Topic 2 — JavaScript and the DOM (Sept. 21)

- DOM: Document Object Model
- JavaScript used for manipulating and editing web pages

- Reflect on what is happening behind the scenes of websites while browsing the internet
- Key to understand how each part is implemented

Web Development

- Involves a *client* and a *server*
  - Examples will be client-side programming today
- Manipulating the HTML that the client would generally receive from the server
  - Currently don't have a server currently so we'll use locally stored HTML files
- Focus on the concepts

Problem Solving

- Think before you start coding
- Compare/contrast the various solutions
  - Some ways are better/more efficient than others

Role of JavaScript

- Use JavaScript to make dynamic web pages
  - Interactive web pages such as adding buttons, text boxes, check boxes, etc.
- Javascript will be used to execute based on the user interaction to make changed to the web page
- Originally intended for client-and-server-side development

- Beginning with client-side programming

Event-Based Programming

- User makes a request (e.g. user clicks button), we process it (e.g in the code we add if statements to cause a certain outcome) and decide how respond (e.g. output cause by the user's request and the code written)

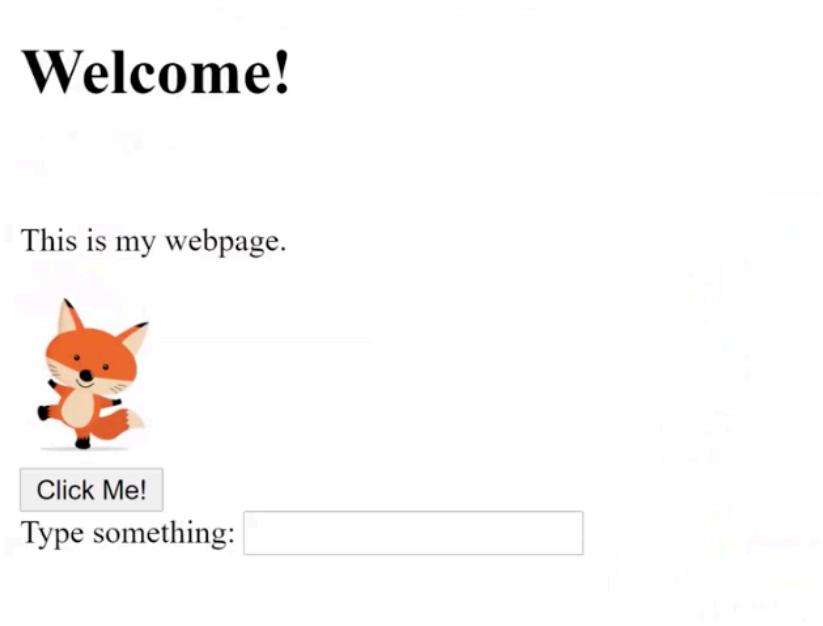
Basic Web Page

- Web page: a document that can be displayed in a browser window or as an HTML source — Same document in both cases, no separation
- When we load the page in the browser it gets parsed and appears on the screen

