Introduction to JavaScript

Learn by Doing

Muhammad Yahya

Supervisor: Prof. Dr. Franz Josef Behr

December 21, 2017

Stuttgart University of Applied Sciences

Table of Contents

1. Introduction

What is JavaScript?

- A scripting language to implement interactivity on web pages
- Third layer of standard web technologies along with HTML and CSS
- Executed by the browser's JavaScript engine after HTML and CSS
- Interpreted language - runs from top to bottom and immediately returns results
- For dynamic modification of HTML and CSS
- Extension: .js

Client Side, Server Side and Dynamic

Client Side:

- a code that runs on user's computer
- in web development, code runs in browser

Server Side:

- a code that runs on server
- Node.js

Dynamic:

- change and update the display according to the user interaction and requirements
- dynamic can be both on client side as well as server side

Libraries and Frameworks

Libraries: a collection of functionalities

- jQuery
- D3.js
- MooTools
- ...

Frameworks: specify application structure

- Angular JS
- React
- Node JS
- ...

A Modern Web Page

- HTML: content handler
- CSS: presentation of HTML contents
- JavaScript: interactive layer

A Modern Web Page

- HTML: content handler
- CSS: presentation of HTML contents
- JavaScript: interactive layer

A Modern Web Page

- HTML: content handler
- CSS: presentation of HTML contents
- JavaScript: interactive layer

Browser Console

Console is the command line for the browser to deal with the JavaScript on the fly.

- Ctrl + Shift + I
- alert('Welcome to JavaScript!');

Exercise:

1. show current date as alert message
2. show current date in log
3. show current date to page body

```
1 document.body.innerHTML = ('<h1>The current date is ' + date + '<h1>')
```

Error Checking

Generally errors can be classified as:

- Syntax error
- Logic error

Other errors like:

1. missing '**;**'
2. missing '**)**'

Embedding JS

Inside the **BODY** of web page, just before closing body tag.

- Internal Embedding

```
<script>
  // JavaScript code will go here
</script>
```

- External Embedding

```
<script src="myJavaScript.js"></script>
```

Writing JS Code

- case sensitive
- camelCase
  - variables with lower case letters
  - objects and classes with upper class letters
  - constants with all upper case letters
- white space
- each statement ends with semicolon
- use comments

2. Basics of JavaScript

Basics

- Variables
- Data Types
- Arithmetic Operators
- Working with Number and Strings
- Conditions
- Arrays

Variable

Variables are containers to store the values.

- Variable name can be letters, numbers, under-score, dollar sign
- variable name has NO SPACE, cannot start with numbers

1. declare variable by using 'var' keyword
2. give a name to variable
3. assign data to store by using = sign

```
1 var x = 5;
2 var y = 4;
3 var total = x + y;
4 // to see the output in console log
5 console.log(total);
```

Variable Scope

Variable can be global or local in scope depending upon the availability of variable.

1. a locally scoped variable is defined inside the function and is available as soon as you are inside the function
2. a globally scoped variable is defined outside the function
3. un-defined variables (**var unDef;**)

```
1 // named function
2 function sum(x,y) {
3   var a = 5;
4   var b = 6;
5   return x+y;
6 }
7 console.log(sum(a,b));
8 //ReferenceError: a is not defined
```

Data Types i

There are six data types:

- numeric
- string
- boolean
- null
- undefined
- symbol

Check the type of variable in console:

```
1 console.log((typeof a));
```

Data Types ii

```
1  // numeric data handles integers and floats
2  var a = 100;
3  var b = 3.14;
4  // sting handles letters, words and sentences
5  var string = 'String handles text and requires
6  single or double qoutes at the ends.';
7  // bolean handles the binary values
8  // withour quotation marks
9  var bTrue = true;
10 var bFalse = false;
11 // null stores intentional empty value
12 // withour quotation marks
13 var emptyVariable = null;
14 // undefined has no set value
15 var undefinedVariable;
```

Arithmetic Operators i

JavaScript support basic arithmetic operators like:

+, -, *, /

```
1 var a = 8;
2 var b = 4;
3 var add = a + b;
4 var subtr = a - b;
5 var mult = a * b;
6 var div = a / b;
7 // JS follows Algebra rules
8 var total = a + (b * c);
```

Arithmetic Operators ii

JS support short hand math.

Exercise:

```
1 var a = 4;
2 a += 2;
3 a -= 2;
4 a *= 2;
5 a /= 2;
6 a ++;
7 a --;
```

Working with Number and Strings i

JS combine number and strings differently.

