JavaScript 2

CS571: Building User Interfaces

Cole Nelson

What we will learn today?

- Working with JSON data
- Working with APIs
- Working with other async functions
- Working with Bootstrap and CSS libraries

What is JSON?

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

Refresher: JS Objects

Definition: Objects are unordered collection of related data of primitive or reference types defined using key-value pairs.

```
const instructor = {
  firstName: "Cole",
  lastName: "Nelson",
  roles: ["student", "faculty"]
}
```

JSON Equivalent

```
{
  "firstName": "Cole",
  "lastName": "Nelson",
  "roles": ["student", "faculty"]
}
```

What's the difference? A JS Object is executable code; JSON is a language-agnostic representation of an object. There are also slight differences in syntax.

You can write comments in JS Objects...

```
const drinks = [
  {
    name: "Mimosa",
    ingredients: [
      {name: "Orange Juice", hasAlcohol: false},
      {name: "Champagne", hasAlcohol: true}
    ]
  },
  {
    name: "Vesper Martini", // shaken, not stirred
    ingredients: [
      {name: "Gin", hasAlcohol: true},
      {name: "Vodka", hasAlcohol: true},
      {name: "Dry Vermouth", hasAlcohol: true},
    ]
  }
]
```

... but not in JSON!

```
[
  {
    "name": "Mimosa",
    "ingredients": [
      { "name": "Orange Juice", "hasAlcohol": false },
      { "name": "Champagne", "hasAlcohol": true }
    ]
  },
  {
    "name": "Vesper Martini",
    "ingredients": [
      { "name": "Gin", "hasAlcohol": true },
      { "name": "Vodka", "hasAlcohol": true },
      { "name": "Dry Vermouth", "hasAlcohol": true }
    ]
  }
]
```

Conversion

Because JS Objects and JSON are so similar, it is easy to convert between them.

- `JSON.parse` JSON String → JS Object
- `JSON.stringify` JS Object → JSON string

Conversion Examples

Using `JSON.parse` and `JSON.stringify`.

```
const myObj = JSON.parse('{"name": "Cole", "age": 24}');
const myStr = JSON.stringify(myObj);

console.log(typeof myObj);
console.log(typeof myStr);
```

```
object
string
```

Data Copying

`json.parse` and `json.stringify` can also be useful for deep data copying.^

^ lodash is the preferred way to copy.

Data Copying - Reference Copy

```
let myBasket = {
  basketId: 154,
  items: ["Apples", "Bananas", "Grapes"]
};
let myRefCopyBasket = myBasket;
myRefCopyBasket.basketId = 999;

console.log(myBasket);
console.log(myRefCopyBasket);
```

```
{basketId: 999, items: ['Apples', 'Bananas', 'Grapes']}
{basketId: 999, items: ['Apples', 'Bananas', 'Grapes']}
```

Data Copying - Deep Copy

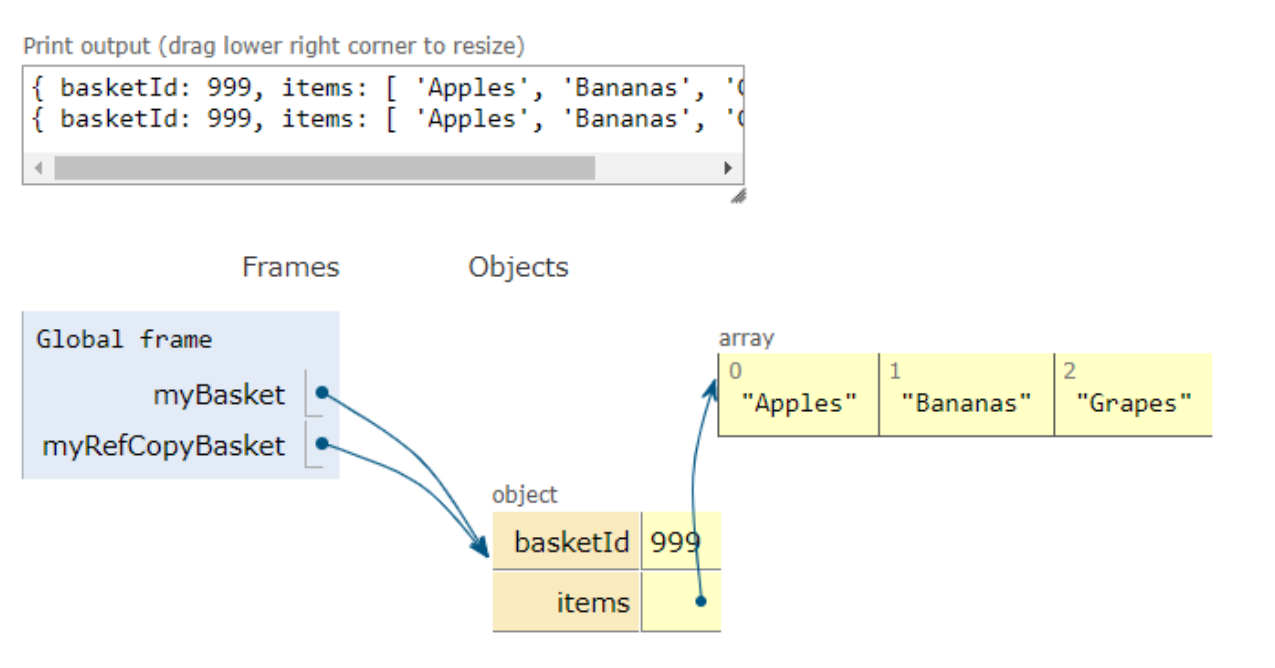
```
let myBasket = {
  basketId: 154,
  items: ["Apples", "Bananas", "Grapes"]
};
let myRefCopyBasket = JSON.parse(JSON.stringify(myBasket));
myRefCopyBasket.basketId = 999;

console.log(myBasket);
console.log(myRefCopyBasket);
```

```
{basketId: 154, items: ['Apples', 'Bananas', 'Grapes']}
{basketId: 999, items: ['Apples', 'Bananas', 'Grapes']}
```

