

Building User Interfaces

javascript

Intermediate Concepts

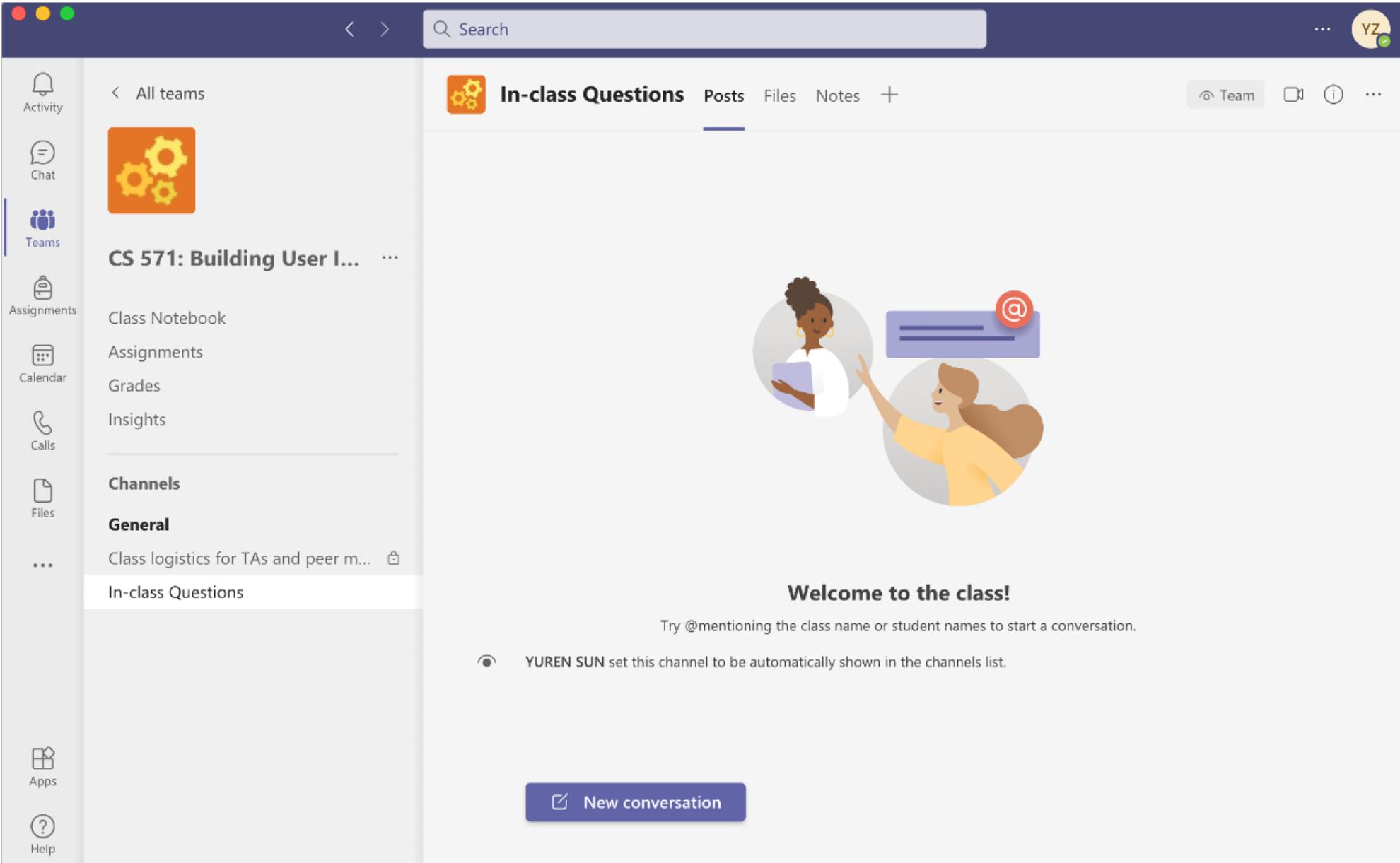
Professor Yuhang Zhao

Adapted from Prof. Mutlu's slides

What we will learn today?

- Working with JSON data
- <div>, CSS/No-CSS
- Working with APIs
- Working with component libraries

Live Q&A Reminder



The screenshot shows the Microsoft Teams interface for the "In-class Questions" channel in the "CS 571: Building User I..." team. The left sidebar lists various team channels: "All teams", "Class Notebook", "Assignments", "Calendar", "Calls", "Files", "General" (which is selected), "In-class Questions", and "Help". The main content area displays a welcome message: "Welcome to the class! Try @mentioning the class name or student names to start a conversation." It features two cartoon characters, one holding a laptop and the other pointing, with a speech bubble containing an '@' symbol. A blue button at the bottom right says "New conversation". The top navigation bar includes a search bar, user profile, and team settings.

