Web Application Programming

Week 3: Client-Side Scripting with JavaScript

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com or jose@kumiuniversity.ac.ug

Agenda

1. Summary of previous lecture
2. JavaScript fundamentals
3. Variables,
4. Data types,
5. Operators

Summary of previous Lecture

1. CSS basics for responsive web page development(Pseudo-class selectors, Pseudo-elements selectors, looked how to attach CSS onto an html file etc.)
2. Responsive web design principles(Mobile-First Approach, Fluid Grid Layout, Media Queries etc.)

JavaScript fundamentals

JavaScript fundamentals

JavaScript is a versatile and widely-used programming language that primarily runs in web browsers but can also be used in server-side development (e.g., with Node.js). Here are some fundamental concepts and features of JavaScript:

1. **Variables:** JavaScript uses the **var**, **let**, and **const** keywords to declare variables. let and const are block-scoped, while var is function-scoped.

```
let x = 10;
const PI = 3.14159;
```

2. **Data Types:** JavaScript has several primitive data types, including numbers, strings, booleans, null, undefined, and symbols. It also has complex data types like objects and arrays.

```
let name = "John";
let age = 30;
let isStudent = true;
```

JavaScript fundamentals++

3. **Operators:** JavaScript supports various operators for performing operations on data, such as arithmetic, comparison, logical, and assignment operators.

```
let a = 5;
let b = 3;
let result = a + b; // 8
```

4. **Functions:** Functions are blocks of reusable code. You can declare functions using the function keyword or use arrow functions (() =>) for more concise syntax.

```
function greet(name) {
    return `Hello, ${name}!`;
}
```

JavaScript fundamentals++

5. **Conditional Statements:** JavaScript provides **if**, **else if**, and **else** statements for conditional execution of code.

```
if (x > 0) {
    // Do something
} else {
    // Do something else
}
```

6. **Loops:** You can use **for**, **while**, and **do...while** loops to execute code repeatedly.

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

7. **Objects:** Objects are collections of key-value pairs and are used for structuring data.

```
let person = {
    firstName: "John",
    lastName: "Doe",
    age: 30
};
```

JavaScript fundamentals++++

8. **Arrays:** Arrays are ordered lists of values and are often used for working with collections of data.

```
let colors = ["red", "green", "blue"];
```

9. **Events:** JavaScript can interact with HTML and respond to events like clicks, form submissions, and keyboard input.

```
document.getElementById("myButton").addEventListener("click", function() {
    alert("Button clicked!");
});
```

JavaScript fundamentals++++

10. **DOM Manipulation:** JavaScript can manipulate the Document Object Model (DOM) to dynamically update web pages.

```
document.getElementById("myElement").innerHTML = "New content";
```

11. **Asynchronous Programming:** JavaScript supports asynchronous operations using callbacks, promises, and async/await, making it capable of handling tasks like fetching data from a server without blocking the main thread.

```
fetch("https://api.example.com/data")
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.error(error));
```

These fundamentals provide a strong foundation for understanding JavaScript. JavaScript's versatility allows it to be used for front-end web development, back-end server development, and even for building desktop and mobile applications with frameworks like React, Angular, and Vue.js.

JavaScript fundamentals
Commonly Asked Questions

1. How do I get JavaScript?
2. Where can I download JavaScript?
3. Is JavaScript Free?
4. What can JavaScript do?

Clear and easy answers

1. You don't have to get or download JavaScript.
2. JavaScript is already running in your browser on your computer, on your tablet, and on your smart-phone.
3. JavaScript is free to use for everyone.

JavaScript fundamentals
Commonly Asked Questions

4. JavaScript is a complete Front-End and Backend Programming language that can do a whole lot of things.

Example.

JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute: Enter the following code on html page and run it.

```
<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello
JavaScript!"'>Click Me!</button>
```

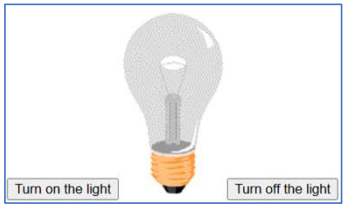
JavaScript fundamentals

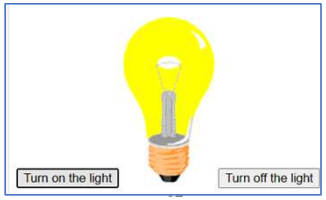
What can JavaScript do?

JavaScript Can Change HTML Attribute Values

In this example JavaScript changes the value of the src (source) attribute of an <img> tag: **NOTE:** for the code below to run as expected, you will need to create or download two gif type images named: pic_bulbon.gif and pic_bulboff.gif and place them inside the folder where your html file exists.

```
<p>JavaScript can change HTML attribute values.</p>

<p>In this case JavaScript changes the value of the src (source) attribute of an image.
</p>

<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the
light</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the
light</button>
```





JavaScript fundamentals

What can JavaScript do?

JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute:

```
<p id="demo">JavaScript can change the style of an HTML element.</p>

<button type="button"
onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!</button>
```

On button-click, the font-size of the paragraph above will change to 35px

JavaScript fundamentals

What can JavaScript do?

