Week 8

205

Week 7 Review

- _____ programming describes what will happen vs how it should happen.
- The .reduce method of an array will do what to an array?
- Which of the following facts about functions are false?
  They can be assigned to a variable
  They can be used as a parameter
  They can be returned from a function
- React is a library for:
  [front end] [back end]
- React uses a [declarative] [imperative] programming methodology

206

React Elements

- React libraries:
  <script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
- A React element defines what is supposed to happen on the page
- React builds/manipulates a "virtual" DOM and then writes it to the page vs the browser DOM which is written directly to the page
- document.createElement and document.appendChild are examples of methods that build the browser DOM
- Create a React element using
  React.createElement()
- Render the element to the page using
  ReactDOM.render()

207

The React Element Object

- Let's say we have an element that will render (display) the following to the page:
  <h1 id="main">Welcome to my Page</h1>
- The corresponding React object looks like the following:

```
{
  $$typeof: Symbol(React.element),
  "type": "h1",
  "key": null,
  "ref": null,
  "props": {id: "main", children: "Welcome to my Page"},
  "_owner": null,
  "_store": {}
}
```

208

The React Element Object - detailed

- The *type* property defines the type of element to create
- *key* may be needed to provide a unique identifier across a set of elements
- *props* defines any classes or id's as well as child elements
- *ref* gives direct access to an element in order to modify its properties. In general, such direct access goes against the principles of React programming and is normally avoided. It is set using React.createRef()
- There are methods to build and access the React element, so you will not need to create the object in that manner.

209

createELement

- createElement() sets up a React object
- Assume you want to create the following: <h1 id="main">Welcome React World!</h1>

```
React.createElement(
   "h1",
   {id: "main"},
   "Hello React World!"
   )
```

Without an id:

```
React.createElement(
   "h1",
   null,     // nothing here
   " Hello React World!",
   )
```

```
{id: "main", className: "title"}
This section can include several
attribute/value pairs separated by
commas.

Note:  Use className for class
```

210

Under the Hood

- React.createElement(
  "h1",
  {id: "main"},
  " Hello React World!“
)

```
$$typeof: Symbol(react.element)
key: null
props: {id: 'main', children: 'Hello
React World!'}
ref: null
type: "h1"
_owner: null
_store: {validated: false}
_self: null
_source: null
[[Prototype]]: Object
```

211

Render to the Page: ReactDOM.render

- ReactDOM.render(my_react, root_element);
- **You *must* render within a root element**
  for example,
- That means your page must have at least one element defined.
  (it is bad practice to render to the <body>)
- Create and render: Given a root element with id, “myapp”

  var hello = React.createElement("h1", {id: "main"}, “Hello React World!”)
  ReactDOM.render(hello, myapp);

212

Try it: Create a React Page

Place in the <head>

```
<script crossorigin src="https://unpkg.com/react@17/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
```

Place in the <body>

```
<div id="myapp"></div>  <!– place to put the React element(s) -->
   <script>
      var hello = React.createElement("h1",{id:"main"},"Hello React World!");
      ReactDOM.render(hello, myapp);
   </script>
```

213

Add styling

In the head section, <style> elements can be added:

```
<head>

<style type = "text/css">
   h1 {font-size: 28px;}
</style>

</head>
```

214

Eammple: Add attributes

- Using createElement to create a link:

```
var link = React.createElement("a",
            {className:"link-button", href: "https://extension.harvard.edu",
             target:"_blank"},
             "Learn @ Extension");
```

215

Try it

- Render the following to the page using React:

  Four score and seven years ago

- Add a css style for the "quote" class to set a width of 400px and have it centered on the page.

216

Try It Part 1

- Use React to create a link to access the Weather (weather.com) – it should open in a new window/tab
- View and test your page

217

Adding Child Elements

- createElement can be nested to include child elements
- For example, an HTML select element:

```
var select = React.createElement("select",{name:"pick"},
              React.createElement("option",null,"Basic"),
              React.createElement("option",null,"Standard"),
              React.createElement("option",null,"Premium")
          );
```

218

Multiple Levels of HTML Elements

- React can render nested HTML elements.
- For example:

```
<div id='nav'>
        <a href="./about">About</a>
        <a href="./services">Services</a>
        <a href="./contact">Contact</a>
</div>
```

219

Try It Part 2

- Create a list (<ol>) with items: Finance, Weather, Sports (or any others you want!)
- View and test your page

220

Using React to implement a nested element

- Implement one link within the <p>

```
var nav = React.createElement("p",
        {id:"nav"},
                React.createElement("a",
                {href: "./about"},
                 "About Us")
          ) //end “p” element
```

221

Example: Using React to implement navigation

```
var nav = React.createElement("p",
        {id:"nav"},
                React.createElement("a",
                        {href: "./about"},
                         "About Us"),
                React.createElement("a",
                        {href: "./service"},
                        “Services"),
                React.createElement("a",
                        {href: "./contact"},
                        “Contact")
          );  //end “p” element
```

222

Try It Part 3

- Update part 2 such that each item (ex, Finance) is rendered as wrapped in an anchor tag (<a>) to link to the corresponding site (ex, Weather links to https://weather.com).
- The links should open in a new window/tab
- View and test

