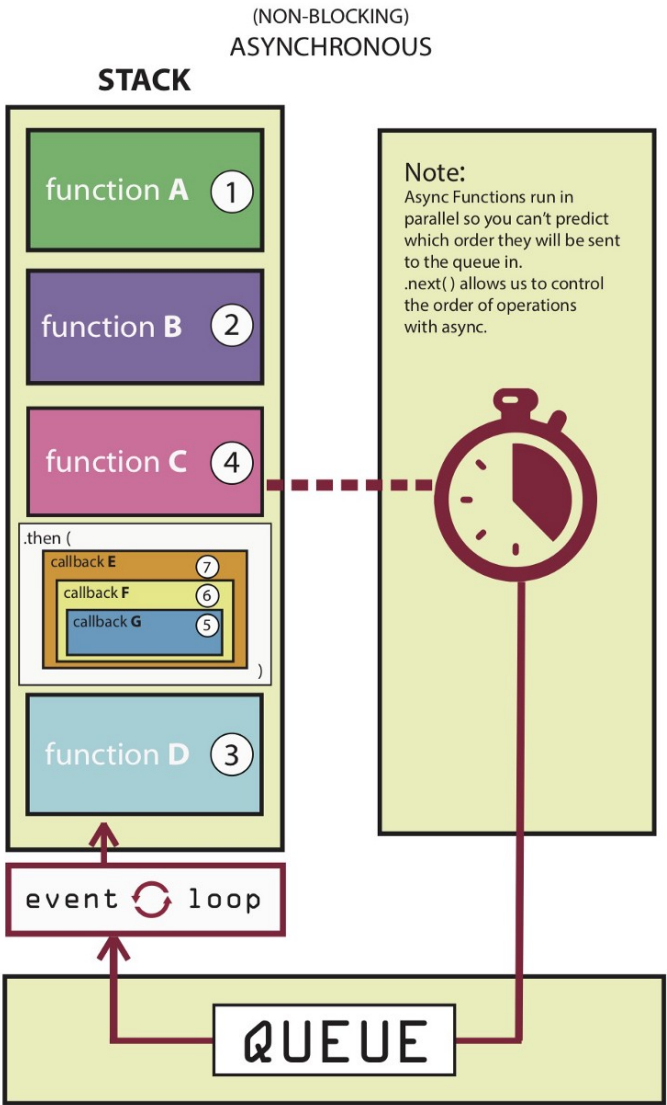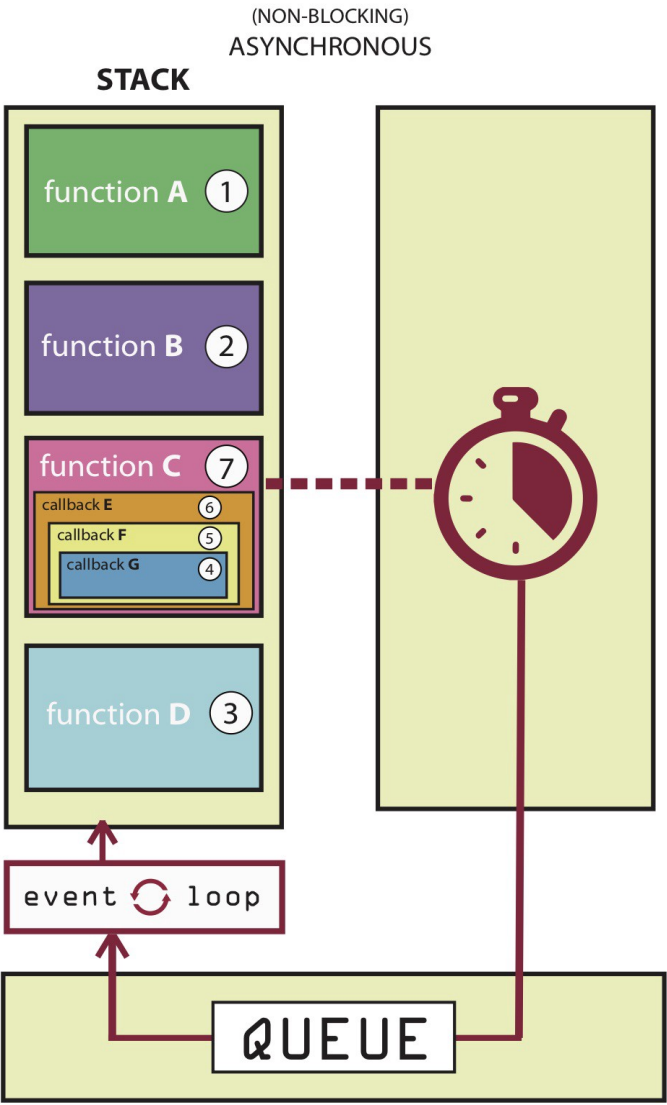Two key methods: `XMLHttpRequest()` (old) and `fetch()` (new)