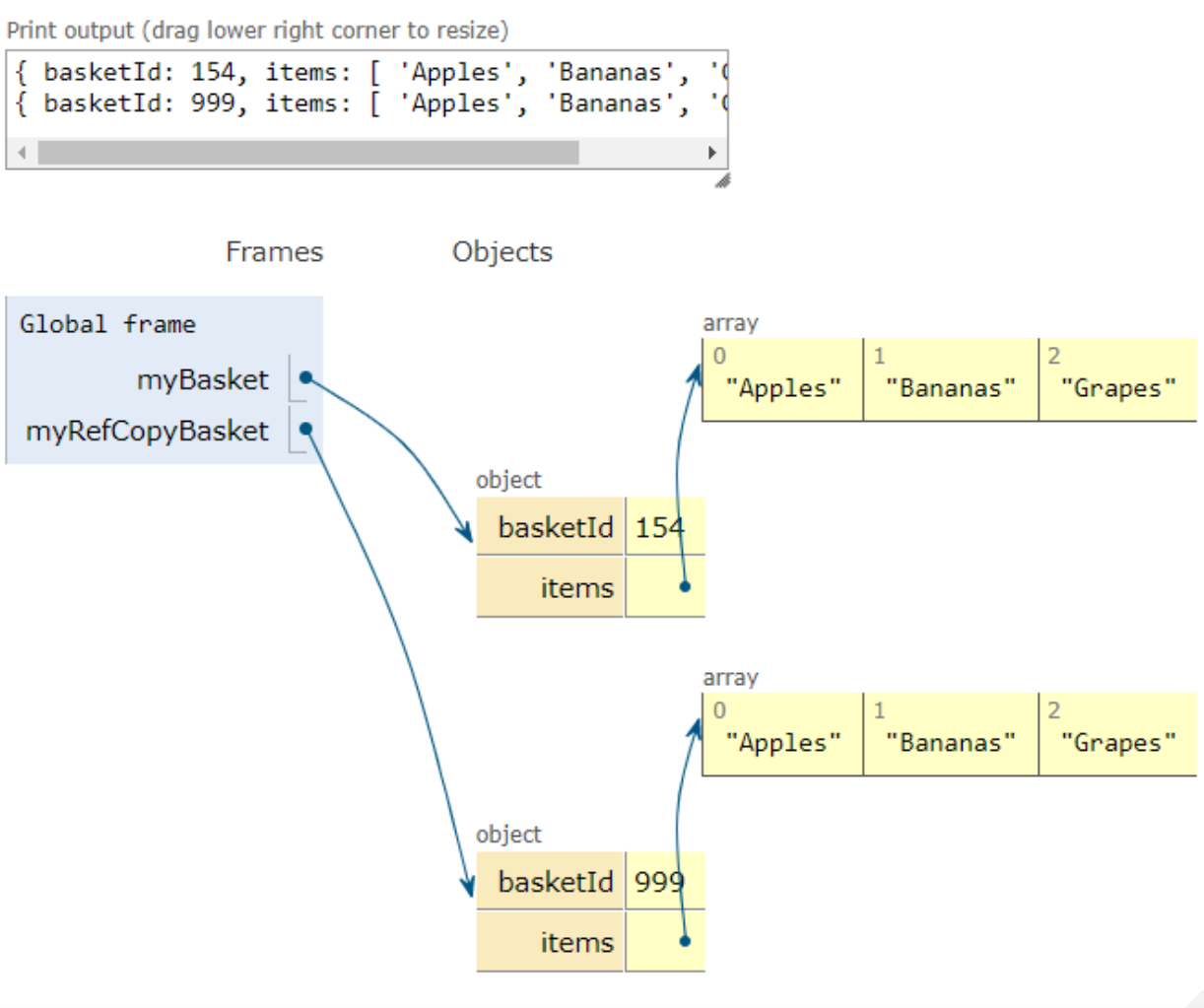
Reference Copy Visualization

Interactive Example



Deep Copy Visualization

Interactive Example



What is an API?

Definition: An application programming interface (API) is a set of definitions and protocols for communication through the serialization and de-serialization of objects.

JSON is a language-agnostic medium that we can serialize to and de-serialize from!

Request for JSON

- Requests can be `synchronous` or `asynchronous`.
- `asynchronous` requests are recommended as they are *non-blocking*. Typically, they use a *callback* when the data is received and lets the browser continue its work while the request is made.

More on synchronous/asynchronous requests

Making Asynchronous Requests

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

fetch()

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

Fetching Jokes API

Fetching Jokes Code

Callback Functions

`then` and `catch` take a *callback function* as an argument.

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

More on callback functions

Callback Functions

```
function greeting1(name) {
  alert('Hello ' + name);
}

function getting2(name) {
  alert('Welcome ' + name);
}

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

processUserInput(greeting1);
processUserInput(greeting2);
```

Using `fetch` with Callbacks

Compare the following...

- Fetching Jokes with Anonymous Function
- Fetching Jokes with Callback Function

Why use one versus the other?

Other `async` Functions

- `setInterval(callback, interval)` perform a callback function every interval milliseconds.*
- `setTimeout(callback, timeout)` perform a callback function in timeout milliseconds.*

Fetch Jokes (w/ `setInterval`)

Fetch Jokes (w/ `setInterval` and `setTimeout`)

* approximately

Your turn!

Use the HW2 API to display information about book(s)!

https://www.coletnelson.us/cs571/f22/hw2/api/book

https://www.coletnelson.us/cs571/f22/hw2/api/books?amount=12

Explore the API, then create a bookstore landing page!