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?

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

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 instructor = {  
  firstName: "Yuhang",  
  lastName: "Zhao",  
  gender: "female"  
}  
instructor;  
> {firstName: "Yuhang", lastName: "Zhao", gender: "female"}
```

JSON Objects:

```
{ "firstName": "Yuhang",  
  "lastName": "Zhao",  
  "role": "instructor",  
  "email": "yuhang.zhao@cs.wisc.edu" }
```

JSON Arrays:

```
{ "TAs" : [  
    { "Name": "Brandon" , "Year": "First" },  
    { "Name": "Sujitha" , "Year": "First" },  
    { "Name": "Salman" , "Year": "First" }]}
```

How to use JSON data¹

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

```
var text = '{ "TAs": [ ' +
  '{ "Name": "Brandon Cegelski" , "Year": "First" },' +
  '{ "Name": "Sujitha Perumal" , "Year": "First" },' +
  '{ "Name": "Salman Munaf" , "Year": "First" }] }';
```

```
obj = JSON.parse(text);
```

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

¹ See a working example in [CodePen](#)

How to request JSON from a server²

- 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 Functions³

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.

```
function greeting(name) {  
    alert('Hello ' + name);  
}  
  
function processUserInput(callback) {  
    var name = prompt('Please enter your name. ');  
    callback(name);  
}  
  
processUserInput(greeting);
```

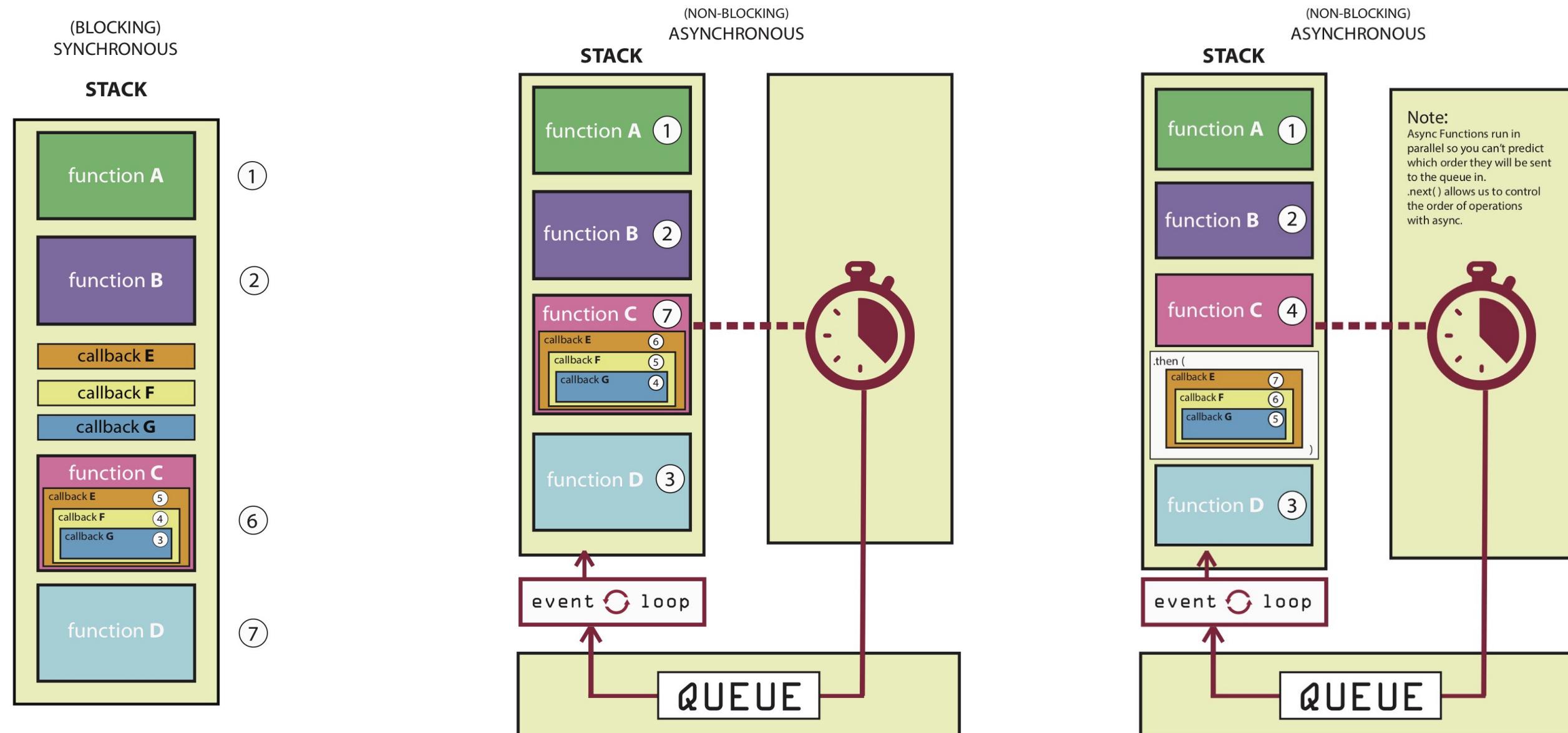
³ More on [callback Functions](#)

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.



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](#)

fetch()⁶

