

CS 2033

# Multimedia & Communications II

LECTURE 6 – JAVASCRIPT

# JavaScript

2

- ▶ So far our websites have been static. What if we want to make the website dynamic and interactive?
- ▶ HTML and CSS are limited.
- ▶ We can do so much more by adding JavaScript code.
- ▶ It is simple to incorporate JS (JavaScript) in websites.

# JavaScript

3

## ► Benefits of JavaScript:

- User input (mouse, keyboard, etc.)
- Modify HTML elements and CSS dynamically (change styles, etc.)
- Analyze data like user input (validating form input, etc.)
- Change content based on specific conditions (different message for students vs. teachers, etc.)

# JavaScript

4

- ▶ First JavaScript commands:
- ▶ `alert("Hello world");`
  - ▶ `alert(_____)` will open a popup window to display the given text.
- ▶ `document.write("Hello world");`
  - ▶ `document.write(_____)` will add the given text directly in the webpage.

# JavaScript

5

- ▶ Variables make it easy to store items that are reused.

- ▶ `var course = "CS2033";`

- ▶ Now we can use this variable by using its name.

- ▶ `alert(course);`

- ▶ `document.write(course);`

# JavaScript

6

- ▶ Variable values can be changed after they are first created.
- ▶ 

```
var course = "CS1033";  
document.write(course);  
course = "CS2033";  
document.write(course);
```
- ▶ Code is executed in top-to-bottom order, so this would first write "CS1033" and then "CS2033".

# JavaScript

7

- ▶ There are several types of variables:

- ▶ String – text, surrounded by quotations

*var x = "hello";*  
▶ x = "hello";

- ▶ Integer – whole number

- ▶ x = 13;

- ▶ Float/Double – decimal number

- ▶ x = 7.3

- ▶ Array – list of multiple items

- ▶ x = [2, 7, 4, 8, 1];

# JavaScript

8

- ▶ Arrays are indexed such that each item has a position, starting at 0.
- ▶ Access individual elements using square brackets and an index.
- ▶ `x = [2, 7, 4, 8, 1];`  
`alert(x[0]); // Displays 2.`  
`x[4] = 3; // Changes last item to 3.`



# JS and CSS

- ▶ A major benefit of JS is its ability to interact with HTML and CSS.
- ▶ Directly change elements' styles.
- ▶ Change classes or IDs.
- ▶ Advanced animations / transitions.
- ▶ Content can also be changed.
- ▶ And more!

# JS and CSS

10

- ▶ How do we change a style?
- ▶ First, we must select/get the HTML element(s) in JS.
- ▶ Several ways to do this:
  - ▶ `document.getElementById(id);`
  - ▶ `document.getElementsByTagName(tag);`
  - ▶ `document.getElementsByTagName(ClassName)(class);`

Notice the id getter is singular while the tag and class getters are plural.

# JS and CSS

11

- ▶ The ID element getter is typically the best one to use since it returns one specific object.
- ▶ Getting elements by tag or class may be helpful in specific cases in which an array of elements need to be accessed at once.

# JS and CSS

12

- ▶ Suppose you have an HTML div with the ID "mydiv".
- ▶ Access that element in JS with:
  - ▶ `document.getElementById("mydiv");`
- ▶ Simplify the code by storing this element in a variable.
  - ▶ `var md = document.getElementById("mydiv");`

# JS and CSS

13

- ▶ Now the styles can be modified.
- ▶ JS uses **dot notation** for accessing levels of properties.
  - ▶ style is a property in HTML elements.
  - ▶ CSS styles are properties within style.
- ▶ Type the element (dot) style (dot), then a style property to access it.
  - ▶ i.e. `md.style.width`

# JS and CSS

14

- ▶ To modify a style, use the dot notation on the element.
- ▶ After specifying a style, simply use an equal sign and set the new value in quotation marks.
  - ▶ `md.style.width = "100px";`
  - ▶ `md.style.margin = "5px";`
  - ▶ `md.style.color = "red";`

# JS and CSS

15

- ▶ Style properties that contained a hyphen in CSS are defined differently in JavaScript.
- ▶ Instead, the words are back to back and the initial of each word (except the first) is capitalized.
  - ▶ i.e. backgroundColor (not background-color)
- ▶ This is called camel case.