Pro Tip: `fetch()` is a `Promise`-based method.

— `Promise` objects represent the eventual completion/failure of an *asynchronous* operation and its resulting value.

— `async / await` — keywords to indicate that a function is *asynchronous* 👈 preferred method

— We'll cover these in-depth in React.

(BLOCKING)
SYNCHRONOUS

STACK

function **A** ①

function **B** ②

callback **E**
callback **F**
callback **G**

function **C**
callback E ⑤
callback F ④
callback G ③ ⑥

function **D** ⑦

Function D has to wait till the very end to run, each function blocks while waiting for requested data

(NON-BLOCKING)
ASYNCHRONOUS

STACK

function **A** ①

function **B** ②

function **C** ⑦
callback E ⑥
callback F ⑤
callback G ④

function **D** ③

event ↻ loop

QUEUE

Here Function D is able to run while data is being fetched or input is being entered

(NON-BLOCKING)
ASYNCHRONOUS

STACK

function **A** ①

function **B** ②

function **C** ④
.then (
callback E ⑦
callback F ⑥
callback G ⑤
)

function **D** ③

event ↻ loop

QUEUE

Note:
Async Functions run in parallel so you can't predict which order they will be sent to the queue in.
.next( ) allows us to control the order of operations with async.

[4] Source

XMLHttpRequest() [5]

```
var requestURL = 'tas.json';
var request = new XMLHttpRequest();
request.open('GET', requestURL, true); // true for asynchronous
request.responseType = 'json';
request.send();
```

[5] See a working example in CodePen ⟵ Try these examples on your own!

fetch() [6]

- Promise based method

```
fetch(url) Fetch this please…
    .then(response => response.json())…when you get it do this…
    .then(data => { …then this…
        // Do something with the data
    })
    .catch(error => console.error(error)) // Print errors
```

…and catch any errors along the way.

[6] See a working example in CodePen

Back to JSON: `parse` and `stringify`

`parse()` takes a JSON string and returns JS objects.

```
var tas = JSON.parse(request.response);
```
Take response string and return JavaScript object

`stringify()` takes a JS object and returns JSON string.

```
var tas = { "name": "Chris", "age": "38" };
var tasJSON = JSON.stringify(tas);
```
Get a string in the end, to send somewhere potentially or just to store

Accessing JS objects from JSON data

```
{ "firstName": "Cole", "lastName": "Nelson",
"role": "TA", "email":"ctnelson2@wisc.edu" }
```

```
         Object                              String
var myTA = JSON.parse(request.response);
console.log(myTA.firstName);
console.log(myTA["firstName"]);
```

Quiz 1

Complete the Canvas quiz.

Using JS to render content

DOM Container

(Document Object Model)

Definition: `<div>` defines a "division" or a section in an HTML document. You can place `<div>`s anywhere on the page and as many as you like. They will serve as canvases to manipulate using JS/React.

Prototype declaration: `<div id="name"></div>`

- Used to make up the sections of a webpage…
you will see and use these a lot

CSS[7]

Consider the following button:

```
<button id="button">Submit</button>
```

We can use CSS to style it:

```
button {
    background-color: #008CBA;
    border: none;
    color: white;
    padding: 15px 32px;
    font-size: 16px; }
```

- Tedious to change in a dynamic way after we set it though

[7] See live at CodePen

No CSS[8]

Consider the same button:

```
<button id="button">Submit</button>
```

We can also style it using JS:

- JavaScript makes it easy to dynamically change elements and styling using functions and flow control within its scripts

Transform HTML into JS object…   …then CSS attributes can be accessed and modified like any JS object

```
document.getElementById("button").style.color = "white";
document.getElementById("button").style.padding = "15px 32px";
document.getElementById("button").style.border = "none";
document.getElementById("button").style["background-color"] = "#008CBA";
document.getElementById("button").style["font-size"] = "16px";
```

[8] See live at CodePen

Working with APIs

What are APIs for Web Development?

