ITMD 441/541
Web Application Foundations
Week 9

FALL 2023 – OCTOBER 16, 2023

Weekly's Agenda

- Revisit Lab 3
- Global Variables and Namespace
- AJAX, API, Demo
- Intro to Web APIs

Global Variables and Namespace

Global Variables and Namespace Problem

- Variable names need to be unique
- All variables by default are placed in the global "window" namespace when defined normally with the var keyword unless defined inside a function. This is Functional Scope. It changes some with ES6+ let and const and there is Block Scope for those.
- Variables defined inside a function are scoped or name spaced to that function.
- If we mix our code with others code, we may overlap variable names
- We want to define our own namespace
- We have a few ways we can do that.

Immediately-invoked Function Expressions (IIFE)

- Sometimes called a self executing protecting function
- Idea is we wrap our code in an anonymous function that is immediately executed.
- All variables and functions will be locally scoped to that function.
- Takes advantage of JavaScript closures

```
(function(){

//anything in here has its own special namespace
//functions and variables are not available to the outside

})();
```

The Big Plain Map/Object Literal

- Create a global variable and store an object in it
- Add all your variables and functions as properties on that object
- Useful for quick solution but doesn't have private variables or any methods

```
var myNamespace = {};

myNamespace.aFunction = function(){
};

myNamespace.aVariable = 7;
```

Exporting a Map to the global Namespace

```
(function(){
    var myNamespace = {};

    //Private variables
    var name = “brian”;

    //Public members/variables
    myNamespace.myName = “brian”;

    //Public Method
    myNamespace.myFunction = function(){ alert(this.myName)};

    window.myNamespace = myNamespace;
})();
```

Exporting a Map to the global Namespace with possible namespace in window already

```
(function(ns){
    //Private variables
    var name = “brian”;

    //Public members/variables
    ns.myName = “brian”;

    //Public Method
    ns.myFunction = function(){ alert(this.myName)};

    window.myNamespace = ns;

})(window.myNamespace || {});
```

Wrapping with an event function

- Another option would be using a window event handler function listening for an event like `DOMContentLoaded` to act as your namespace protector.
- Since the event handler is a function, it has function scope and keeps the variables and functions local to that function.

Reading/Assignments

- Midterm Exam is due in end of day Wednesday October 18 (covers weeks 1-7)
- Lab 3 Due, Sunday October 29
- Quiz will come this weekend to cover Week 8 and Week 9 lecture
- Lab 4 will come soon
- Midterm grades are due for Illinois TECH on October 20.

Agenda for October 19

- Revisit Week 8 and do additional DEMOs for AJAX
- Complete audio and video demonstrations from Week 9
  - (only variables, audio and video will be covered on Quiz)
- Quiz will contain 10 questions (multiple choice and T/F). Not included in midterm grade.

Agenda for October 24

- DEMOs for AJAX
- Geolocation
- Web workers
- Storage API
- Indexed DB
- Will start Canvas next class.

API, AJAX Demo

API, AJAX Demo

- We are going to do a short demo application which basically replicates the breed list functionality on the Dog API site
- https://dog.ceo/dog-api/
- Open API that returns images of various dog breeds

Intro to Web APIs

HTML5/JS APIS

HTML5 Video & Audio

- Built in support for playing audio and video in the browser without plugins like flash or silverlight
- Uses the `<video>` or `<audio>` tag for basic support.
- Traditionally format/codec support was mixed between browser manufacturers, and we needed to supply different formats
- As of 2016 h.264 mp4 and mp3 is supported in most browsers
- In Oct 2013 Cisco announced they would make a h.264 module available to all
- You can provide fallback content inside the tag if the browser doesn't support it
- There are attributes for controls, autoplay, loop, preload. See element docs
- https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML5_audio_and_video
- https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats
- Use examples with VSCode

HTML5 Video & Audio

- The `<source>` tag can be nested inside the `<video>` or `<audio>` tag to supply multiple formats

```
<video src="http://v2v.cc/~j/theora_testsuite/320x240.ogg" controls>
     <p>Your browser does not support the <code>video</code> element.</p>
</video>
```

```
<video controls>
     <source src="SampleVideo.ogv" type="video/ogv">
     <source src="SampleVideo.mp4" type="video/mp4">
     <p>Your browser does not support the <code>video</code> element.</p>
</video>
```

- The `<video>` and `<audio>` elements have methods attached for controlling playback
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content

Geolocation

- The geolocation API allows the user to provide their location to web applications if they so desire. For privacy reasons, the user is asked for permission to report location information.
- The API is published through the `navigator.geolocation` object
- `getCurrentPosition()` method is used to query the browser for a location object
- Takes a callback function that runs when the browser responds with a location and also takes an optional second callback function to run if there is an error
- `watchPosition()` method can be used to continually update the position

```
navigator.geolocation.getCurrentPosition(function(position) {
     do_something(position.coords.latitude, position.coords.longitude);
});
```

- https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API/Using_the_Geolocation_API
- DEMO using VSCODE

Web Workers

- Web Workers provide a simple way for web applications to run scripts in background threads. Basically, adds a multithreading model to JavaScript
- The worker threads do not block or interfere with the user interface or main thread
- The main script and worker scripts post messages to each other and can pass standard JavaScript data between each other.
- This is done with the `postMessage()` method and the `onmessage` event
- https://github.com/michaeltreat/Web-Worker-Demo
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers
- http://www.html5rocks.com/en/tutorials/workers/basics/
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API
- https://en.wikipedia.org/wiki/Web_worker

