Reminders

- Attendance is 15% of grade. Please scan the QR code.
- Quiz 6 Due Thursday, October 5
- Lab 2 Due Tuesday, October 3 by midnight NOT noon, that was a typo.
- T.A. Hours
  - Monday and Wednesday 4-5 pm or by appointment
- **Midterm Exam will be available from October 12-18:**
  - **You must complete it by Wednesday, October 18 midnight. No exceptions.**
  - It will cover weeks 1-7 and both Labs
  - It will contain 50-60 questions and worth 100-120 pts. It will be online.
  - There are not retakes or make ups
  - Midterm grades are due on October 20th

ITMD 441/541
Web Application Foundations

Week 7

FALL 2023 – OCTOBER 2, 2023

This Week's Agenda

- HTML Forms Overview
- DOM Selection and Modification
- Event Handling
- Assignments

Additional resource for **classes**
You are responsible for the content in this video:

https://youtu.be/gPi3nMF1u-Q?si=hbso2QRTJ-YBCiww

HTML Forms

HTML Forms

- Forms are used to collect and send data back to a server or resource that can process the data
- Forms **CAN NOT** be nested inside other forms
- Forms collect data with form controls
- Really two parts
  - **`form`** element and form controls in the HTML page that the user interacts with
  - Some kind of server-side resource that the form will transmit the data to
- You can use form controls outside a form but they will not transmit any data without using some type of JavaScript to read and submit the data from them
- If you send the data using JavaScript, you will want to stop the default form processing with the submit events preventDefault method
- Form controls provide us a way to get data from the user

HTML Forms

- Form element contains all form controls
  - **`<form></form>`**
- Has two primary attributes you need
  - action & method
  - **`<form action="URLToScript" method="post"></form>`**
- Action attribute
  - The URI of a program that processes the form information.
- Method attribute
  - The HTTP method that the browser uses to submit the form.

Form Method

- The HTTP method that the browser uses to submit the form. Possible values are:
- **POST**
  - Corresponds to the HTTP POST method
  - Form data is included in the body of the HTTP Request and sent to the server.
  - Most often used
  - No character or size limit to data
- **GET**
  - Default if no method is specified
  - Corresponds to the HTTP GET method
  - Form data is appended to the action attribute URI with a '?' as separator, and the resulting URI is sent to the server.
  - http://server.com/script.php?name=Bob&city=chicago
  - Only use this method if the form has no side-effects and contains only ASCII characters.
  - URL length is limited and varies by browser & server
- **Neither method does any encryption of the data. That is the responsibility of TLS (SSL) during the connection.**
- https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form

Form Controls

- Form controls are used to collect input data from the user and submit them to some resource that processes that data.
- In basic HTML forms, the form controls need to be nested inside the <form></form> tags.
- Form controls can be used outside a <form> tag in an HTML page but there is no way for it to submit data without using JavaScript to read and submit the data.
- All form controls use a **name attribute**, or the data will not be submitted when using the form tag
- Form controls use a type attribute to set the type of control (there are a couple exceptions).
- You should always use label elements for accessibility
- https://developer.mozilla.org/en-US/docs/Learn/Forms/Basic_native_form_controls
- https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit
- STOP HERE AND COMPLETE A FULL FORM EXAMPLE FOR DIFFERENT DATA TYPES

Form Resources

- https://youtu.be/2O8pkybH6po?si=iFG0icnyvmLxrC2E
- https://developer.mozilla.org/en-US/docs/Learn/Forms/How_to_structure_a_web_form
- https://developer.mozilla.org/en-US/docs/Learn/Forms/Basic_native_form_controls
- https://developer.mozilla.org/en-US/docs/Learn/Forms/HTML5_input_types
- https://developer.mozilla.org/en-US/docs/Learn/Forms/Other_form_controls
- https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending_forms_through_JavaScript

Reminders

- Attendance is 15% of grade. Please scan the QR code.
- Quiz 6 Due Thursday, October 5
- Lab 2 was due Tuesday, October 3. No late assignments after this class.
- **Midterm Exam will be available from October 12-18:**
  - **You must complete it by Wednesday, October 18 midnight. No exceptions.**
  - It will cover weeks 1-7 and both Labs
  - It will contain 50-60 questions and worth 100-120 pts. It will be online.
  - **You have approximately 2 hours to complete. Maybe more or less depending on the test.**
  - There are no retakes or make ups
  - Midterm grades are due on October 20th
- We will go over the Lab 2 assignment briefly before covering DOM

Introduction to the DOM

- WILL COMPLETE ABOUT ½ OF THE SLIDES.
- DOM DEMO NEXT WEEK.
- ONLY WHAT IS COVERED IN CLASS WILL BE ON THE NEXT QUIZ.