```
fetch(url)
  .then(response => response.json())
  .then(data => {
    // Do something with the data
  })
  .catch(error => console.error(error)) // Print errors
```

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);
```

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

```
var tas = { "name": "Chris", "age": "38" };
var tasJSON = JSON.stringify(tas);
```

Accessing JS objects from JSON data

```
{ "firstName": "Brandon", "lastName": "Cegelski",  
"role": "TA", "email": "bmcegelski@wisc.edu" }
```

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

Using JS to render content

DOM Container

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>
```

CSS⁷

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; }
```

⁷ See live at [CodePen](#)

No CSS⁸

Consider the same button:

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

We can also style it using JS:

```
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.

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.

An Example⁹

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](#)

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

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

Step 3: Create an audio element

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

const track = audioContext.createMediaElementSource(audioElement);

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

Step 4: Control the element

```
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

¹¹ A comparison of the frameworks

Bootstrap

- Download for offline development

```
$ npm install bootstrap
```

- BootstrapCDN (Content Delivery Network)

```
<link  
  rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  
  integrity="sha384-gg0yR0iXcbMqv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"  
  crossorigin="anonymous">  
<script  
  src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"  
  integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFF/nJGzIxFDsf4x0xIM+B07jRM"  
  crossorigin="anonymous">  
</script>
```



How Bootstrap Works

Main categories of HTML specification:

- * Layouts
- * Content
- * Components
- * Utilities

There is much more!

Bootstrap Categories: [Layouts](#)

- Containers are the most basic element of layouts.
 - *Responsive, fixed-width, fluid-width.*