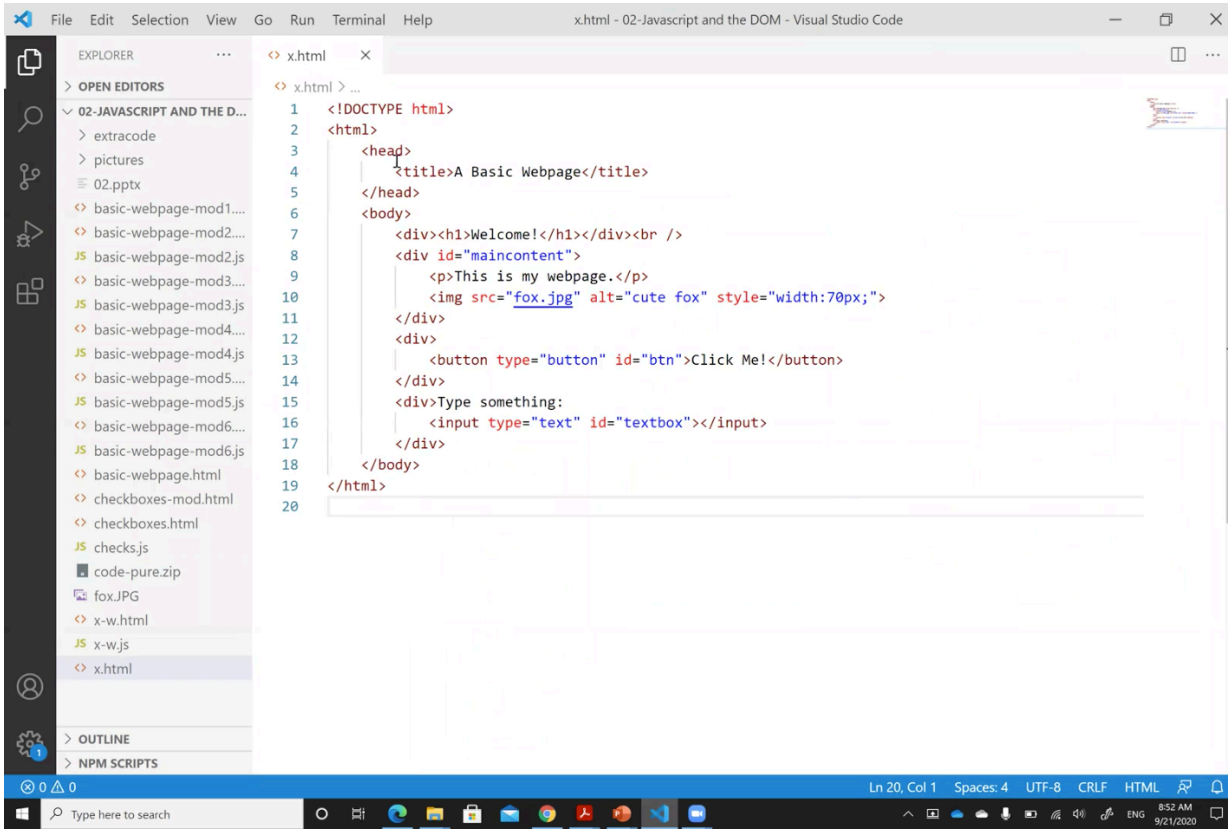
- The browser uses an underlying model of the document (called the DOM) that is used to represent the structure of the current webpage

Coding Example:

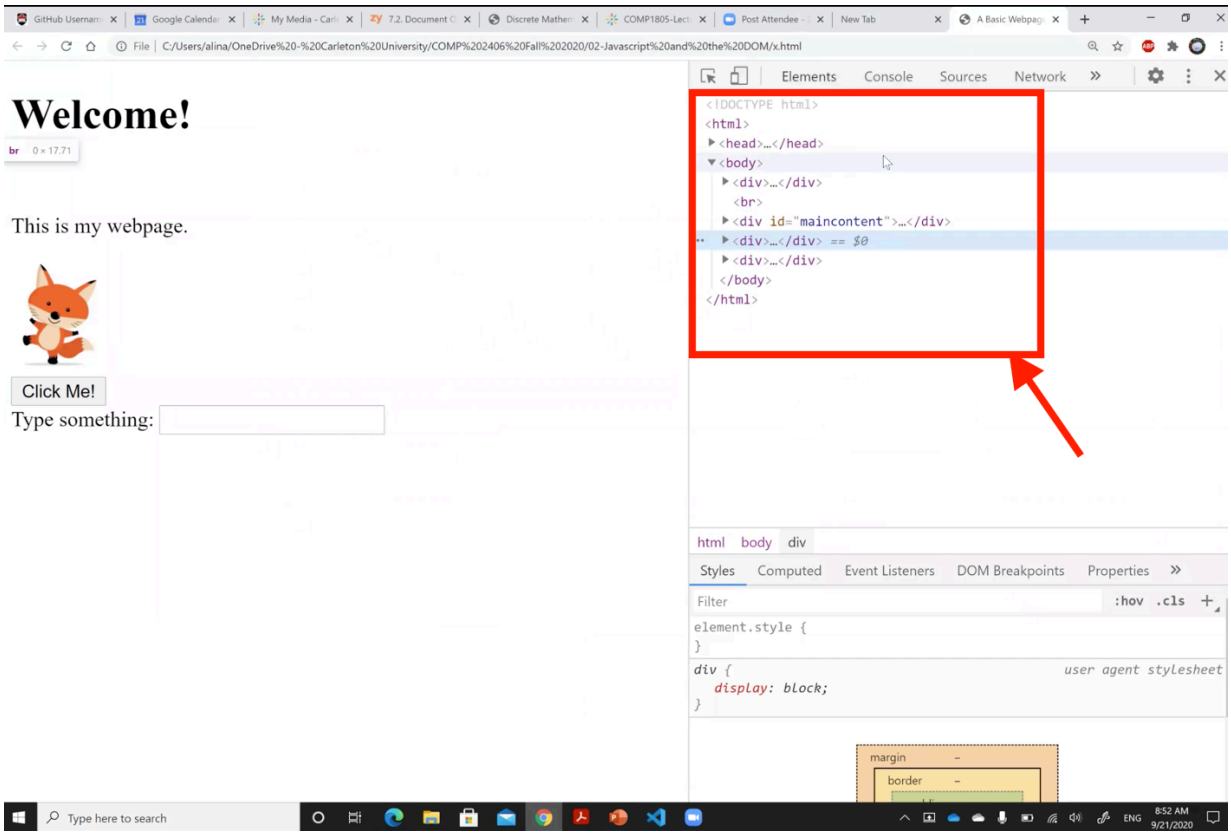
Browser HTML Page:



Code for HTML Page:



Document Object Model (DOM): Reflects what we have on the web page



- can be imagined as “invisible boxes”
  - Each reserves a space for whatever you have coded within

- Adding <script> tags to change background colour that references to

```
<div id="maincontent">
    <p>This is my webpage.</p>
    <img src="fox.jpg" alt="cute fox" style="width:70px;">
</div>
```

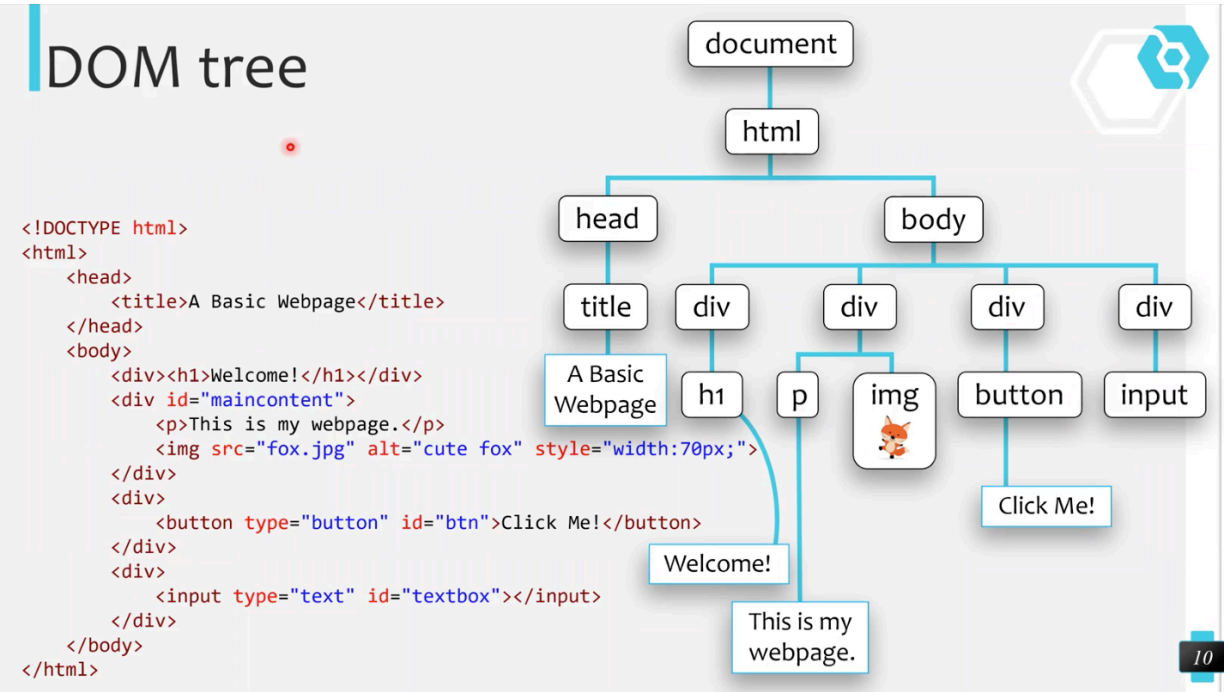
```
<script>
    let x = document.getElementById("maincontent");
    x.style.backgroundColor="green";
</script>
```