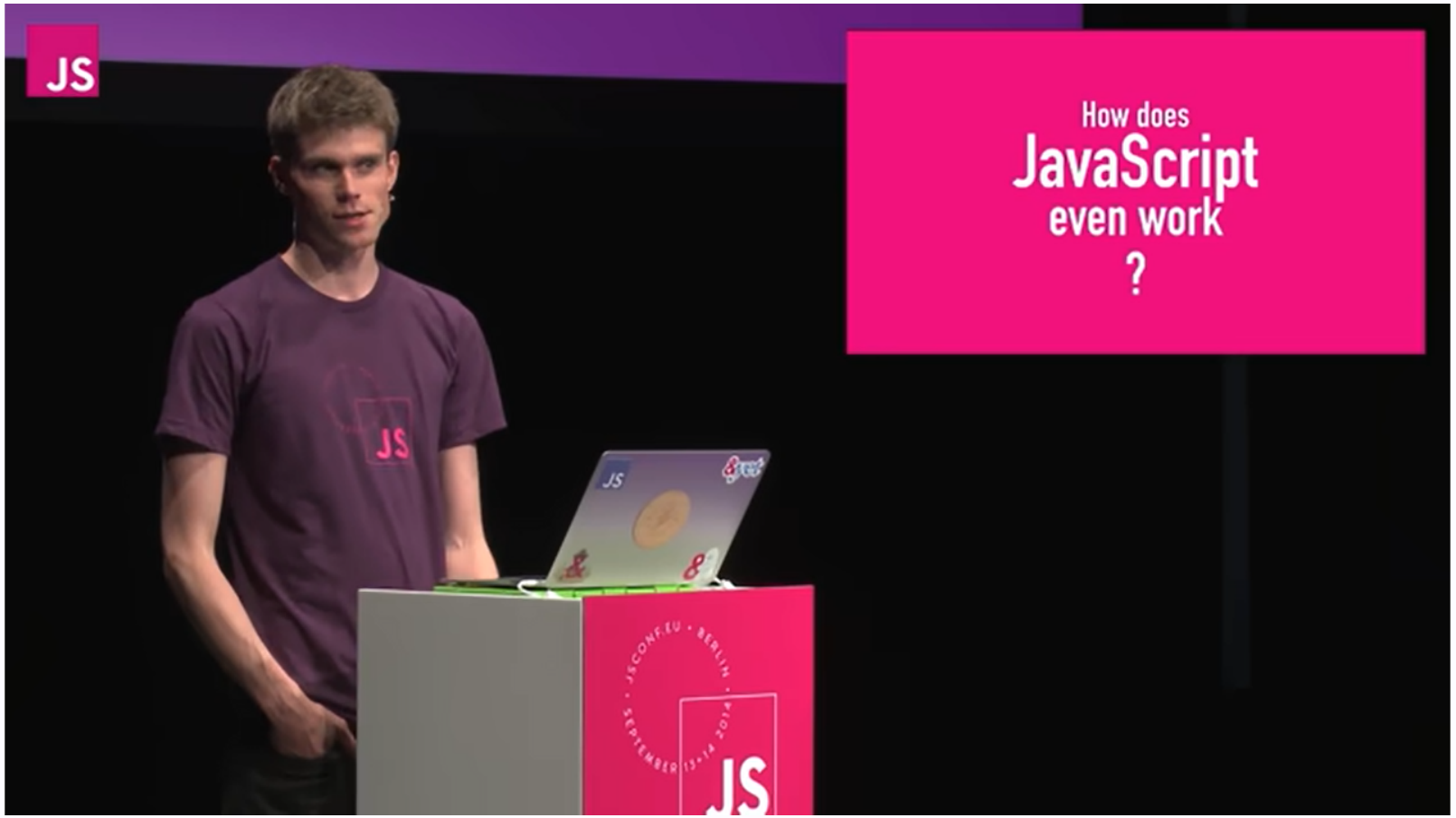
The JS Event Loop

Wait, but JavaScript is a high-level, **single-threaded**, garbage-collected, interpreted, prototype-based, multi-paradigm, dynamic language with **a non-blocking event loop**^...

So... how can we have this *async* behavior?!

We'll come back to this!

^ JavaScript in 100 seconds



What the hell is the event loop anyway?

Working with CSS Libraries

What are CSS Libraries?

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

Some popular libraries include...

- Bootstrap
- Foundation
- Semantic UI
- Pure
- UIkit

Bootstrap

getbootstrap.com

How Bootstrap Works

Bootstrap provides us with...

- Layouts
- Content
- Components
- Utilities

There is much more!

Responsive Design

Definition: Responsive design adapts content to a variety of devices and screen sizes.

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

InVision

Bootstrap Categories: Layouts

Containers are the most basic element of layouts.

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

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

```