223

DGMDE-28 so far …

224

URL's and HTTP Requests

- Client/Browser vs Server
- HTTP Request – how it works
- The role of the client vs the server
- How an SPA is different from a website
- Advantages of an SPA
- Disadvantages of an SPA

225

HTML Refresher

- Tags
  - Container tags
- Attributes and Values
- Entities
- Commonly used tags:
  - `H1 – H6`
  - `UL, OL, LI`
  - `FORM, INPUT, SELECT`
  - `A`
  - `DIV, SPAN`
  - `IMG`

226

CSS Refresher

- Selector
- Property / Value pairs
- Selector variations

.a	#a
a b	a>b
a,b	a.b
a:hover	a:first-child

- Where styles live: external, internal, inline
- Hex colors vs rgb color vs rgba
- Box model (padding, margin, box-sizing:border-box)
- Display, position, z-index

227

JavaScript

- JavaScript vs ECMAScript vs ES6
- <script> tag
- var vs const vs let
- Scope
- null
- Loosely typed, parseInt, parseFloat
- Strings
- Operators: arithmetic, comparison, logical, assignment, conditional
- Precedence, associativity, unary vs binary vs ternary operators

228

Loops and Conditionals

- if, else if, else
- for
- for in (objects)
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in
- for of (arrays)
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in
- while, do while
- break, continue

229

Functions

- Basic syntax
- Arrow functions
- Function definition as a value

```
f = (x,y)=> {parseInt(x)>y}
f()
foo(f)
```

  - Functions as expressions
  - Assigning a function to a variable
  - Functions as return values
  - Functions as object methods

230

Event-driven programming

- onclick
- onchange
- When & where to associate an event with an event handler
- Displaying results from an event handler

231

Arrays

- The Array object
  - Some useful methods
    - sort
    - join
    - map
    - forEach
    - reduce
    - Filter
  - Immutable vs Mutable methods
- Destructuring an array
  - var [a,b,c] = [1,2,3,4];
- The spread operator
  - var numbers = [1,2,3,4];
  - x = [...numbers, 5];
- Associative arrays

232

Asynchronous Operations and API's

- Callback function
- XMLHttpRequest
  - Status codes
- CORS
- API's
- Postman.com
- Promises and fetch
- await / async

233

Using Objects - overview

- Constructor
- Methods, properties, members
- Instance an object with new
- this
- Dot notation to access a member of an object
- extends
- Private members (#)

234

Associative Arrays & Objects

- Associative array
- Simple object
- function object
- class object
- JSON object
  - JSON.parse()
  - JSON.stringify()

235

JSON vs Object vs Associative Array

- JSON

```
var e = {
"name": "Bill",
"yearsOfService": 6,
"certifications": ["abc","dra","fmr"]
}
```

- Simple object

```
var e = {
name: "Bill",
yearsOfService: 6,
certifications: ["abc","dra","fmr"]
}
```

- Associative Array

```
e["name"] = "Bill";
e["yearsOfService"] = 6;
e["certifications"]= ["abc","dra","fmr"]
```

236

Function Object vs Class Object

```
function Employee(name,years) {
        this.name = name;
        this.yearsOfService = years;
        this.certifications= [];
        this.addCert = (certName) =>
        {
          this.certifications.push(certName)
        }
}
```

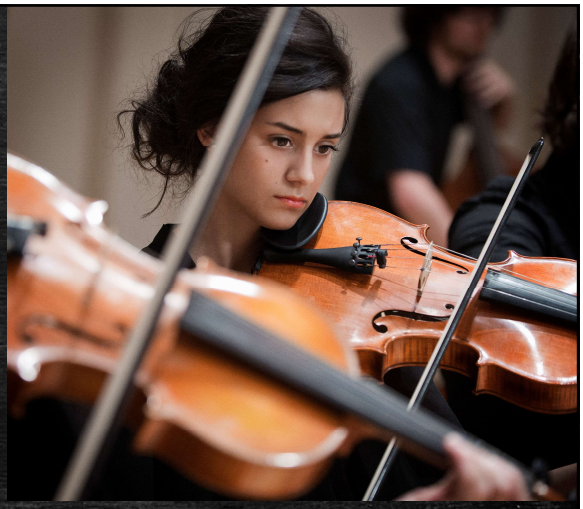
```
class Employee {
        constructor(name,years)
        {
                this.name = name;
                this.yearsOfService = years;
                this.certifications= [];
        }
        addCert(certName)
        {
           this.certifications.push(certName);
        }
}
```

```
e = new Employee("Bill",6);
e.addCert("abc");
document.write(e.name+ " " + e.yearsOfService + " " + e.certifications);
```

237

OOP – The Process

1. Identify all possible object types
2. Identify data elements in each object
3. Identify actions for each object
   - Setter
   - Getter
   - Page update
4. Identify nesting – i.e., which objects manage/control other objects



Each object should handle only its own tasks

238

Example: Card Game

- Objects
  - Card
  - Deck
  - Game
- Hierarchy
  - Game uses the deck
  - Deck has cards
- Class members: Card
  - suit
  - rank
  - value
  - getValue()
  - display()
- Class members: Deck
  - cardsLeft
  - shuffle()
  - deal()
- Class members: Game
  - players[]
  - startGame()
  - playGame()
  - checkWin()

239