What is a DOM?

- A Document Object Model (DOM) is a W3C or World Wide Web Consortium that provides an interface to dynamically modify a page displayed in a browser (see image above)
- Allows JavaScript to edit the document structure, style, and content
- Represents a document as a tree of objects

DOM Tree

- Node for each element
- Hierarchy where body is the parent node, to each div which are child nodes



JavaScript and the DOM

- Using JavaScript we can:
  - Make changes, remove and add HTML elements and CSS styles
  - Respond to actions on HTML elements (clicks, mouse hoover, etc.)
  - Create new events to respond to

Browser Object Model (BOM)

- BOM allows “communication” between JavaScript and the browser

- Browser objects
  - Window (open, close, print, scroll, resize, etc.)
  - User’s screen
  - Document inside the window
  - Navigation buttons
  - Location or URL
  - Popup boxes, etc.

The document Object

- JavaScript code in the browser has access to a **document** object
  - This object represents the root node of the DOM tree
- **document** object is the owner of all other objects in your web page

Finding an Element

- In order to make changes we need to get an element from the page
- Everything on a web page is stored in some element
- Using the **id** HTML attribute within elements, we provide a unique name that can be used to refer to the specific element

- The **document** object allows us to find an element by the element id:

```
document.getElementById("id-name")
```

- Method will return the elements on the page with the given **id**, or **null** if there are no matches
- Many possibilities once we have a variable referencing an element

First Modification

- Setting the inner HTML content of some element

```
let x = document.getElementById("maincontent");
x.innerHTML = "You can modify any HTML you want!";
```

- Above image accesses by ID but there are other ways to access such as class

Coding Example:

- Editing script tag to change webpage to remove image and <p> tag but does not change HTML
- Overlap of JavaScript in HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>A Basic Webpage</title>
        <style>
            body {background-color: light
            div {color: blue}
            #maincontent {color: red}
        </style>
    </head>
    <body>
        <div><h1>Welcome!</h1></div><br /
        <div id="maincontent">
            <p>This is my webpage.</p>
            <img src="fox.jpg" alt="cute
        </div>
        <div>
            <button type="button" id="btn
        </div>
        <div>Type something:
            <input type="text" id="textbo
        </div>
        <script>
            let x = document.getElementBy
            x.innerHTML = "You can modify
        </script>
    </body>
</html>
```