Definition: Application Programming Interfaces (APIs) are constructs that facilitate the programming of complex functionality.

APIs abstract away the low-level implementation of tools and services and provide the programmer with easier syntax.

- Gives you a high-level, simple interface to use

How do APIs work?

Browser APIs (e.g., fullscreen API, screen orientation API, vibration API), vs. **third-party APIs** (e.g., Google Maps API, Twitter API).

JS interacts with APIs over JS objects.

- Each API gives you an object to create, and associated methods within each

An Example [9] [10]

Play an `mp3` file using the *Audio API*:

1. Create the audio and control elements — HTML
2. Create an *audio context* — JS
3. Create an audio element — JS
4. Control the element — JS

[9] See live at CodePen

[10] The version on CodePen will not play the audio due to Cross-Origin Resource Sharing (CORS) errors

Step 1: Create elements

```
<audio src="Haydn_Adagio.mp3" type="audio/mpeg"></audio>
<button data-playing="false" role="switch" aria-checked="true">
    <span>Play | Pause</span>
</button>
```

Step 2: Create an audio context

```javascript
const audioContext = new AudioContext();
```

Step 3: Create an audio element

```javascript
const audioElement = document.querySelector('audio');

const track = audioContext.createMediaElementSource(audioElement);

track.connect(audioContext.destination);
```

Step 4: Control the element

```javascript
playButton.addEventListener('click', function() {
    if (audioContext.state === 'suspended') { audioContext.resume();}
    if (this.dataset.playing === 'false') {
        audioElement.play();
        this.dataset.playing = 'true';
        console.log("Playing...");
    } else if (this.dataset.playing === 'true') {
        audioElement.pause();
        this.dataset.playing = 'false';
        console.log("Stopped..."); }
}, false);

audioElement.addEventListener('ended', () => {
    playButton.dataset.playing = 'false';
}, false);
```

Working with Component Libraries

What are Component Libraries?[11]

Definition: Software libraries that abstract away the low-level CSS implementation of user-facing elements.

Some popular libraries:
* Bootstrap
* Foundation
* Semantic UI
* Pure
* UIkit

- Provides easy, consistent interface for styling

- Think of these as ready made CSS blocks that you can apply wherever you want

[11] A comparison of the frameworks

Bootstrap

— Download for offline development

`$ npm install bootstrap`

— <u>BootstrapCDN</u> (Content Delivery Network)

```html
<link
    rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
    crossorigin="anonymous">
<script
    src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
    integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
    crossorigin="anonymous">
</script>
```
(Grabs script from CDN on page load)

How Bootstrap Works

Main categories of HTML specification:
* Layouts   (Structure, organization)
* Content   (The web content)
* Components (Styled object that shows content)
* Utilities   (Preconfigured stylings)

There is much more!

Bootstrap Categories: Layouts

— **Containers** are the most basic element of layouts.

