CSE 106

Lecture 8 – JavaScript

Acknowledgement: w3schools.com, developer.mozilla.org

JavaScript Functions Continued

- JavaScript function arguments are passed by value
  - Values are copied into the function
  - If a function changes an argument's value, it doesn’t change the original value

```
function tryToChangeMe(value) {
    value = 10;
}
let number = 20;
tryToChangeMe(number);
console.log(number);
```

JavaScript Functions Continued

- Objects passed as function arguments are passed by reference
  - Objects (non-primitives) are given a reference to a value (like a pointer)

```
function tryToChangeMe(value) {

   value.color = "blue";

}

let fruit = {color:"red", weight:.5};

tryToChangeMe(fruit);

console.log(fruit);
```

JavaScript Functions Continued

- Function definitions can be passed as arguments into other functions

```
function greeting(greeting_func, name) {
  greeting_func(name);
}
function hello(name) {
    alert('Hello ' + name);
}
const name = "Sam";
greeting(hello, name);
```

JavaScript Classes

- Introduced in ES6 (2015)
- A JavaScript class is not an object, it is a template for an object
- Has a method named constructor() to initialize member variables
- You can add methods to the class that have access to members

```
class ClassName {
  constructor() { ... }
  method_1() { ... }
  method_2() { ... }
  method_3() { ... }
}
```

JavaScript Classes

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  age(current_year) {
    return current_year - this.year;
  }
}

let myCar = new Car("Ford", 1982);
console.log("My car is " + myCar.age(2021) + " years old.");
```

JavaScript `this` Keyword

- `this` has different values depending on where it is used:
  - **In a method, `this` refers to the owner object**
  - Alone, `this` refers to the global object*
  - In a function, `this` refers to the global object*
  - In a function, in strict mode**, `this` is undefined
  - **In an event, `this` refers to the element that received the event**

* The global object is an object in JavaScript that holds global variables and functions

** Strict mode is a more secure variant of JS that prohibits sloppy syntax leading to errors

JavaScript - Loops

- For and while loops syntax is similar to C++ or Java

```
let text = "";
for (let i = 0; i < 10; i++) {
  text += "The number is " + i + "\n";
}

while (i < 10) {
  text += "The number is " + i + "\n";
  i++;
}
```

JavaScript - Loops

- For In Loop let’s you iterate through all items in an object or array

```
const person = {fname:"John", lname:"Doe", age:25};

let text = "";

for (let x in person) {

  text += person[x];

}

const numbers = [45, 4, 9, 16, 25];

let txt = "";

for (let x in numbers) {

  txt += x;

}
```

JavaScript - if else and else if

- Syntax is similar to C++ or Java

```
const time = 9;
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

JavaScript - Comparison Operators

- JS uses both double equals (==) and tripe equals (===) comparisons
- Given that x = 5, the table explains the comparison operators:

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
		x == "5"	**true**
===	equal value and equal type	x === 5	true
		x === "5"	**false**
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== 5	false
		x !== "5"	true
		x !== 8	true

Wake-up!

- https://youtu.be/pmTr0oCx6Og

JavaScript Events

- “Things” that happen to HTML elements
- Can triggered by the browser or user’s action
- Here are some examples of HTML events:
  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked
- JavaScript lets you execute code when events are detected

JavaScript Events

- Event handler attributes can be added to HTML elements
- HTML allows event handler attributes, with JavaScript code, to be added to HTML elements `<element event="some JavaScript">`

```
<body>

<button onclick="document.getElementById('demo').innerHTML=Date()">The
time?</button>

<p id="demo"></p>

</body>
```

JavaScript Events

- More common to see events calling JS functions

```
<body>
  <script>
  function displayDate() {
    document.getElementById("demo").innerHTML = Date();
  }
  </script>
  <button onclick="displayDate()">The time?</button>
  <p id="demo"></p>
</body>
```

Common Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

jQuery

- Very popular JavaScript library (used by ~75% of websites)
- Declining in popularity due to the introduction of front end frameworks (e.g. Angular, React, Vue), but still valuable
- Simplifies HTML DOM tree traversal and manipulation, event handling, CSS animation, and Ajax
- Accomplish more, with less code
- It can be used with JavaScript (just a JS library)

jQuery - Getting Started

- Download the jQuery library from jQuery.com OR
- Include jQuery from a CDN, like Google
  - This is the easiest route (see below)

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>
</head>
```

jQuery DOM Selectors

- JavaScript – find element by Id

```
myElement = document.getElementById("id01");
```

- jQuery – find element by Id

```
myElement = $("#id01");
```

```
<p id="id01">Hello World!</p>
<p id="demo"></p>
<script>
  var myElements = $("#id01");
  $("#demo").text("The text from id01 is: " + myElements[0].innerHTML);
</script>
```

jQuery DOM Selectors

- JavaScript – find element by Tag name

```
myElements = document.getElementsByTagName("p");
```

- jQuery – find element by Tag name

```
myElements = $("p");
```

```
<p>Hello World!</p>
<p>Hello Sweden!</p>
<p id="demo"></p>
<script>
  var myElements = $("p");
  $("#demo").text("The text in the first paragraph is" + myElements[0].innerHTML);
</script>
```

jQuery DOM Selectors

- JavaScript – find element by Class name

```
myElements = document.getElementsByClassName("intro");
```

- jQuery – find element by Class name

```
myElements = $(".intro");
```

```
<p class="intro">Hello World!</p>
<p class="intro">Hello Sweden!</p>
<p id="demo"></p>
<script>
  var myElements = $(".intro");
  $("#demo").text("The text from id01 is: " + myElements[0].innerHTML);
</script>
```

jQuery Set/Get Text Content

- JavaScript – set/get text content

```
myElement.textContent = "Hello Sweden!"; //set

myText = document.getElementById("02").textContent; //get
```

- jQuery – set/get text content

```
myElement.text("Hello Sweden!"); //set
myText = $("#02").text(); //get
```

jQuery Set/Get HTML Content

- JavaScript – set/get HTML content

```
myElement.innerHTML = "<p>Hello World</p>"; //set

content = myElement.innerHTML; //get
```

- jQuery – set/get HTML content

```
myElement.html("<p>Hello World</p>"); //set

content = myElement.html(); //get
```

jQuery CSS Styles

- JavaScript – Hide/Show an HTML Element

```
myElement.style.display = "none";  //hide

myElement.style.display = "";      //show
```

- jQuery – Hide/Show an HTML Element

```
myElement.hide();

myElement.show();
```

jQuery CSS Styles

- JavaScript – Change the font size

```
document.getElementById("demo").style.fontSize = "35px";
```

- jQuery – Change the font size

```
$("#demo").css("font-size","35px");
```