```
<script>
    let x = document.getElementById("maincontent");
    x.innerHTML = "You can modify your page !" + "<br />" +
    '<img src="fox.jpg" alt="cute fox" style="width:100px;">';
</script>
```

- Separate JS & HTML

Responding to Events

- In general we respond to events and handle those events by making changes
- A JS can be executed when an event occurs, such as a user clicking on an HTML element
- Various event types

Browser Event Types

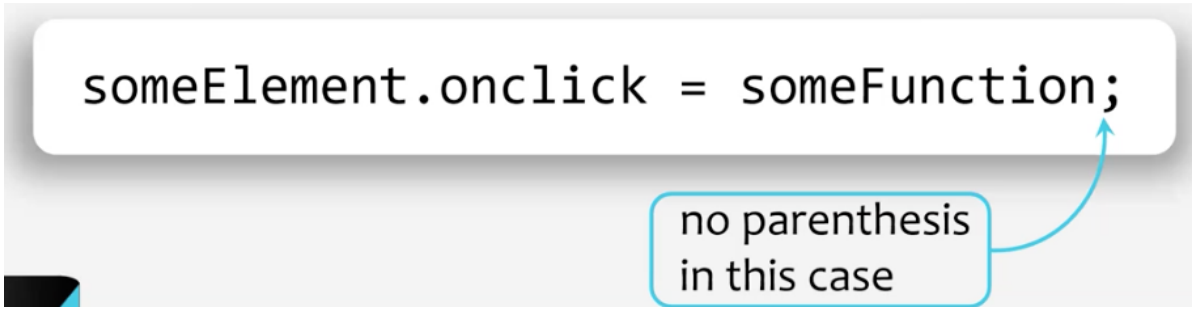
- Clicks
- Loading/unloading page
- Loading an image
- Mouse movement
- Input fields change, keyboard buttons
- Form is submitted
- Etc.

The onclick Event

- Triggered when one of the HTML elements is clicked by the user

```
<p onclick="this.innerHTML = 'Ouch!'">Click on this text!</p>
```

- onclick is an attribute of the element, so you can also add a handler through JS
- Where someElement is a variable representing an HTML element, then the function will be executed when the element is clicked



Coding Example:

- In HTML:

```
<div>
    <button type="button" id="btn" onclick="buttonClicked()">Click Me!</button>
</div>
```

- In JS:

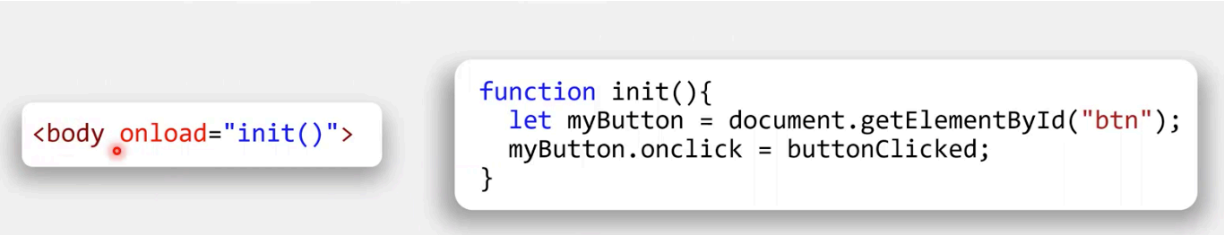
```
let count = 0;