Exercise:

```
1 var a = 5;
2 var b = '4';
3 var c = 4;
4 var total = a + b;
5 var sum = a + c;
6 console.log(total);
7 console.log(sum);
```

Conditions i

Conditions allow to run the code under certain conditions.

```
// structure of conditional statement
if (condition is true) {
  do this;
} else {
  otherwise do this;
}
```

Following conditions can be evaluated:

=, ==, ===, <, >, <=, >=, !=,!==

Conditions ii

Exercise:

Evaluate different conditions described on previous slide.

```
1 var a = 5;
2 var b = 10;
3 if (a==b) {
4   console.log('a is equal to b!');
5 } else {
6   console.log('a is not equal to b');
7 }
```

Some String Methods

```
// open console, define a string
var myString = 'This is my string';
// length of myString
myString.length;
// accesing 1st character
myString[0];
// accesing last character
myString[myString.length-1];
// finding position of letter or substring
myString.indexOf('i');
// slice a myString
myString.slice(0, 4);
// changing case
myString.toLowerCase();
myString.toUpperCase();
// replace a part
myString.replace('old', 'new');
```

Arrays

- An array is collection of multiple values stored as a list of objects
- Arrays are created using square brackets containing list of items that are separated by commas
- An array can be created inside an array - multidimensional array

```
1 var myArray = ['a', 'b', 'c', 'd', 1, 2, 3];
2 console.log(myArray);
```

Exercise:

Some Array Methods i

```
1  //converting string to array
2  var myString = 'this is my string that will be converted into an array';
3  var myArray = myString.split(' ');
4  console.log(myArray);
5  // length of array
6  console.log(myArray.length);
7  // retrieve item at a specific position
8  //access first item
9  console.log(myArray[0]);
10 // adding a new item at a specific position
11 myArray[0] = 'newItem';
12 // access last item
13 console.log(myArray[myArray.length-1]);
14 // adding new item at the end
15 myArray.push('addItem');
16 // removing item from the end
17 myArray.pop();
```

Some Array Methods ii

```
1  // adding item in the start
2  myArray.unshift('firstItem');
3  // removing first addItem
4  myArray.shift();
5  // reverse the arrayMethods
6  myArray.reverse();
7  myArray.sort();
8  // remove item from a specific position
9  // position from where to start, number of items to remove
10 myArray.splice(2, 1);
11 // create a new arrayMethods
12 var newArray = myArray.slice();
13 // finding the position of an item
14 // zero is the index position from where we want to start search
15 myArray.indexOf('an', 0);
```

3. Functions

Functions

- functions are mini programs
- function can run immediately or provide output for other function
- functions can be called using a short command rather than writing full piece of code - re-usable block of code

The structure of function:

1. declaring function
2. give a name to function
3. pair of arguments
4. block of code
5. output

Types of Functions i

Some different type of functions:

- built-in browser function
- named functions
- anonymous functions
- custom functions

```
1 // named function
2 function sum(x,y) {
3   return x+y;
4 }
5 console.log(sum(4,5));
```

Types of Functions ii

```
1  // built-in function
2  var myString = 'built in functions performs useful things without
3  writing the full piece of code'
4  var newString = myString.replace('things', 'actions')
5  console.log(newString);
6  // invoking functions
7  function invokingFunc(){
8    alert('You have invoked the function!');
9  }
10 invokingFunc()
11 //anonymous function
12 var anonFunc = function () {
13   alert('anonymous functions do not have a name');
14 }
15 console.log(anonFunc());
```

4. Loops

Loops

Loops run the functions as many times as necessary and in some cases even endlessly.
Loops are mainly:

- for loop
- while loop

Basic Structure:

1. defining loop type - for or while
2. specifying conditions - a starting value and an exit condition
3. an incremental
4. code block in curly braces

For and While Loop

```
// basic structure of for loop
for (starting value; condition; incremental) {
  block of code;
}
// basic structure of while loop
starting value;
while (condition) {
block of code;
incremental;
}
```

For and While Loop

Exercise:

```
1  // for loop
2  for (var i = 0; i < 10; i++) {
3    console.log(i);
4  }
5  // while loop
6  var i = 0;
7  while (i <10) {
8    console.log(i);
9    i++;
10 }
```

do - while Loop

To run the while loop at least one, do - while loop is used.

```
1 // do-while loop
2 var i = 0;
3 do {
4   i += 1;
5   console.log(i);
6 } while (i < 10);
```

Break and Continue

- **Break:** to jump out from the loop and continue executing the code after the loop
- **Continue:** breaks iteration in the loop and continues with the next iteration in the loop, also continue executing the code after the loop

```
1  // break statement
2  for (i = 0; i < 10; i++) {
3      if (i === 3) { break; }
4      console.log(i);
5  }
6  console.log('This is the statement outside the loop after break!');
7  // continue statement
8  for (i = 0; i < 10; i++) {
9      if (i === 3) { continue; }
10     console.log(i);
11 }
12 console.log('This is the statement outside the loop after continue!');
```