# JS and CSS

16

- ▶ `box.style.backgroundColor = "red";`
- ▶ `box.style.color = "rgb(100,78,19)";`
- ▶ `mydiv.style.width = "400px";`
- ▶ `mydiv.style.borderColor = "black";`
- ▶ `mydiv.style.borderWidth = "3px";`
- ▶ `mydiv.style.left = "50%";`
- ▶ `tb.style.display = "block";`
- ▶ `tb.style.display = "none";`



# JS and CSS

17

- ▶ If several styles need to be changed, it's inefficient to do each one individually.
- ▶ A better option is to change the class on the object within JS.
- ▶ In CSS, an element can have multiple classes. It's the same in JS.

# JS and CSS

18

- ▶ Get the element the same way:
  - ▶ `var tb = document.getElementById("topbox");`
- ▶ Use dot notation to access **className** or **id** on the object.
- ▶ It's safer to leave the id alone and just change classes.

# JS and CSS

19

## ▶ HTML:

▶ `<div id="tb" class="box"></div>`

## ▶ JS:

▶ `var tb = document.getElementById("tb");`

▶ `tb.className = "newbox"; // Single class.`

▶ `tb.className = "box newbox"; // 2 classes.`  
① ②

# JS and CSS

20

- ▶ Notice the difference between the two class change options.
  - ▶ Single class is usually a replacement.
  - ▶ Multiple classes are usually additions.
    - ▶ Retain original class styles.
    - ▶ Additional class(es) provide new or modified styles.

# JS and CSS

21

- ▶ We can now add and remove CSS animations dynamically with JS.
- ▶ Create a CSS class selector that triggers an animation.
- ▶ Use JS to add/remove the class to an element.
  - ▶ Adding the class starts the animation.
  - ▶ Removing the class stops it.

# JS and CSS

22

- ▶ JS can also change the content within an HTML element.
- ▶ Use dot notation to access the `innerHTML` property of an element.
- ▶ `tb.innerHTML = "Text shown in div.";`
- ▶ As its name suggests, this supports HTML as well.
- ▶ `tb.innerHTML = "<p>Hi there!</p>";`

# Event handling

23

- ▶ Style/content changes will typically be done as a result of an event.
- ▶ Events are a huge part of JS!
  - ▶ User input events
    - ▶ Mouse-based
    - ▶ Keyboard-based
  - ▶ Load event
  - ▶ Timer events

# Event handling

24

- ▶ Events are handled with **event listeners** applied to HTML elements.
- ▶ Event listeners are always watching for specific events to occur.
- ▶ When an awaited event occurs, the listener detects it and triggers the event's code.



# Event handling

25

- ▶ There are two main ways to add event listeners to elements.
  - ▶ Inline: attach the event listener as an attribute in the HTML element tag.
  - ▶ Dot Notation: use the `addEventListener` function as a property on the element using dot notation.

# Event handling

26

- ▶ Inline event listeners are the easiest.
- ▶ Although inline CSS and JS is generally frowned upon, event listeners are a different story!
- ▶ Inline event listeners start with "on" followed by an event name (sometimes abbreviated).
  - ▶ All lowercase, no spaces or hyphens.

# Event handling

27

## ▶ Common mouse events:

- ▶ Click: onclick
- ▶ Double click: ondblclick
- ▶ Mouse over: onmouseover
- ▶ Mouse out: onmouseout
- ▶ Focus a form field: onfocus
- ▶ Leave a form field: onblur
- ▶ Change a form value: onchange

# Event handlers

28

- ▶ Keyboard events:
  - ▶ Key press: `onkeypress`
  - ▶ Key down: `onkeydown`
  - ▶ Key up: `onkeyup`
- ▶ Differentiate `keypress`, `keydown`, and `keyup` events.

# Event handlers

29

- ▶ `<div id="box" onclick="document.getElementById('box').style.width = '300px'"></div>`
- ▶ In this example, the style change applies to the element itself, so we can use a shortcut: `this`
- ▶ `<div id="box" onclick="this.style.width = '300px'"></div>`

# Event handlers

30

- ▶ Dot notation is the other method for creating event listeners.
- ▶ This can go in internal or external JavaScript code (after the element is created!)
- ▶ `box.addEventListener("click", function() { this.style.width = "300px" });`

# Other events

31

- ▶ We talked about mouse events and keyboard events already.
- ▶ There are two other common event types in JS: the loading of the webpage and custom timers.
- ▶ The loading is simple. Just attach an **onload** listener to the body and it will trigger when everything loads.