<div class="container-{breakpoint}"> <!-- xs, sm, md, lg, xl-->
  ...
</div>
```

Containing a Grid

Basic usage of a grid...

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

Where `*` is *grid class* and `^` is *column size*.

	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 Categories: Content

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

Basic HTML examples:

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

These will get the default Bootstrap styling.

Styling of other elements

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

```
<table class="table">
  <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.

```
<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.

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

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

Example Home Page

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.

See icon libraries.

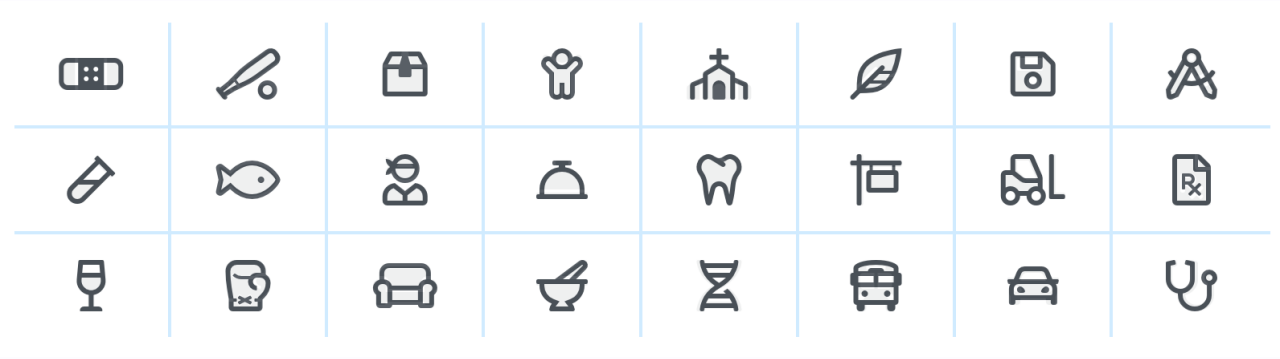


Image Source

What did we learn today?

- Working with JSON data
- Working with APIs
- Working with other async functions
- Working with Bootstrap and CSS libraries

On to Prototyping!