function buttonClicked(){
    count++;
    let x = document.getElementById("maincontent");
    x.innerHTML = "You clicked the button " + count + " times";
}
```

The onload Event

- Triggered when an element on the page is loaded by the browser
- Typically used on the <body> tag for initialization
- ** There is also an **onunload** event when the user leaves the page

- To add an unload even that will create a button handler



The onchange Event

- Triggered when:
  - a radio button is checked/unchecked
  - a checkbox is checked/unchecked
  - Text and other components lose focus (after being modified)
- Typically use validating inputs or selections

The onblur Event

- **onchange** event is only triggered when the text in a field is modified
- To trigger an event whenever focus is lost you use **onblur**

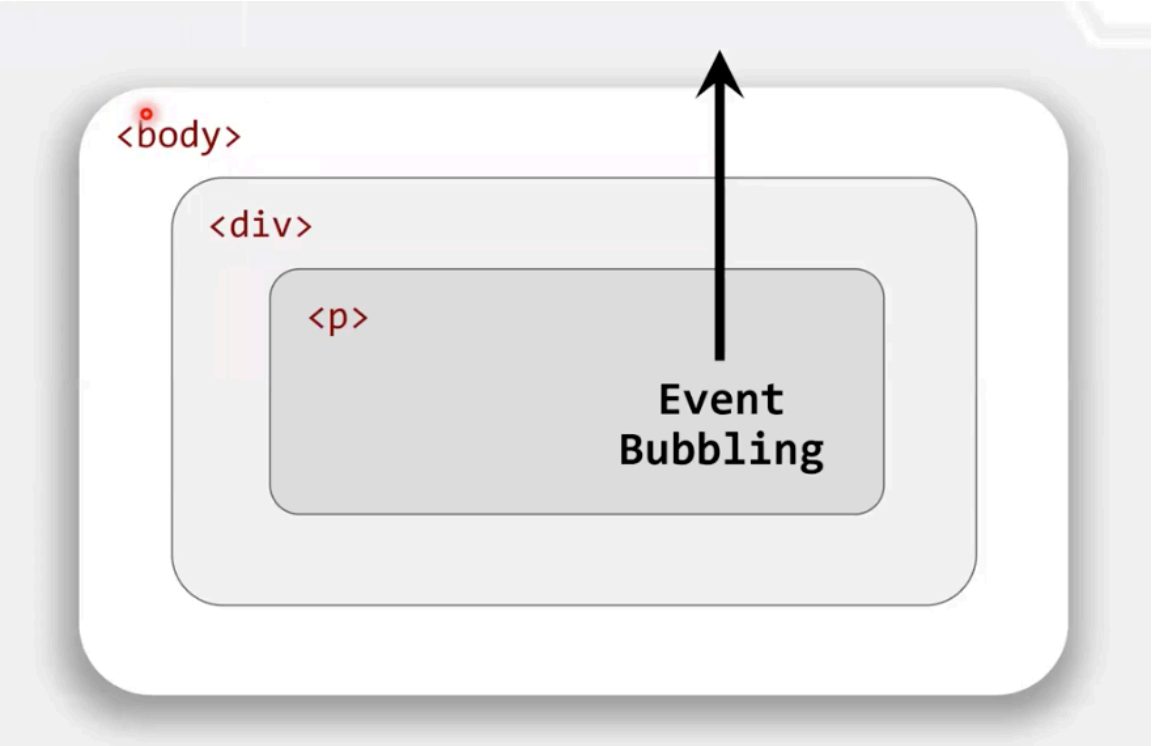
Mouse Events

- onmouseover - mouse enters space element
- onmouseout - mouse leaves space element

- onmousedown - mouse button is pushed down
- onmouseup - mouse button is lifter
- onclick - when element is clicked

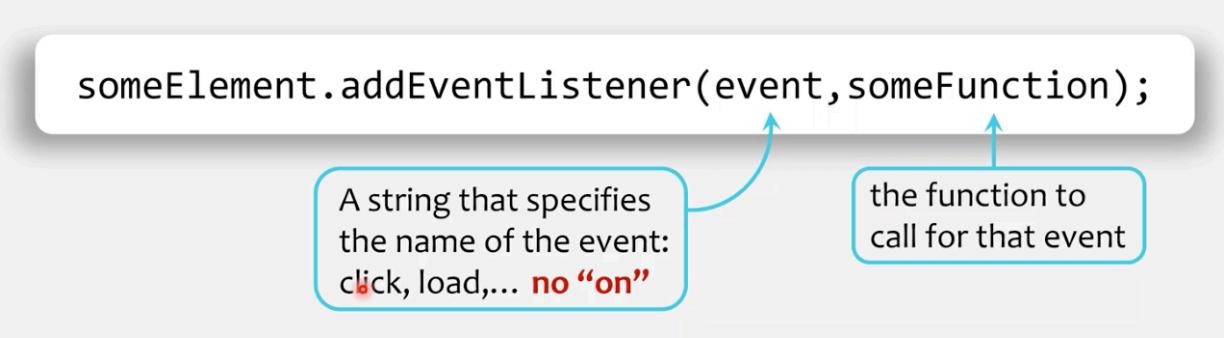
Event Propagation

- Events “bubble” up from the DOM tree to outer elements
- To stop the bubbling you can add: event.stopPropagation() to your code



Event Listener

- To add event handlers you can add:



- addEventListener() doesn't overwrite existing event handlers which provides flexibility

- We can add multiple event handler to one element
- We can add event handlers of the same type to one element
- We can add handlers to a page (even when not in HTML)
- We can add event listeners to any DOM object (not specific to HTML elements)
- Allows for a separation in the display content (HTML) and the behaviour (JS handlers)