

# CMSC335

---

## Web Application Development with JavaScript



## Asynchronous Processing

Department of Computer Science

University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

# Extending Built-in classes

---

- **Example:** ExtendedArray.html

# Function Context

---

- **Problem**
  - **Example:** FunctionContextIncorrect.html to see state of **ButtonState**
- Approaches to address the problem?
  - Renaming 'this' in function (closure)
  - Use arrow functions
    - » Arrow functions do not have their own this reference; they remember 'this' at the point they are defined
  - Use bind 'this' to the desired object
- **Example:** FunctionContextCorrectA.html – using closure
- **Example:** FunctionContextCorrectB.html – using arrow function
- **Example:** FunctionContextCorrectC.html – using bind



# Callbacks

---

- Many actions in JavaScript are ***asynchronous*** – *i.e.*, they are initiated but finish later
  - e.g., setTimeout, loading scripts and modules, ...
- A function that does something asynchronously should provide a callback argument where we put the function to run after it's complete
- **Lodash** – JavaScript utility library
  - <https://lodash.com/>
- **Example:** CallbackIncorrect.html
  - Trying to use Lodash to compute intersection of arrays
- **Example:** CallbackCorrect.html
  - Trying to use Lodash to compute intersection of arrays

# Callbacks and (Deeply) Nested Callbacks

---

- Code with nested callbacks is considered difficult to read
  - Called “callback hell” or “pyramid of doom”
- For example,
  - Get a user from user lists, errorCallback
    - Get posts by the user , errorCallback
      - Get the users liked a post , errorCallback
        - Display # of likes each user clicked, errorCallback
- **Example:** nestedCallbacks folder
  - To run – node .\asynchNested.js

# Promises

---

- **Promise** - object that represents the eventual completion (or failure) of an asynchronous operation
  - We attach callbacks to the promise object
  - Allows promise chaining
    - » Execution of two or more asynchronous operations back to back where results of one step are used by the next
- First, we will explore how to use promises by using the Fetch API
- Later one we will see how we can define our own promises
- Reference
  - [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using\\_promises](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises)

# Fetch API

---

- Provides an interface for fetching resources (including across the network)
  - Takes one argument: the path to the resource
  - **Returns a Promise** that resolves to the Response to that request, whether it is successful or not
  - HTTP error status (e.g., HTTP 404) will be resolved normally (with ok status set to false)
  - Only reject on network failure
- Example where we display json

```
    fetch(url)
      .then(response => response.json())
      .then(json => console.log(json));
```
- **Example:** PromisesFetch\*.html
  - The node.js example requires the node-fetch package and updating package.json with “type”: “module”

# Fetch API

---

- By default (by just providing URL) we are generating GET request
  - A second **options** parameter allows you to issue a POST request
- URL has to be an absolute URL
- Response object has information such as:
  - `Json()` – parses the body of the response into a JSON object and generates an error if the parsing fails
  - `text()` – returns the body of the response as text
  - `status` and `statusText` – Information about HTTP status code
  - `ok` – true if status is a 2xx status code
  - Headers
  - Reference:
    - <https://stackabuse.com/making-http-requests-in-node-js-with-node-fetch/>



# HTML5 Local Storage API

---

- **localStorage** → stores data with no expiration date
- To store data:
  - `localStorage.setItem("name", "Mary");`
- To retrieve data:
  - `localStorage.getItem("name");`
- You can only store strings. To store objects you could use `JSON.stringify`
- **sessionStorage** → equivalent to `localStorage`, but data is deleted when the browser is closed
- **Example:** `ToDoList.html`
  - Notice the use of the **contenteditable** property in the HTML
  - Try the property in other HTML elements
- To clean `localStorage`
  - Right click on page in Chrome
  - Select "Inspect"
  - Select "Console"
  - Type `"window.localStorage.clear()"`
- Reference: [http://www.w3schools.com/html/html5\\_webstorage.asp](http://www.w3schools.com/html/html5_webstorage.asp)

# HTML5 Canvas

---

- **Canvas**
  - Container for graphics
  - It is a rectangular area on the page
  - You use JavaScript to draw on the fly
  - Suited for game applications
  - You can store the image as a jpg or png file
- Reference and Examples:
  - [http://www.w3schools.com/html/html5\\_canvas.asp](http://www.w3schools.com/html/html5_canvas.asp)
- **Example:** DrawingPointer.html
- How would you create an animation of a drawing you have done?

# HTML5 SVG

---

- SVG – Scalable Vector Graphics
  - Language use to describe 2D graphics
  - Pure XML
  - SVG graphics do not lose any quality when resized or zoomed
  - Best suited for applications with large rendering areas
  - You can embed in your HTML
- <svg> element → container for SVG Graphics
- Reference, Examples and comparison table available at:
  - [http://www.w3schools.com/html/html5\\_svg.asp](http://www.w3schools.com/html/html5_svg.asp)
- Open-source vector graphics editor
  - <https://inkscape.org/en/>

# HTML5 Geolocation API

---

- **Example:** Geolocation.html
- You can try the following link on your phone  
<https://www.cs.umd.edu/~nelson/geoLocationExamples/>
- Reference:
  - [http://www.w3schools.com/html/html5\\_geolocation.asp](http://www.w3schools.com/html/html5_geolocation.asp)

# Nonextensible and Sealed Obj

---

- In JavaScript you can add properties and methods to an object any time (extensible)
- You can restrict this behavior by using `Object.preventExtensions()`
- What if you don't want properties deleted as well?
  - Seal the object
  - By sealing an object you create the same abstraction in class definition where once a class is defined the class values and methods are set
- **Example:** `ExtensibleSealed.html`

# Freeze

---

- Strictest protection
- Not extensible, sealed, and data properties can not be modified
  - Constant object
- **Example:** Freeze.html