The DOM

- Document Object Model (DOM)
- Object representation of a HTML or XML Document in memory
- All elements are represented by objects
- DOM is an API that can be used in many languages
- **JavaScript uses DOM scripting to modify the elements on a page**
- DOM is a collection of nodes in a tree
- Also provides standard methods to traverse the DOM, access elements and modify elements
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

JavaScript DOM

- Accessing the DOM elements
- Use methods of the document object
- Most common by id
  - `let a = document.getElementById("elementid");`
- Can also access by class, tag, selector
- Use the `Element.getAttribute("src");` method to get an attribute's value from an element. Use the `setAttribute` to set or change one.
- Many other methods and properties to access, transverse, and manipulate DOM objects.
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- **Click on "Introduction to the DOM" Examples are near the bottom**

DOM Selection Basics

- **By Id** – always will return one DOM element since id needs to be unique on the page.
  `let element = document.getElementById('someId');`
  https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById
- **By Class Name** – returns an HTMLCollection
  `let elements = document.getElementsByClassName('classnames');`
  https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByClassName
- **By Tag Name** – returns an HTMLCollection
  `let elements = document.getElementsByTagName('tagname');`
  https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementsByTagName
- **By Selector** – returns one element or a NodeList of element objects, uses CSS style selectors
  `let element = document.querySelector('css selector string');`
  `let elementList = document.querySelectorAll('css selector string');`
  https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector
  https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelectorAll

DOM Event Handling

JavaScript Event Handling

- Three Methods – 3rd method is the most preferred way
  1. As attribute on HTML element
  2. As a method attached to a DOM object
  3. Using the `addEventListener` method of an object

JavaScript Event Handling Method 1

- As attribute on HTML element
- Not suggested, mixes JavaScript and HTML structure in the HTML markup.
- Uses the attribute that pertains to the particular event. Usually in the form of on + something. Click event is `onclick` for example.
- JavaScript code is embedded in the event attribute's value.

```
<div onclick="alert('I was clicked')">click me</div>
```

JavaScript Event Handling Method 2

- As a method attached to a DOM object event property
- Not suggested, while it separates the event handler logic from the HTML markup you still have limitations.
- Can only apply one event handler to an element using this method.
- JavaScript function is assigned to the event name property on the element. Click is onclick for example.
- https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Event_handlers

```
<div id="aButton">Click Me</div>

document.getElementById('aButton').onclick =  function(){
    //code in this block
};
```

JavaScript Event Handling-Method 3

- Using the add event listener method of an object
- **This is the preferred modern standards compliant method. This is what I want to see unless you have good reason to use the other two and can explain why.**
- This allows you to bind multiple handlers on the same element to listen for the same event.
- Event types are without the word "on" before them. Click is click. https://developer.mozilla.org/en-US/docs/Web/Events
- `element.addEventListener("click", myFunction, false);`
- Make sure you pass the function object, not execute the function. Notice no parenthesis. Or declare an anonymous inline function.
- https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
- **EXAMPLE NEAR THE BOTTOM**
- **STOP HERE-WEEK 7. JUMP TO ASSIGNMENTS AT THE END.**

Capture/Bubble Event Phases

- Event handlers can be registered in either the bubbling (default) or capturing phases of the event cycle.
- This determines how nested elements with the same event handlers will process them.
- The capture phase starts at the highest element in the chain down through the ancestors to the target element
- The bubble phase starts at the target element and then goes up through the chain
- See demo and links
- https://javascript.info/bubbling-and-capturing
- https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events
- https://blog.logrocket.com/deep-dive-into-event-bubbling-and-capturing/

Waiting for the DOM

Waiting for the DOM to be ready

- How do you prevent your JS from running before the page is ready to be manipulated?
- You could add and an event handler to the widow's load event
  - This will fire the event handler at the end of the document loading process. All objects in the document are in DOM and all images, scripts, and other assets are also loaded.
  - Sometimes this will take longer than you really need. Most of the time you really just want the DOM to be ready to be manipulated.
  - https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event
- Better option is often using the `DOMContentLoaded` event on the `window` or `document` object.
  - This event fires when the HTML document has been fully loaded and parsed. It does not wait for other assets like images and stylesheets.
  - This event is often what you want vs listening for the full load event.
  - https://developer.mozilla.org/en-US/docs/Web/API/Window/DOMContentLoaded_event
  - https://developer.mozilla.org/en-US/docs/Web/API/Document/DOMContentLoaded_event
- https://gist.github.com/jsonberry/0d71007ea188785e1a3d13d2e30d58a5

Load vs DOMContentLoaded

- **Load event**

```
window.addEventListener('load', function (evt) {
  console.log('page is fully loaded');
});
```

