OIM3690 - Web Technologies

Introduction to JavaScript

What is JavaScript?

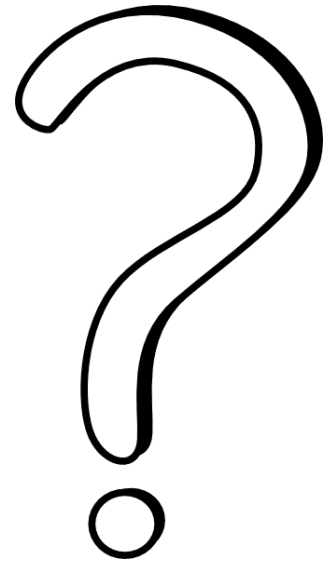
- A programming language created in 1995
  - Originally used only in web browsers (with JavaScript engine)
  - Now can be implemented in servers, usually via Node.js
- **Client-side JavaScript**
  - The main focus of this course
  - Runs on user's browser
  - Enables interactive web pages by creating dynamically updating content, controlling multimedia, animating images, and more
- Server-side JavaScript
  - Runs on the server-side environment
  - Used for building APIs, handling HTTP requests, accessing databases, etc.

How to add JavaScript

- **Internal** JavaScript
  - Can be added in `<head>` or bottom of `body` (preferably, to improve page loading performance)
  - We will be using this in class.
- **External** JavaScript
  - Create a separate `.js` file
  - Use `defer` / `async` to control when the script is loaded and executed
  - Syntax: `<script src="script.js" defer></script>`
- **Inline** JavaScript handlers
  - Connects events directly with elements.
  - It is considered **bad practice** because it mixes presentation and behavior, making the code harder to read, maintain, and test.
  - **Not Allowed in this class!**

JavaScript Examples

- Download *lec13-js-demo.html* from GitHub (*OIM3690/resources/templates*)
- Open the file in web browser and interact
- Read the source code
    - Any questions?



JavaScript - Basic Concepts

Document Object Model (DOM)

- What is **DOM**?
  - A programming interface for HTML and XML documents.
  - It represents the page so that programs can change the document structure, style, and content.
- See example
- **Data Types** in DOM
  - Document
  - Node
  - Element
  - NodeList

Data Types in DOM - Document

- The **Document** is the root of the **DOM** hierarchy.
- Every web page loaded in a browser has a **Document** object, which represents the entire document as a single entity.
- The **Document** object provides methods and properties for working with the document as a whole, such as `getElementById` and `getElementsByTagName`.

Data Types in DOM - Node

- A **Node** is a fundamental interface in the **DOM**, representing a single object in the document tree.
- This includes **elements**, **attributes**, and other types of nodes such as text nodes and comment nodes.
- Nodes can be accessed and manipulated using **methods** and **properties** of the Node interface, such as `parentNode`, `childNodes`, and `textContent`.

Data Types in DOM - Element

- An **Element** is a specific type of node that represents an HTML element.
- **Elements** have **properties** that allow access to their attributes, such as `getAttribute` and `setAttribute`
- **Elements** have **methods** for manipulating their content, such as `appendChild` and `removeChild`.

Data Types in DOM - NodeList

- A **NodeList** is an array-like object that represents a collection of nodes.
- It can be used to access and manipulate multiple nodes at once, such as using a **loop** to iterate over a list of elements and apply a common operation to each one.
- **NodeList** objects are typically returned by methods such as `getElementsByTagName` and `querySelectorAll`.

What is an *Event*?

When a page load happens, do play the video of a cat sliding into cardboard.

When a click happens, do submit my online purchase.

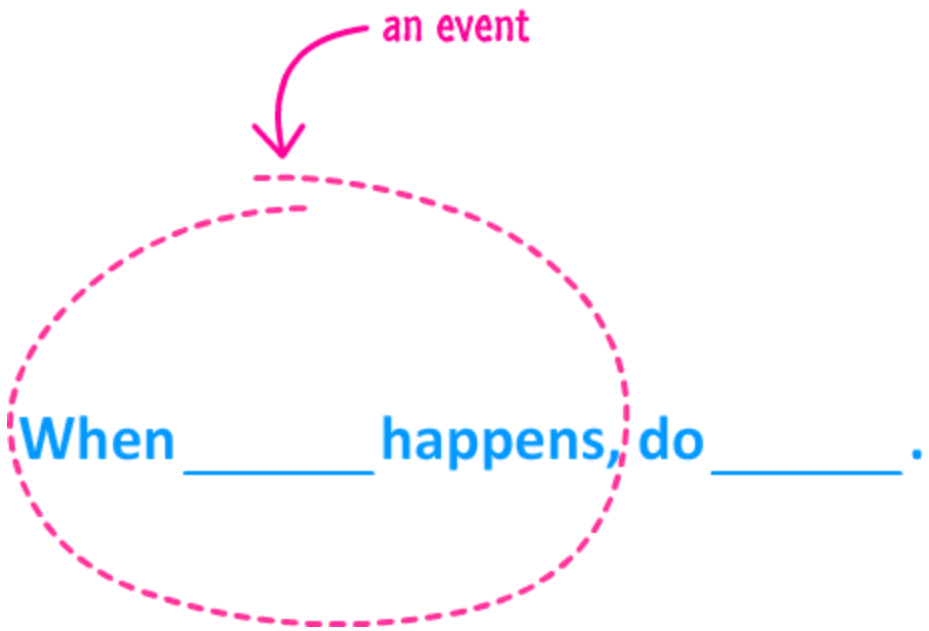
When a mouse release happens, do hurl the giant/not-so-happy bird.

When a delete key press happens, do send this file to the Recycle Bin.

When a touch gesture happens, do apply this old timey filter to this photo.

When a file download happens, do update the progress bar.

What is an *Event*?



Handling Events

- **Event Listener**
  - A function that listens for a specific event to occur on a target element, and performs an action in response to that event.
  - Syntax: `EventTarget.addEventListener()`
    - This method takes two arguments:
      - the type of event to listen for (e.g. "click")
      - the function to be called when the event occurs
    - Example: `element.addEventListener('click', myFunction)`;
- **Event Handler**
  - Syntax: **on*event***
  - More in the next slide

Registering *onevent* Handlers

- The **on*event*** handlers are properties on certain DOM elements to manage how that element reacts to events.
- Two common ways to register event handlers:
  i. Adding an HTML *attribute* named on:

```
<button onclick="handleClick()"> <!-- Again, it is a bad practice. -->
```

  ii. Setting the corresponding property from JavaScript:

```
document.querySelector("button").onclick = function(event) {
    ...
}
```

- **Practice:** modify *lec13-js-demo.html*, so it uses **the second way** to handle the event of clicking on the button.

Object Properties

- In JavaScript, everything is treated as an ***object***
- ***Properties*** describe the characteristics of an object.
  - Use *dot notation*: `object.property`
    - Examples:
      - `document.title` - the title property of a web page doc
      - `image.src` - the source property of the image element
  - Different types of objects have different properties.

Object Methods

- *Methods* are functions that are performed by an object
  - Think of them as verbs that perform actions on the object.
- Use *dot notation*: `object.method(arguments)`
  - `arguments` could be empty.
- Examples:
  - `document.getElementById("a")`
    - `document` - the object.
    - `getElementById("a")` - the method that is part of this object.
    - It gets the "puppet strings" to the element whose `id` is `"a"`
  - `console.log(message)`
    - Outputs `message` to the web console.
    - This is an important way to debug and to display messages for the user.

Practice

Play with *lec13-js-demo.html* with what you just learned.

Questions?