   — *Responsive, fixed-width, fluid-width.*

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


<div class="container-fluid">
  ...
</div>
```
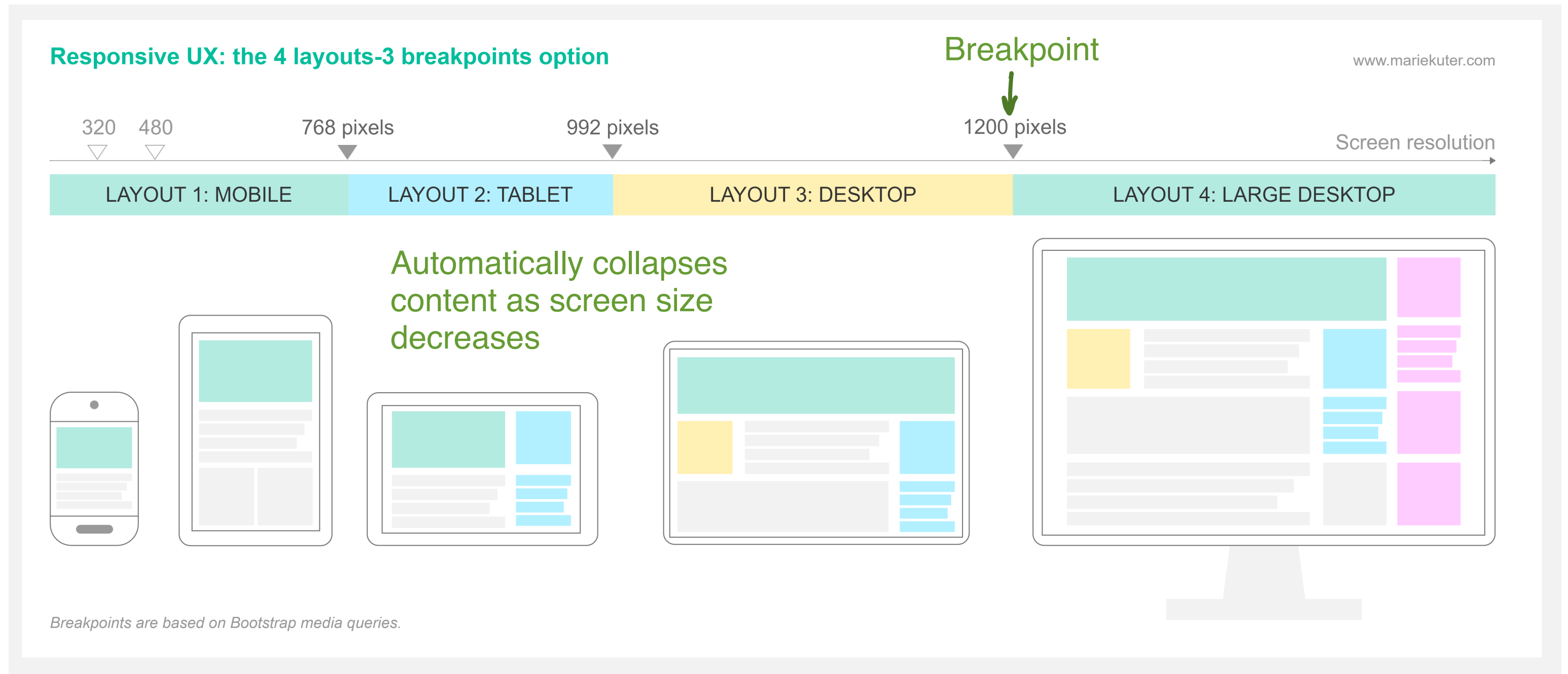
Layouts: Responsive Design[12]

Definition: Responsive web design (RWD) is an approach that adapts web content to a variety of devices and window or screen sizes.[13]

Width breakpoints determine whether the design will scale or be reorganized.

[12] Wikipedia: Responsive Web Design

[13] Image Source: InVision

Responsive UX: the 4 layouts-3 breakpoints option

Breakpoint

www.mariekuter.com

320 480 768 pixels 992 pixels 1200 pixels

Screen resolution

LAYOUT 1: MOBILE   LAYOUT 2: TABLET   LAYOUT 3: DESKTOP   LAYOUT 4: LARGE DESKTOP

Automatically collapses content as screen size decreases

Breakpoints are based on Bootstrap media queries.

[14] Image Source: Marie Kuter

How does Bootstrap do this?[15]

px = pixels

```
// Extra small devices (portrait phones, less than 576px)
// No media query for `xs` since this is the default in Bootstrap

// Small devices (landscape phones, 576px and up)
@media (min-width: 576px) { ... }