# Other events

32

- ▶ There are two types of timers.
- ▶ Timeout: trigger a function once after a specified time has elapsed.
- ▶ Interval: trigger a function repeatedly in intervals of the specified time.
- ▶ Start them with setTimeout(function, ms) or setInterval(function, ms)



# Other events

33

▶ Examples of various events:



<https://www.csd.uwo.ca/courses/CS2033b/samples/lec6/>

# JavaScript

34

- ▶ Most lines of code end with a semi-colon ;
- ▶ There are structures of code that may contain multiple lines in a sequence.
- ▶ The first and last line of those structures end with a curly bracket { or } instead of a semi-colon.

```
{  
  - - -  
}
```

# Conditionals

35

- ▶ Conditionals are an important structure in programming.
- ▶ Portions of code will only execute if specific conditions are met.
- ▶ For example, checking if a number is less than 10
- ▶ The if-else statement is used for conditionals in JavaScript.

# Conditionals

36

- ▶ if (**condition**) {  
    // do stuff;  
}

- ▶ The condition can be anything that boils down to a True or False value (this is called a **Boolean** variable).

- ▶  $x > 5$  (*greater than*)

- ▶  $x == 1$  (*note the double equal signs*)

- ▶  $\text{course} \neq \text{"CS2033"}$  (*not equal*)

# Conditionals

37

- ▶ We can add multiple conditionals using the **else if** operator.

- ▶ 

```
if (x == 1) {  
    // do stuff.  
} else if (x == 2) {  
    // do different stuff.  
} else if (x == 3) {  
    // do other different stuff.  
}
```

# Conditionals

38

- ▶ The **else** operator is a catch-all for any cases not yet accounted for.

- ▶ 

```
if (x == 1) {  
    // do stuff.  
} else if (x == 2) {  
    // do different stuff.  
} else {  
    // do other different stuff.  
}
```

# Conditionals

39

- ▶ Note that once a condition is satisfied, the "else-if" statements below it will not be checked.

- ▶ 

```
x = 2;  
if (x < 5) {  
    // do stuff.  
} else if (x < 10) {  
    // do different stuff.  
}
```

# Conditionals

40

- ▶ Suppose we received the user's age online and then checked if the user was allowed to buy booze...
- ▶ 

```
if (age >= 19) {  
    canDrink = true;  
} else {  
    canDrink = false;  
}
```



# Conditionals

41

- ▶ Conditionals might be composed of multiple sub-conditions that all have to be met.
- ▶ We combine them with the `&&` (and) operator.
- ▶ 

```
if (age >= 19 && pregnant != true) {  
    canDrink = true;  
} else {  
    canDrink = false;  
}
```

# Conditionals

42

- ▶ There also might be sub-conditions such that at least one must be met.
- ▶ In this case, use the `||` (**or**) operator.
- ▶ 

```
if (age >= 19 || allowed == true) {  
    canDrink = true;  
} else {  
    canDrink = false;  
}
```

# Functions

43

- ▶ Another common structure in code is a **function**.
- ▶ A function is a process that can be executed at any time, and any number of times.
- ▶ Great for routine processes that need to be used multiple times.

# Functions

44

```
▶ function myFunction() {  
    // code here.  
    // code here.  
}
```

▶ The above code **creates** a function but does not actually call (run) the function. Calling it looks like this:

```
▶ myFunction();
```

# Functions

45

- ▶ Many functions have input **parameters** which are placed within the parentheses.
- ▶ Parameters make the function reusable and flexible to work in different scenarios.
- ▶ To call a function with parameters, include the parameter values in the function call parentheses.

# Functions

46

```
▶ function average(x, y) {  
    var z = (x + y) / 2;  
    document.write("Result: " + z);  
}
```

▶ This has 2 parameters: x and y.

▶ Call this function:

▶ `average(5,9);` // Result: 7

▶ `average(10,20);` // Result: 15

# Loops

47

- ▶ Another special code structure is the loop. This is used to run code repeatedly in a row.
- ▶ There are two main types of loops: while-loop and for-loop but we will focus on the for-loop.
- ▶ They contain 3 parts: variable initialization, condition, increment.

# Loops

48

- ▶ 

```
for (x = 0; x < 5; x++) {  
    alert(x);  
}
```

- ▶ They also work great with arrays.

- ▶ 

```
data = [9, 4, 7, 3];  
for (x = 0; x < data.length; x++) {  
    alert(data[x]);  
}
```