Websockets

- The WebSocket Protocol is an independent TCP-based protocol.
- WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application.
- WebSocket is a protocol providing full-duplex communication channels over a single TCP connection.
- Often used for real time data communications from browser to servers.
- https://en.wikipedia.org/wiki/WebSocket
- https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
- https://html.spec.whatwg.org/multipage/comms.html#network
- https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications
- https://javascript.info/websocket
- https://medium.com/@cn007b/super-simple-php-websocket-example-ea2cd5893575

Reminders

- Lab 3 Due, Sunday October 29
- Quiz covers Week 8 and ½ of Week 9 lecture. Only variables, audio and video from Week 9. **Due: Thursday, October 26**
- Lab 4 will come soon
- Midterm grades were submitted on October 20 by midnight.
- **Final exam will be posted on Sunday, December 3 and due by Thursday, December 7 by midnight. (Final Exam week Dec. 4-9th)**
- **Very similar to the midterm except it will cover all class material and assignments. No project. 60-75 questions.**

HTML5 Storage

Examples of Storage APIs

- Short Google video
  - http://www.html5rocks.com/en/features/storage
- Storage API example
  - https://www.w3schools.com/JS/tryit.asp?filename=tryjs_api_storage_example
- Web Storage
  - https://html.spec.whatwg.org/multipage/webstorage.html
    - Storage additional items. Get an item that does not exist in the storage
- Indexed Database
  - https://www.w3.org/TR/IndexedDB/
- Web SQL Database – This has been deprecated and will no longer be developed or supported in the future.
  - https://www.w3.org/TR/webdatabase/
- File Access
  - https://www.w3.org/TR/FileAPI/
  - https://developer.mozilla.org/en-US/docs/Web/API/File
  - https://developer.mozilla.org/en-US/docs/Web/API/File/Using_files_from_web_applications

Web Storage

- Simple and fairly widely supported way to store data in the client browser.
- Simple string only Key/Value storage – **IMPORTANT if you want to store a JavaScript object you must use JSON.stringify() and JSON.parse().**
- Different than cookies as this is a JavaScript API
- There two areas you can store data in
  - `localStorage` – persistant storage after browser is closed
  - `sessionStorage` – only stores for current browser session
- Data is saved per origin and there is a data limit
  - Firefox is around 10MB
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

Web Storage

- The storage object is exposed on the window object
- `window.localStorage` or just `localStorage`
- You can access key/value pairs in multiple ways

```
localStorage.colorSetting = '#a4509b';
localStorage['colorSetting'] = '#a4509b';
localStorage.setItem('colorSetting', '#a4509b');
localStorage.getItem('colorSetting’);
localStorage.removeItem('colorSetting’);
```

- There is also a storage event that fires that you can listen for and react to

Indexed DB

- IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs, which also enables high performance searches of this data using indexes.
- Transactional based database system, non-sql based, JavaScript-based object-oriented database.
- Store and retrieve objects based on a key.
- Asynchronous API that uses a lot of function callbacks
- Storage limits do exist but much larger than Web Storage and differ by browser
  - https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Browser_storage_limits_and_eviction_criteria
- https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

Indexed DB

- The basic pattern that Indexed DB encourages is the following:
  - Open a database.
  - Create an object store in the database.
  - Start a transaction and make a request to do some database operation, like adding or retrieving data.
  - Wait for the operation to complete by listening to the right kind of DOM event.
  - Do something with the results (which can be found on the request object).
- https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB
- http://code.tutsplus.com/tutorials/working-with-indexeddb--net-34673

File Access

- Allows you to interact with local files using the File API
- Can access the file attributes and its data in JavaScript using the FileReader interface.
- Does not allow you to access files on the user's computer without the user providing them to you with an input of type=file or using the drag and drop api.
- https://web.dev/read-files/
- https://developer.mozilla.org/en-US/docs/Web/API/File
- We used this link below earlier:
- https://developer.mozilla.org/en-US/docs/Web/API/File/Using_files_from_web_applications

HTML Canvas

Canvas 2D, Web GL 3D

- The HTML Canvas API gives the developer a way to draw graphics using JavaScript inside an HTML <canvas> element
- The Canvas API is primarily a 2D graphics API
  - Useful for animations, games, data visualization, video and photo processing, and more
  - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
- The WebGL API is used to draw 2D and 3D graphics using hardware-acceleration but still uses the HTML <canvas> element
  - API is similar to OpenGL ES 2.0
  - https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
  - Go down the page until you see EXAMPLES
- We will probably do another lecture on Canvas

Intersection Observer

Intersection Observer

- Provides an API to watch for changes where observed elements intersect with an ancestor element and it does it asynchronously to not slow down the UI.
- Typically, the ancestor element is the viewport.
- Gives a standardized solution instead of old unreliable and slow solutions to determine this intersection information.
  - Lazy load images as they are scrolled into the viewport
  - Create infinite scrolling sites where more content loads automatically as you scroll
  - Determine to only run tasks or things like animations if something is in view

Intersection Observer (extra notes)

- Popular use
  - Reporting of visibility of advertisements in order to calculate ad revenues.
- Basic set up example:

```
let options = { root: document.querySelector("#scrollArea"), rootMargin: "0px", threshold: 1.0, };

let observer = new IntersectionObserver(callback, options);
```

A threshold of 1.0 means that when 100% of the target is visible within the element specified by the root option, the callback is invoked.

- All areas considered by the Intersection Observer API are rectangles; elements which are irregularly shaped are considered as occupying the smallest rectangle which encloses all of the element's parts.

- https://developer.mozilla.org/en-US/docs/Web/API/Intersection_Observer_API

Assignments

Reading/Assignments

- Lab 3 Due, Sunday October 29
- Quiz will come this weekend to cover Week 9 lecture
- Lab 4 will come soon