// Medium devices (tablets, 768px and up)
@media (min-width: 768px) { ... }

// Large devices (desktops, 992px and up)
@media (min-width: 992px) { ... }

// Extra large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }
```

[15] Bootstrap Layout Overview

Detour: Responsive Layouts using CSS Flexbox

Definition: A CSS layout mode for responsive content.[16] [17] [18]
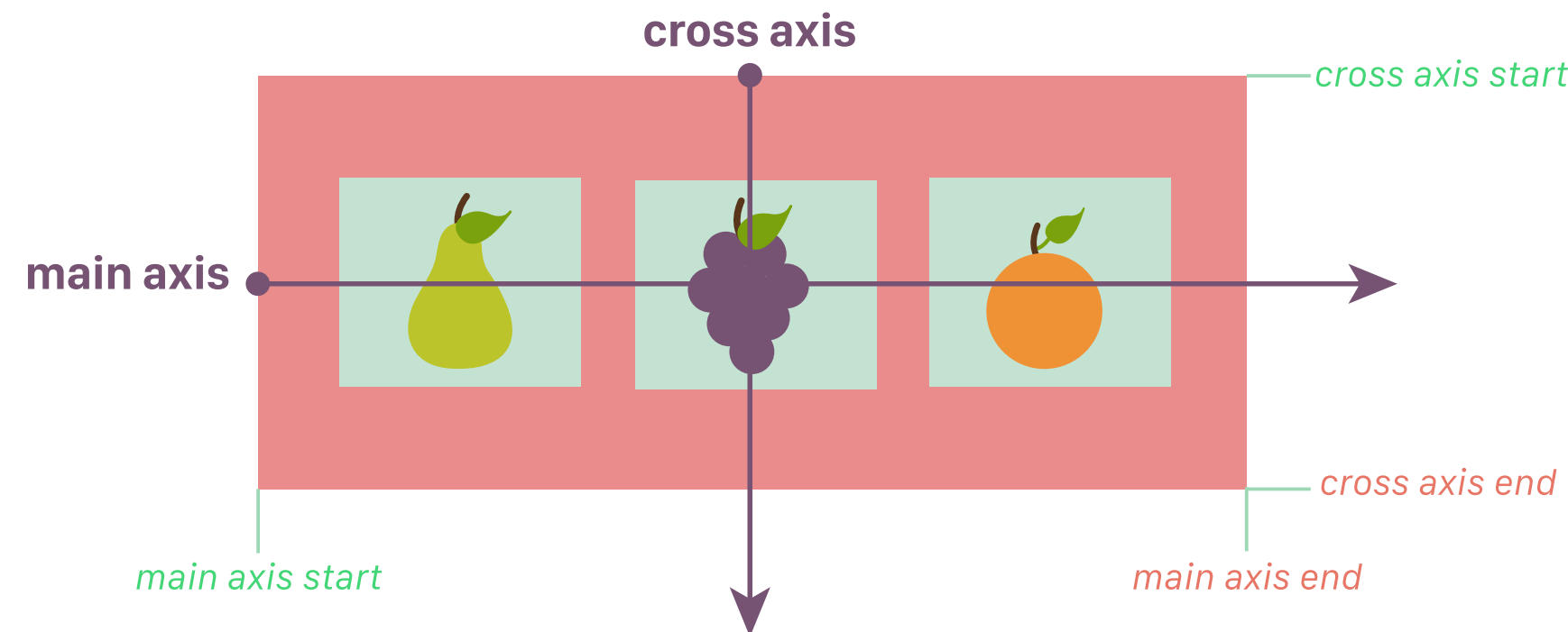
```css
.flex-container {
    display: flex;
}
```

```html
<div class="flex-container">
  <div>Content A</div>
  <div>Content B</div>
  <div>Content C</div>
</div>
```

---

[16] Excellent Flexbox Cheatsheet

[17] See example on CodePen   Check these out!

[18] Image source

Quiz 2

Complete the Canvas quiz.

Layouts: **Grids**

Basic usage:

```
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
```

Where the first * is *grid class*.

The Bootstrap grid system classes:[19]

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Bootstrap
[19] Bootsrap grid

Second * is the number of grid columns (max = 12). [20] [21]

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1

span 4	span 4	span 4

span 4	span 8

span 6	span 6

span 12

```html
<div class="row">
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-4">.col-sm-4</div>
</div>
```

[20] W3 Schools: Bootstrap

[21] See in CodePen

Bootstrap Categories: Content

Content styling includes basic HTML elements, typography, code, images, tables, figures.

Basic HTML examples:

```
<h1></h1>
<ul></ul>
<input></input>
<button></button>
```

Using class you can string together multiple classes for compound styling

Pro Tip: Note the possibility of using, e.g., `<h1>` and `class="h1"`.

Styling of other elements

```
<img src="..." class="img-fluid">

<table class="table">        Coloring
  <thead class="thead-dark">
    <tr>
      <th scope="col">...</th>
      ...

