JavaScript and the DOM

INFO 253A: Front End Web Architecture

Kay Ashaolu

General Purpose

- JavaScript full programming language
- Started in the browser
- Now used on servers, command line, devices...

Limited

- Manipulate the DOM
- Capture form values
- Make asynchronous web requests (AJAX)

Review: How to link a JavaScript File

```
<body>
        <h1>Test</h1>
        <script src= "../js/script.js" type="text/javascript" ></script>
</body>
```

DOM

- Document Object Model
- document is a JavaScript Object
- You can modify it and reflect the changes

Inspecting the DOM

- You can use childNodes to explore children
- Will return a list
- document.childNodes[0]

Traversing the DOM

html/index.html

```
<!DOCTYPE HTML>
<html>
        <head>
                <meta charset="utf-8" />
                <link rel="stylesheet" href="../css/style.css" />
        </head>
        <body>
                <div id="title" class="heading">
                        Info Web Arch Languages/Frameworks!
                </div>
                <ul>
                        <li class="heading">HTML/CSS/JavaScript</li>
                        <li>Python / Flask</li>
                        <li>Heroku</li>
                </ul>
                <!-- a comment node -->

                <script src="../js/script.js"></script>

        </body>
</html>
```

Traversing the DOM

css/style.css

```
.heading {
        color: red;
        background-color: yellow;
}
```

Traversing the DOM

js/script.js

```
function printBodyDOM() {
        let childNodes = document.body.childNodes;

        for(let i=0; i<childNodes.length; i++) {
                alert(childNodes[i]);
        }
}

printBodyDOM();
```

The DOM contains everything

- Note that even the whitespace in between elements are included
- Note that even comments are included

Selecting Nodes

- document.querySelectorAll(*[selector]*)
  - Input: string selctor, Output: a **list** of nodes
  - Can put **CSS like selectors** into this function
  - Will return a list of nodes that match the selector
  - Will give you access to change anything about those nodes
  - Note the difference between JavaScript and CSS: can do more than presentation, **can change the content!**

Selecting Nodes Example

```
function printHeadingClassElements() {
        let childNodes = document.querySelectorAll(".heading");

        for(let i=0; i<childNodes.length; i++) {
                alert(childNodes[i]);
        }
}

printHeadingClassElements();
```

Selecting Nodes Example

```
function printTitleIDElements() {
        let childNodes = document.querySelectorAll("#title");

        for(let i=0; i<childNodes.length; i++) {
                alert(childNodes[i]);
        }
}

printTitleIDElements();
```

Selecting Nodes Example

```
function printLIElements() {
        let childNodes = document.querySelectorAll("li");

        for(let i=0; i<childNodes.length; i++) {
                alert(childNodes[i]);
        }
}

printLIElements();
```

Other selectors

These run significantly faster than querySelectorAll, and should be used when possible

- document.querySelector()
  - Returns the first matched selector. Useful when you only want or are looking for the first node found
- document.getElementById()
- document.getElementsByTagName()
- document.getElementsByClassName()

Modifying DOM

- The property .innerHTML is the text inside the element
- You can modify the text inside an element with modifying the .innerHTML property

innerHTML example

```
1 function changeTitleText(titleText) {
2         let titleNode = document.querySelectorAll("#title")[0];
3         titleNode.innerHTML = titleText;
4 }
5
6 let titleText = prompt("Change title text");
7 changeTitleText(titleText);
```

Dynamically changing css

- You can use JavaScript to change the presentation of text
- A very common approach is to add/remove classes to elements
- That way based on user action an element can change its look and feel without having to add new styles dynamically to spreadsheet

Dynamic CSS example

```
function addClass(className) {
        let ulNode = document.querySelectorAll("ul")[0];
        ulNode.classList.add(className);
}

let className = prompt("Add Class");
addClass(className);
```

Questions?

Synchronous Processing

- Linear execution, waiting for each function to finish
- "End" of a program when all statements executed
- Similar to calling and being on hold

Synchronous Python

Note: to run, may need to run: "[sudo] pip install requests" to install requests package

```
1 import requests
2
3 link = "https://www.ischool.berkeley.edu/people/kay-ashaolu"
4 f = requests.get(link)
5 print(f.text)
```

Asynchronous Processing

- Respond to events independently
- Run functions in response to actions
- "Callbacks" instead of being "on hold"

Callbacks

- You want to make a request to your bank
- Dial their number... on hold (synchronous)
- Or have them call you back? (asynchronous)

Why Async?

- When do you want your JavaScript to "finish"?
- What should UI do while waiting?
- What should UI do while animating?

Event -> Function

On a user event, run this function

```
let titleNode = document.querySelector("#title");
titleNode.addEventListener("click", function(event) {
        titleNode.classList.add("highlight");
})
```

Grand Example

html/index.html (1/2)

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8" />
                <link rel="stylesheet" href="../css/style.css" />
        </head>
        <body>
                <form id="locationForm">
                        <label for="name">Name: </label>
                        <input type="text" name="name" id="name" /><br />

                        <label for="state">State</label>
                        <select name="state">
                                <option value="CA">CA</option>
                                <option value="NY">NY</option>
                                <option value="TX">TX</option>
                        </select><br />

                        <input type="submit" value="Save Location" />

                </form>
                <br />
```

Grand Example

html/index.html (2/2)

```
<div id="locationList">
                </div>

                <script src="../js/script.js"></script>

        </body>
</html>
```

Grand Example

css/style.css (1/1)

```
.highlight {
        color: red;
        background-color: yellow;
}
```

Grand Example

js/script.js (1/1)

```
1  /* Populate with current location */
2  let locationDiv = document.getElementById("locationList");
3
4  if (localStorage.curLocation != null) {
5          locationDiv.innerHTML = localStorage.curLocation;
6  }
7
8  /* Run code on submit button push */
9  let locationForm = document.getElementById("locationForm");
10 locationForm.addEventListener("submit", function(event) {
11         let name = locationForm.elements.namedItem("name").value;
12         let state = locationForm.elements.namedItem("state").value;
13
14         localStorage.curLocation = name + ": " + state;
15
16         let locationDiv = document.getElementById("locationList");
17         locationDiv.innerHTML = localStorage.curLocation;
18         locationDiv.classList.add("highlight");
19
20         /* This stops the usual function of "submit" which is to send data
21         to another server */
22         event.preventDefault();
23 })
```

What is localStorage?

- Browsers now have its own database that you can use
- Values persistent until user clears their own browsing history
- sessionStorage: lasts throughout the life of the browser tab

Async/Await

- In the newer versions of JavaScript there has been a effort to make these Async functions easier to reason against
- Sometimes always having to write a callback function for all response modes can be onerous
- This is where Async/Await comes in

Async/Await Example

```
let getSummaryData = async () => {
    let response = await fetch('https://raw.githubusercontent.com/UCB-INFO-F

    if (!response.ok) {
            throw new Error(`HTTP error! status: ${response.status}`);
    } else {
      return response.json();
    }
}

getSummaryData().then((response) => {
        /* This is where your code should be */
        console.log(response);
        /* End section where your code should be */
});
```

Questions?