5. DOM Manipulation

Object Models i

Objects are data models that make it possible to combine methods and properties for a particular data set in a structured way. **Browser Object Model (BOM):**
BOM models browser, windows inside the browser, document inside the window and other graphic user interface (GUI) elements like navigation buttons etc.
Window is top level object in BOM and has many properties and methods to interact with browser objects:

- window.innerWidth
- window.innerHeight
- window.open()
- ...
- window.document

Object Models ii

Document Object Model (DOM):

When a document is loaded in the browser, it is loaded into document model of the BOM. The loaded document creates its own model that is called document object model (DOM).
Every element in HTML document is DOM node. A standard HTML document has two nodes:

- head node
- body node

DOM Access Methods

The elements inside the document can be accessed by:

```
1 // get element by id
2 document.getElementByID('id');
3 // get element by class name
4 document.getElementByClassName('className');
5 // get element by HTML tag
6 document.getElementByTagName('HTMLtag');
```

Query Selector

Query selector returns the first instance that matches with the specified selector.

- document.querySelector('h1')
- document.querySelectorAll('p')

Access and Change Elements

```
// accessing the inner html property
document.querySelector('h1').innerHTML
// accessing the outer wrapper
document.querySelector('h1').outerHTML
// set the inner property to new one
document.querySelector('h1')
        .innerHTML='This is new property'
// changing id name
document.querySelector('#oldID').id='newID'
```

Access and Change Class

```
// access the name of classes
document.querySelector('.w3-bar').className
// acees all list of classes
document.querySelector('.w3-bar').classList
// add new classes
document.querySelector('.w3-bar').classList.add('new-class')
// remove a class
document.querySelector('.w3-bar').classList.remove('new-class')
// toggle between classes
document.querySelector('.w3-bar').classList.toggle('w3-theme')
// check if the class exists
document.querySelector('.w3-bar').classList.contains('w3-theme')
```

Changing Attributes

The following example access and changes the attributes of an element.

```
// access the attributes
document.querySelector('.w3schools-logo')
// create a variable to store the element
var logo = document.querySelector('.w3schools-logo')
// use variable to set new properties
logo.setAttribute('target', '_blank')
```

DOM Events

- Everything happening inside the browser is an EVENT
- Every time we interact with the browser, we fire an event and browser responses to those events in pre-defined ways

Mainly events can be classified as:

- browser-level events that deals with browser behavior
- DOM-level events that interacts with contents

Simple Mouse Click Function

Creating Button in HTML:

```
<body>
<!-- create button in html -->
<button> click here </button>
</body>
```

Assigning function in JavaScript:

```
// accessing button from HTML document
var button = document.querySelector('button');
// assigning function to button
button.onclick = function () {
  var name = prompt('Please enter your name: ');
  alert('Hi ' + name + ', welcome to learn JavaScript!');
}
```

Adding Element to DOM

```
<body>
<!-- creating div element with Paragraph in html -->
<div id="new-div"><p>The above element is newly created.</p></div>
</body>
```

```
// create a new div element
var newDiv = document.createElement("div");
// create a new text node
var text = document.createTextNode("This is the element inserted in
HTML document!");
// add the text node to the div element
newDiv.appendChild(text);
// add the element and its content to the DOM
var firstDiv = document.getElementById("new-div");
document.body.insertBefore(newDiv, firstDiv);
```

Example DOM Manipulation i

```
</body>
<!-- creating form -->
<form action="">
    <!-- creating selection menu -->
    <select id="elementType" size="1">
      <option value="h1" selected>Heading 1</option>
      <option value="h2">Heading 2</option>
      <option value="h3">Heading 3</option>
      <option value="p">Paragraph</option>
      <option value="hr">Horizontal Rule</option>
    </select>
    <!-- creating text bar to add the text -->
    <input type="text" id="elementContent" size="50">
    <!-- creating add button with on click mouse function -->
    <input type="button" value="Add" onclick="addDOMElements()">
  </form>
  <div id="container"></div>
</body>
```

Example DOM Manipulation ii

```
1  // defining a function to add dom element
2  function addDOMElements () {
3    // accessing existing dom element and storing them in variables
4    var elementType = document.getElementById("elementType");
5    var elementContent = document.getElementById("elementContent");
6    var type = elementType.options[elementType.selectedIndex].value;
7    var Element = document.createElement(type);
8    if (type != "HR") {
9      // creating text nodes and appending to elements
10     var Text = document.createTextNode(elementContent.value);
11     Element.appendChild(Text);
12   }
13   // appending elements to body
14   document.getElementById("container").appendChild(Element);
15 }
```

6. Additional Resources

Useful Links

- https://www.w3schools.com/

  Some useful books:

- A Smarter Way to Learn JavaScript: The new approach that uses technology to cut your effort in half, by Mark Myers
- JavaScript and JQuery: Interactive Front-End Web Development, by Jon Duckett
- JavaScript: The Definitive Guide, by David Flanagan

Questions?