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

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

Layouts: Responsive Design¹²

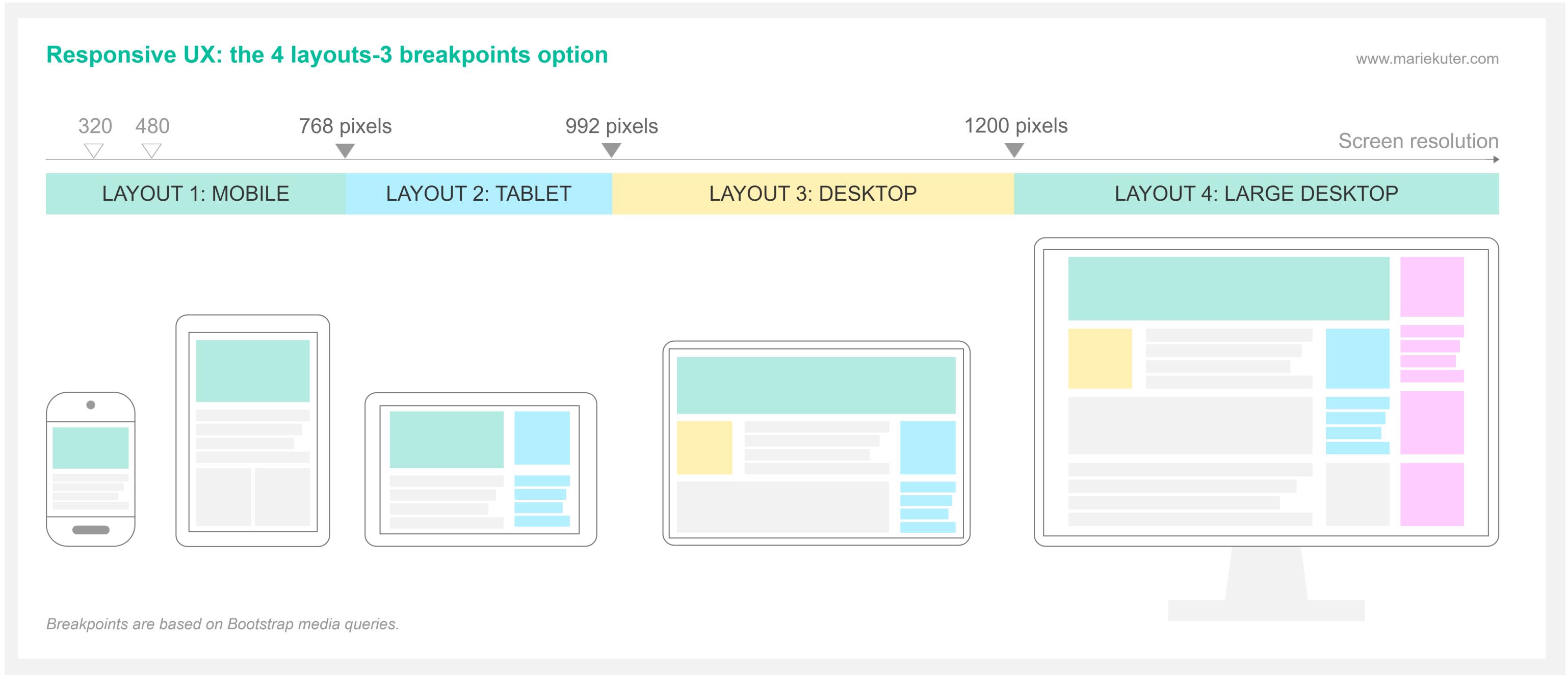
Definition: Responsive web design (RWD) is an approach that adapts web content to a variety of devices and window or screen sizes.¹³

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



¹² Wikipedia: Responsive Web Design

¹³ Image Source: [InVision](#)



¹⁴ Image Source: [Marie Kuter](#)

How does Bootstrap do this?¹⁵

```
// 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) { ... }

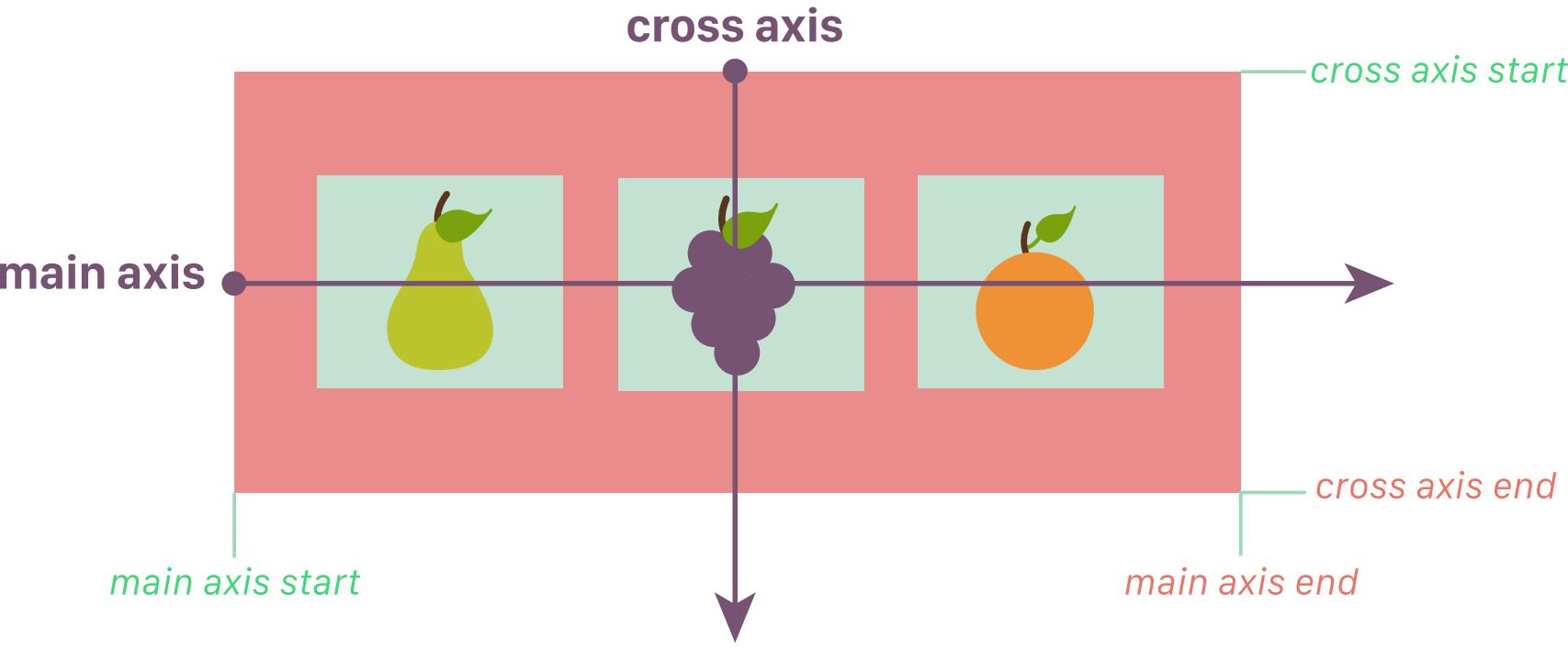
// Extra large devices (large desktops, 1200px and up)
@media (min-width: 1200px) { ... }
```

