

Important Links

Monday, November 15, 2021 9:48 AM

- **This Notebook Link:** https://1drv.ms/u/s!AknT1SrRuCz-uKpeoi0c2rqk_rtx1Q
- **Github Link:** <https://github.com/vivekduttamishra/202111-walmart-react>
- **Participant's Information Form:**

Javascript Prerequisites

Monday, November 15, 2021 10:38 AM

1. Basic variables —> var, let, const
2. Array and Array Functions
 - a. Basic syntax
 - b. Array Functions
 - i. append
 - ii. map
 - iii. filter
 - iv. forEach
3. Functions
 - a. Basic functions Syntax
 - b. Function as variable syntax
 - c. Arrow functions
 - d. arguments object
 - e. Params types
4. Object Notion
 - a. JavaScript Objects (NO Class syntax)
 - b. Constructor model
 - c. Json model
 - d. Dictionary model
 - e. Class
 - f. Inheritance
 - g. Bound Methods
5. Scopes and Closures
6. Object/Array De-structuring
7. Params
8. **Async programming using Promises**
 - a. **Async-await keyword**
9. Import export

Javascript Functions

Monday, November 15, 2021 11:25 AM

Argument and Parameter relationship

```
20
21 function act( action, param){
22
23     action(param);
24
25 };
26
27 act(person.eat, "Lunch");
28 act(person.move, "Home", "Office");
```

Not assigned
No error

- Unlike most other languages, in javascript it is allowed to pass different number of arguments than formal parameters required by the function
- You may pass more arguments than function formal parameter list suggests
 - The additional arguments are ignored for formal parameter list
 - They are still retained as part of 'arguments' object
- If you call method with lesser arguments, javascript assumes that you have passed **undefined** for the remaining arguments.

- All these elements are assigned to an arguments object present inside the function

Params and Spread Syntax

```
15
16 person.move=function(from,to){
17     console.log(`Person ${this.name} moves from ${from} to ${to}`);
18 }
19 person.move=person.move.bind(person);
20
21 function act( action, ...param){ //params syntax
22
23     action(...param); //spread operator
24
25 };
26
27 act(person.eat, "Lunch");
28 act(person.move, "Home", "Office", "Club");
29
```

Ignored

- Does opposite of params
 - It breaks a list into individual values

person.move, ["Home", "Office", "Club"]

- Collects all remaining arguments as an array.

Bound Methods

Monday, November 15, 2021 10:45 AM

JS 02-object-method.js > ...

```
1 var person = {  
2   name: 'Vivek'  
3 };  
4  
5  
6 const show = function(){  
7   console.log('Person [${this.name}]');  
8 };  
9  
10 person.show=show;  
11  
12  
13 person.show();
```

Object Methods

- We can assign a function as a property of an object
- The invoking object is automatically injected as 'this'

Version #2

JS 03-object-method.js > ...

```
1 var person = {  
2   name: 'Vivek'  
3 };  
4  
5  
6 person.show= function(){  
7   console.log('Person [${this.name}]');  
8 };  
9  
10  
11 person.show(); //person is passed as 'this'  
12  
13  
14 const display= person.show; // 'show' is assigned to display.  
15  
16 display(); // doesn't know about person. Has no 'this'
```

- This function is still a normal function assigned to object

- This will work as long as we are calling it with object reference

- Display is just the show function without any relation to the object person.

- Display is called directly (without object .)
 - This refers to a global context
 - In browser applications, this refers to window object

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\MyWorks\Corporate\202111-walmart-react\basic-js-demos>node "d:\MyWorks\(  
Person [Vivek ]  
Person [undefined ]  
D:\MyWorks\Corporate\202111-walmart-react\basic-js-demos>
```

Passing an Object method as parameter

```

JS 04-object-method.js > ...
1  var person = {
2    name: 'Vivek'
3  };
4
5
6  person.show= function(){
7
8    console.log(`Person [${this.name} ]`);
9  };
10
11 person.eat=function(food){
12   console.log(`Person ${this.name} eats ${food}`)
13 }
14
15 person.move=function(from,to){
16   console.log(`Person ${this.name} moves from ${from} to ${to}`);
17 }
18
19
20 function act( action, param){
21
22   action(param);
23
24 };
25
26
27 act(person.eat, "Lunch");
28

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

Person [undefined ]
D:\Myworks\Corporate\202111-walmart-react\basic-js-demos>node "d:\Myworks\Corporate\
Person undefined eats Lunch

```

- Same as
 - `const action=person.eat`
 - You lose the person object and pass only action
 - When action is called, person has not invoked it
 - this doesn't refer person.

This binding

- We can bind a method with a particular Object such that it always treats that object as this
 - Even if called without explicit **object**. Syntax

```

JS 04-object-method.js > ...
1  var person = {
2    name: 'Vivek'
3  };
4
5  person.show= function(){
6
7    console.log(`Person [${this.name} ]`);
8  };
9
10 person.eat=function(food){
11   console.log(`Person ${this.name} eats ${food}`)
12 }
13
14 person.eat= person.eat.bind(person);
15
16 person.move=function(from,to){
17   console.log(`Person ${this.name} moves from ${from} to ${to}`);
18 }
19
20
21 function act( action, param){
22
23   action(param);
24
25 };
26
27 act(person.eat, "Lunch");
28 act(person.move, "Home", "Office");

```

person.eat= person.eat.bind(person);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

D:\Myworks\Corporate\202111-walmart-react\basic-js-demos>node "d:\Myworks\Corporate\