- **DOMContentLoaded event**

```
document.addEventListener('DOMContentLoaded', function (evt) {
  console.log('DOM fully loaded and parsed');
});
```

See demo

DOM Manipulation Basics

DOM Element Manipulation

- DOM elements have a property, `innerHTML`, that sets or gets the HTML syntax that describes all the element's children.
- DOM elements also have an `innerText` property that can be used if the inner content is only text with no HTML syntax. HTML will not be parsed and inserted as plain text.

```
let content = element.innerHTML;
element.innerHTML = ‘<p>New HTML</p>’;
```

```
let content = element.innerText;
element.innerText = ‘this is some text’;
```

- `innerHTML` can introduce security concerns so you need to use it carefully.
- There is also a `textContent` property that is very similar to `innerText`. It differs in that it `textContent` will also get hidden text text content like a `<script>` tag not just visible text.
- https://developer.mozilla.org/en-US/docs/Web/API/Node/textContent#differences_from_innertext
- https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML
- https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/innerText

DOM Element Manipulation

- Getting values from form fields.
- Most basic is the **input element**. This could be of type text, button, checkbox, radio, and others that use the HTML `<input>` element form control.
  https://developer.mozilla.org/en-US/docs/Web/API/HTMLInputElement
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#Form_<input>_types
- Many properties to interact with the control. Very important one that applies to all types of input controls is value. Gets or Sets the value

```
element.value = '5';
let result = element.value;
```

- **All values are get or set as a string from a form control. You must do any type conversions.**

DOM Element Manipulation

- Other popular form controls include
- **Select lists**
  https://developer.mozilla.org/en-US/docs/Web/API/HTMLSelectElement
  See properties **`selectedIndex`**, **`selectedOptions`**, **`value`**, and others
- **Text Areas**
  https://developer.mozilla.org/en-US/docs/Web/API/HTMLTextAreaElement
  `value` property to get/set value same as input
- Helper methods you should looks into.
- **`Number()`** - Object
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number
- Other built in global functions and objects
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects
- Be very careful with the parseInt() function’s behavior as demoed in class and the NaN type.

DOM Element Manipulation

- Elements are the basic object everything descends from. Review this link for basic properties and methods that apply to all elements.
- https://developer.mozilla.org/en-US/docs/Web/API/Element
  https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement
- To properly create, move, delete, or modify elements you need to use properties or methods on the elements.
- **Changing Inline CSS Styles**
  ```
  document.getElementById('anId').style.backgroundColor = '#000000';
  ```
  https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/style
- CSS properties on style object usually use a camel case version of the name. Look it up if you can not figure it out.

DOM Element Manipulation

- Methods used to create and insert an element
- `document.createElement()`
  https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement
- `document.createTextNode()`
  https://developer.mozilla.org/en-US/docs/Web/API/Document/createTextNode
- `node.appendChild()`
  https://developer.mozilla.org/en-US/docs/Web/API/Node/appendChild
- `node.insertBefore()`
  https://developer.mozilla.org/en-US/docs/Web/API/Node/insertBefore
- `node.removeChild()`
  https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild
- `node.replaceChild()`
  https://developer.mozilla.org/en-US/docs/Web/API/Node/replaceChild

DOM Element Manipulation

- For example, to insert a new h3 in the body of an HTML page

```
let head3 = document.createElement(‘h3’);
let headtext = document.createTextNode(‘This is my H3 headline’);
head3.appendChild(headtext);
let b = document.getElementsByTagName(‘body’);
b[0].appendChild(head3);
```

DOM Resources

- https://developer.mozilla.org/en/docs/JavaScript
- https://developer.mozilla.org/en-US/docs/Web/API/Node
- https://developer.mozilla.org/en-US/docs/Web/API/Element
- https://developer.mozilla.org/en-US/docs/Web/API/Document
- https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild
- https://developer.mozilla.org/en-US/docs/Web/API/Node/appendChild
- https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement
- https://developer.mozilla.org/en-US/docs/Web/API/Document/createTextNode
- https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById
- https://developer.mozilla.org/en-US/docs/Web/API/Window/alert
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseInt
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/NaN

DOM Element Creation and Deletion Demo

Assignments

Reading/Assignments

- Quiz 6 – Week 6 Content – Due end of day October 5
- Reading:
  - Eloquent JavaScript Chapter 14 DOM
  - MDN Introduction to the DOM Section
    - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
  - MDN Learn Forms Section
    - https://developer.mozilla.org/en-US/docs/Learn/Forms
- Midterm Exam will be available by October 12 (covers weeks 1-7)
- Lab 3 – Coming soon