<div class="table-responsive-sm">
  <table class="table">
    ...
```

Bootstrap Categories: Components

Components include all other visual/interactive elements that make up the design, e.g., buttons, forms, navbar, tooltips, etc.

```html
<button type="button" class="btn btn-primary">Fill button</button>
```
Bootstrap predefines all of these, checkout their docs
```html
<button type="button" class="btn btn-outline-primary">Outline button</button>
```
Button group
```html
<div class="btn-group-toggle" data-toggle="buttons">
  <label class="btn btn-secondary active">       Button w/ label
    <input type="checkbox" checked autocomplete="off"> Switch
  </label>
</div>
```

Bootstrap Categories: Utilities

Utilities are not elements themselves, but they modify/control other elements, e.g., adding rounded corners to an image.

```html
<img src="..." class="rounded">

<div class="shadow p-3 mb-5 bg-white rounded">Shadow</div>
```

Quiz 3

Complete the Canvas quiz.

Example HomePage[22]
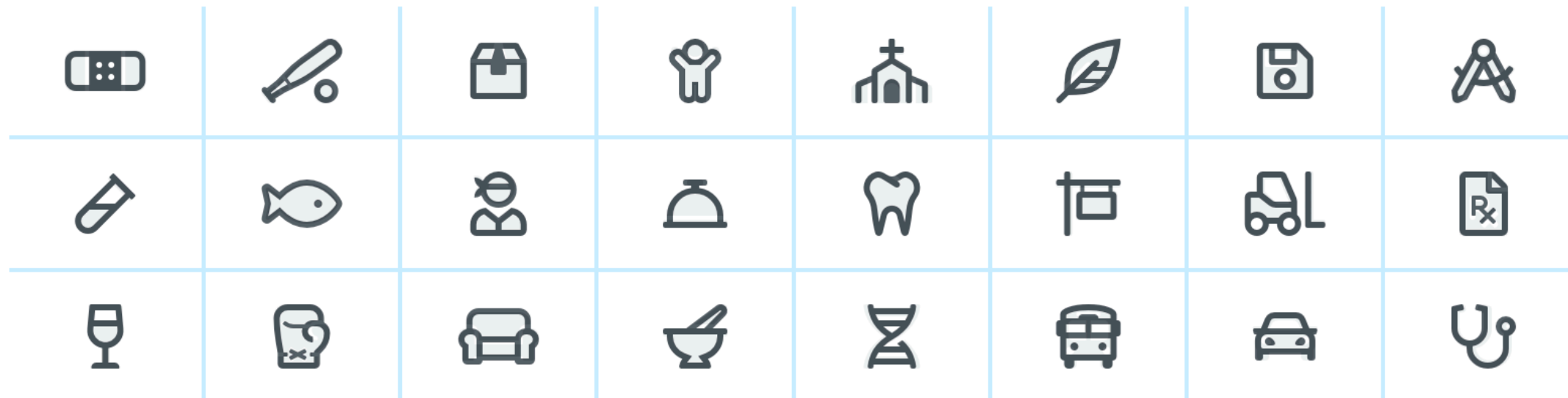
Additional Resources

— <u>Bootstrap documentation</u>

— <u>Tutorial Republic</u>

— <u>W3 Schools</u>

- Be sure to check these out!

Assets

Asset libraries, e.g., icons, are usually used in conjunction with frameworks such as Bootstrap.[23] [24]

[23] Icon libraries

[24] Image source

What we learned today

— Working with JSON data

— \<div\>, CSS/No-CSS

— Working with APIs

— Working with component libraries

Assignment

Javascript α released — due next week, Thursday

— Implement the functionality supporting **Badger Bank**

— In Javascript β, to be released next Wednesday, we will improve on the visual design