```

- bind is a builtin method for all Javascript Objects
- It binds the 'this' context of current method with the given object permanently
- Now even if the method is used as a standalone method, method will continue to use same 'this' reference

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\Myworks\Corporate\202111-walmart-react\basic-js-demos>node "d:\Myworks\Corporate\  
Person Vivek eats Lunch  
Person undefined moves from Home to undefined
```

Ajax flow (code)

The screenshot shows the Google Account creation page. A red box labeled "1. On text change" points to the first name input field. A yellow box highlights the first and last name input fields. A yellow arrow points from the last name input field to a server icon in the top right corner. Another yellow arrow points from the server icon to the available usernames list. A third yellow arrow points from the available usernames list to the password input field. A fourth yellow arrow points from the password input field to the JavaScript code block.

Create your Google Account
to continue to Gmail

First name: Vivek
Last name: Dutta Mishra

Username: duttamishravivek5@gmail.com
You can use letters, numbers, & periods

Available: duttamishrav99, vivekduttamishra6, vduttamishra026

Password: Confirm:

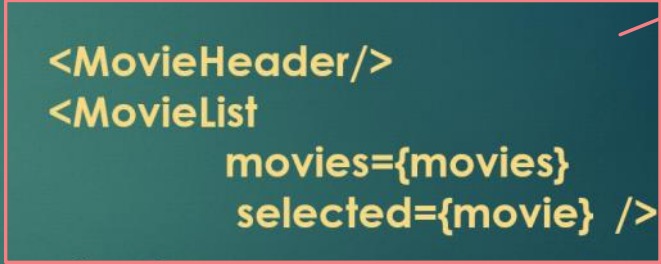
Sign in

```
async function onChange(){  
  const name= firstNameTextbox.value+" "+ lastNameTextbox.value;  
  
  const response=await fetch(requestUrl+"?name="+name);  
  
  userNameTextbox.value= response.body.options[0];  
  recommendationBox.innerHTML= createResponseText(response.body.options)
```

React is Component Driven Design

Monday, November 15, 2021 12:40 PM

```
<html>
...
<body>
  <MovieHeader/>
  <MovieList
    movies={movies}
    selected={movie} />
</body>
</html>
```



- Create custom HTML like Tags
- To represent our Business
- We can create
 - New Elements
 - New Attributes
 - New Events

React Links

Monday, November 15, 2021 10:00 AM

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

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

React Components are not DOM Elements

Monday, November 15, 2021 1:10 PM

[object Object]

- React Component is like an HTML component
- Current component translates to
 - `<h1>Welcome to React</h1>`
- But this is not a real HTML "h1" tag
 - It is a part of **Virtual DOM**
- It can't be displayed within your DOM directly and properly
 - It is treated as a normal Object

```
index.html<html><body>5  <meta http-equiv="X-UA-Compatible" content="IE=edge">6  <meta name="viewport" content="width=device-width, initial-scale=1.0">7  <title>Document</title>8  </head>9  <body>10   <h1>Hello React World</h1>11   <div id="placeholder">12     If you see this message, react is not working!13   </div>14   <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin="anonymous"></script>15   <!-- <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin="anonymous"></script>16   <script src="app.js"></script>17   </body>18 </html>
```

```
app.js1 console.log("hello world");22 const placeholder = document.getElementById("placeholder");34 //placeholder.innerHTML = "Hello World From App";56 const component=React.createElement("h1",null,"Welcome to React");78 placeholder.innerHTML=component;91011
```

React DOM library

- A library responsible for converting React Virtual Component to HTML DOM objects
- It updates the HTML DOM with React Virtual DOM

The image shows a web browser and two code editors. The browser displays the text "Hello React World" and "Welcome to React". The code editors show the HTML and JavaScript files. Blue arrows indicate the flow of data from the JavaScript code to the HTML elements and then to the browser display.

HTML File (index.html):

```
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <title>Document</title>
8 </head>
9 <body>
10   <h1>Hello React World</h1>
11
12   <div id="placeholder">
13     If you see this message, react is not working!
14   </div>
15
16   <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
17   <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>
18   <script src="app.js"></script>
19
20 </body>
21 </html>
```

JavaScript File (app.js):

```
1 console.log("hello world");
2
3 const placeholder = document.getElementById("placeholder");
4
5 //placeholder.innerHTML = "Hello World From App";
6
7 const component = React.createElement("h1", null, "Welcome to React");
8
9 //placeholder.innerHTML=component;
10
11 ReactDOM.render(component, placeholder);
12
```

Browser DOM Tree:

```
<body>
  <h1>Hello React World</h1>
  <div id="placeholder">
    <h1>Welcome to React</h1>
  </div>
  <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
  <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>
</body>
```

Nest React Elements with attributes

Monday, November 15, 2021 1:59 PM

HTML vs React Component

HTML

```
<div class="header">
  <h1>ReactWeb</h1>
  <p class="slogan">Welcome to React</p>
</div>
```

Note: Nested Child

React

```
const header= React
.createElement("div", { className:"header" },
  React.createElement("h1",null, "ReactWeb"),
  React.createElement("p",{
    className:"slogan"
  },
    "Welcome to React"
  ));
```

Assignment01

Monday, November 15, 2021 1:59 PM

Create the following UI using React Component