¹⁵ [Bootstrap Layout Overview](#)

Detour: Responsive Layouts using CSS Flexbox

Definition: A CSS layout mode for responsive content.^{16 17 18}

```
.flex-container {  
    display: flex;  
}  
  
<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](#)

¹⁸ [Image source](#)

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				

¹⁹ [Bootstrap grid](#)

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



```
<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>
```

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

Styling of other elements

```

```

```
<table class="table">
  <thead class="thead-dark">
    <tr>
      <th scope="col">...</th>
    ...
  </thead>
  <tbody>
    <tr>
      <td>...</td>
    </tr>
  </tbody>
</table>
```

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

Bootstrap Categories: Components

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

```
<button type="button" class="btn btn-primary">Fill button</button>
```

```
<button type="button" class="btn btn-outline-primary">Outline button</button>
```

```
<div class="btn-group-toggle" data-toggle="buttons">
  <label class="btn btn-secondary active">
    <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.

```

```

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

Example HomePage²²

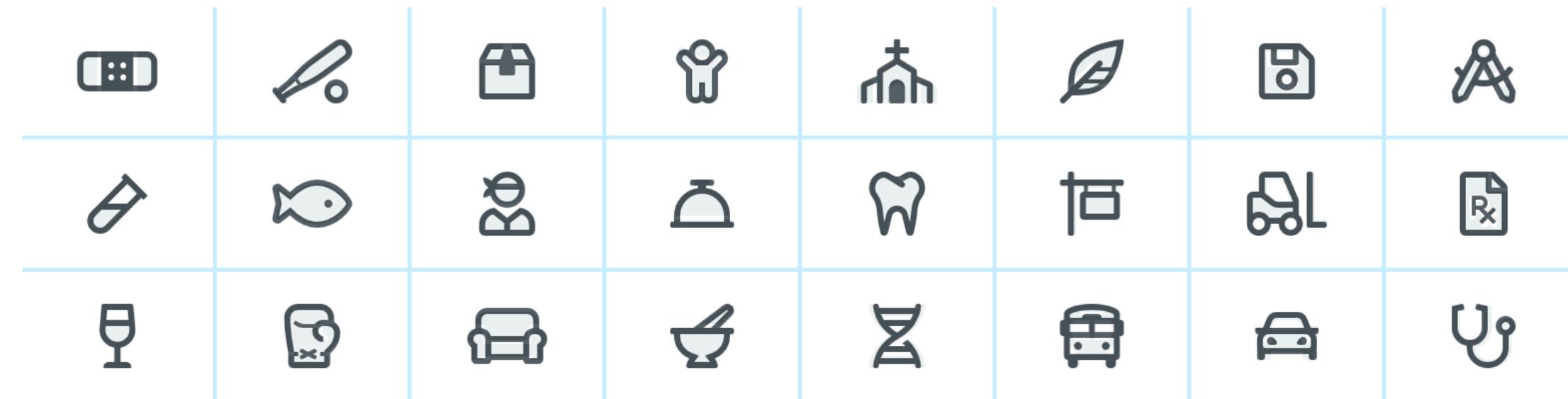
²² See in [CodePen](#)

Additional Resources

- Bootstrap documentation
- Tutorial Republic
- W3 Schools

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, Friday

- Implement the functionality supporting Badger Shop
- In Javascript β , to be released next Tuesday, we will improve on the visual design