CSE-465

Web Programming

Core Concepts of JavaScript

Md. Saidul Hoque Anik
anik@cse.uiu.ac.bd

The overview

- HTML is the markup language that we use to structure and give meaning to our web content, for example defining paragraphs, headings, and data tables, or embedding images and videos in the page.
- CSS is a language of style rules that we use to apply styling to our HTML content, for example setting background colors and fonts, and laying out our content in multiple columns.
- JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

JavaScript

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive.

JavaScript contains a standard library of objects (Array, Date, Math etc.) and a core set of language elements (operators, control structures, statements). Core JavaScript can be extended for a variety of purposes.

- **Client-side JavaScript** extends the core language by supplying objects to control a browser and it's Document Object Model (DOM).

  For example, client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.

- **Server-side JavaScript** extends the core language by supplying objects relevant to running JavaScript on a server.

  For example, server-side extensions allow an application to communicate with a database, provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

Example

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>Some Web Page</title>
	</head>
	<body>
		<h1>Some Web Page</h1>
	    <!-- Rest of the page content -->
	<script>
		//Write your code here
	</script>
	</body>
</html>
```

Input

```
<script>
    let person = prompt("Please enter your name", "Mr. A");

</script>
```

Output

```
<script>
    let person = prompt("Please enter your name", "Mr. A");

    alert("Hello there, " + person);
</script>
```

Output

```
<script>
    let person = prompt("Please enter your name", "Mr. A");

    console.log("Hello there, " + person);
</script>
```

Data Types

Variable	Explanation	Example
String	This is a sequence of text known as a string. To signify that the value is a string, enclose it in single quote marks.	`let myVariable = 'Bob';`
Number	This is a number. Numbers don't have quotes around them.	`let myVariable = 10;`
Boolean	This is a True/False value. The words `true` and `false` are special keywords that don't need quote marks.	`let myVariable = true;`
Array	This is a structure that allows you to store multiple values in a single reference.	`let myVariable = [1,'Bob','Steve',10];` Refer to each member of the array like this: `myVariable[0]`, `myVariable[1]`, etc.
Object	This can be anything. Everything in JavaScript is an object, and can be stored in a variable. Keep this in mind as you learn.	`let myVariable = document.querySelector('h1');` All of the above examples too.

Operators

Operator	Explanation	Symbol(s)	Example
Addition	Add two numbers together or combine two strings.	`+`	`6 + 9;` `'Hello ' + 'world!';`

Subtraction, Multiplication, Division	These do what you'd expect them to do in basic math.	`-`, `*`, `/`	`9 - 3;`
			`8 * 2; // multiply in JS is an asterisk`
			`9 / 3;`

Assignment	As you've seen already: this assigns a value to a variable.	`=`	`let myVariable = 'Bob';`
Equality	This performs a test to see if two values are equal. It returns a `true`/`false` (Boolean) result.	`===`	`let myVariable = 3;` `myVariable === 4;`

Not, Does-not-equal	This returns the logically opposite value of what it precedes. It turns a `true` into a `false`, etc.. When it is used alongside the Equality operator, the negation operator tests whether two values are *not* equal.	`!`, `!==`	For "Not", the basic expression is `true`, but the comparison returns `false` because we negate it:
			
			`let myVariable = 3;`
			`!(myVariable === 3);`
			
			"Does-not-equal" gives basically the same result with different syntax. Here we are testing "is `myVariable` NOT equal to 3". This returns `false` because `myVariable` IS equal to 3:
			
			`let myVariable = 3;`
			`myVariable !== 3;`

Conditional Statement

```
let person = prompt("Please enter your name", "Mr. A");
if (person === null)
{
        console.log("Hello Stranger!");
}
else
{
        console.log("Hello there, " + person);
}
```

Compound Conditional Statements

```
let person = prompt("Please enter your name", "Mr. A");
if (person === null)
{
        console.log("Hello Stranger!");
}
else if (person === '')
{
        console.log("Hello Unnamed!");
}
else
{
        console.log("Hello there, " + person);
}
```

Complex Conditional Statement

```
let person = prompt("Please enter your name", "Mr. A");
if (person === null || person === '')
{
        console.log("Hello Stranger!");
}
else
{
        console.log("Hello there, " + person);
}
```

Conversion to integer

```
var a = "10.5";
var b = Number(a);
```

Other functions:

- parseInt()
- parseFloat()

Read More:
https://gomakethings.com/converting-strings-to-numbers-with-vanilla-javascript/

For Loop

```
for (let i = 0; i < 5; i++) {
  console.log("The number is " + i);
}
```

Read More:
https://www.w3schools.com/js/js_loop_for.asp

For Loop

```
let value = prompt("Please enter a number", "10");
let n = Number(value);
for (let i = 0; i < n; i++) {
  console.log("The number is " + i);
}
```

Read More:
https://www.w3schools.com/js/js_loop_for.asp

Function

```
function getSquare(a){
        return a * a;
}
```

Function

```
let value = prompt("Please enter a number", "Mr. A");
let n = Number(value);
for (let i = 0; i < n; i++) {
  console.log("The number is " + i + ", and square is " + getSquare(i));
}
```

Using a DOM Element

```
let btn = document.querySelector('#btn1');
```

- querySelector is a function to select any element in the HTML page in a similar manner like CSS.
- If there are multiple elements, only the first element is selected.
- To select all the elements, we have to use querySelectorAll()

Using a DOM Element

```
<!doctype html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>Some Web Page</title>
	</head>
	<body>
		<h1>Some Web Page</h1>
		<button id="btn1">Set name</button>
	<script>
		let btn = document.querySelector('#btn1');
	</script>
	</body>
</html>
```

Adding an Event

```
btn.addEventListener('click', your_function);
```

Common Events

Event	Description
change	An HTML element (usually form input) has been changed
click	The user clicks an HTML element
mouseover	The user moves the mouse over an HTML element
mouseout	The user moves the mouse away from an HTML element
keydown	The user pushes a keyboard key
load	The event occurs when an object has loaded

The total list is much longer. See them here:
https://www.w3schools.com/jsref/dom_obj_event.asp