JavaScript Can Hide HTML Elements

Hiding HTML elements can be done by changing the display style

```
<p id="demo">JavaScript can hide HTML elements.</p>

<button type="button"
onclick="document.getElementById('demo').style.display='none'">Click Me!</button>
```

JavaScript fundamentals

What can JavaScript do?

JavaScript Can Show HTML Elements

Showing hidden HTML elements can also be done by changing the display style

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can show hidden HTML elements.</p>

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button"
onclick="document.getElementById('demo').style.display='block'">Click Me!</button>

</body>
</html>
```

JavaScript fundamentals

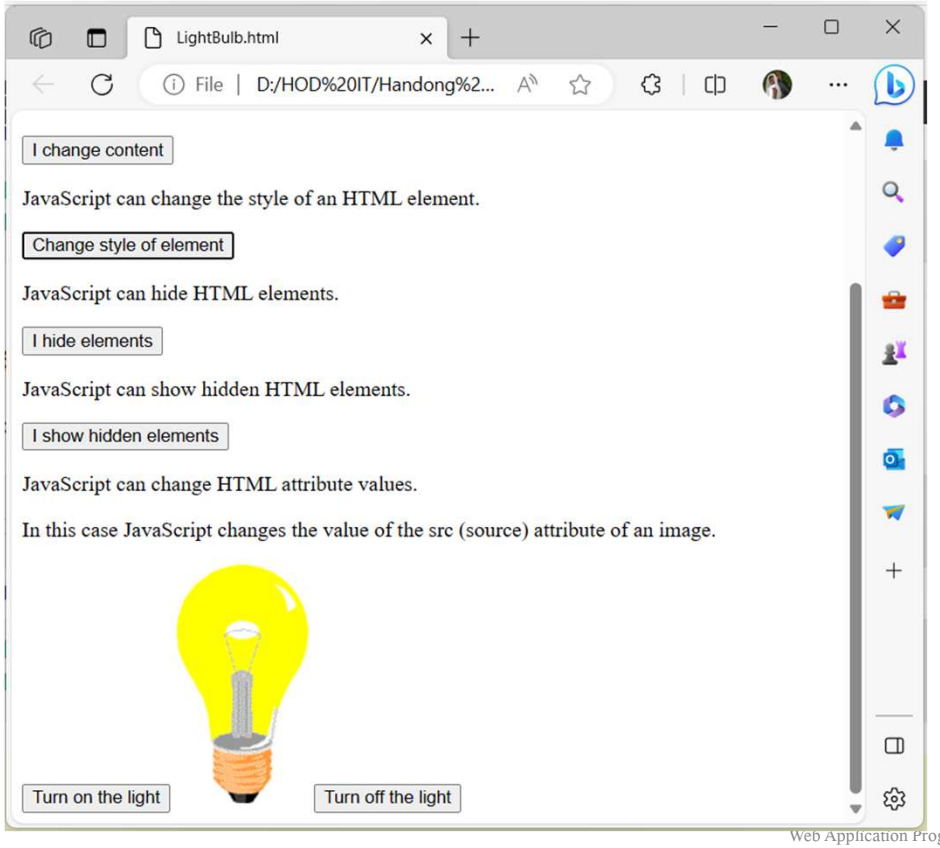
What can JavaScript do?

If you entered you the above code on one page, you should be seeing a page like below

```
<!DOCTYPE html><html><body>

<h1>What Can JavaScript Do?</h1>

<p id="demo">JavaScript can change HTML content.</p>
<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>
I change content</button>

<p id="demo">JavaScript can change the style of an HTML element.</p>
<button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">
Change style of element</button>

<p id="demo">JavaScript can hide HTML elements.</p>
<button type="button" onclick="document.getElementById('demo').style.display='none'">I hide elements</button>

<p>JavaScript can show hidden HTML elements.</p>
<p id="demo" style="display:none">Hello JavaScript!</p>
<button type="button" onclick="document.getElementById('demo').style.display='block'">
I show hidden elements</button>

<p>JavaScript can change HTML attribute values.</p>
<p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>
<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="pic_bulboff.gif" style="width:100px">
<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the light</button>

</body></html>
```

JavaScript fundamentals

What can JavaScript do?



The final out put of you code should look like the page above

JavaScript fundamentals

Did You Know?

Did You Know?

1. JavaScript and Java are completely different languages, both in concept and design.
2. JavaScript was invented by **Brendan Eich** in 1995, and became an ECMA-European Computer Manufacturers Association standard in 1997.
3. ECMA-262 is the official name of the standard. ECMAScript is the official name of the language.

JavaScript fundamentals

Did You Know?

4. ECMA, which stands for "European Computer Manufacturers Association," is an international standards organization that plays a crucial role in the development and standardization of technologies, particularly in the realm of information and communication technologies. One of the most notable contributions of ECMA is its involvement in the standardization of the JavaScript language

5. ECMA Script (ECMAScript): ECMAScript is a standardized scripting language specification that serves as the foundation for several scripting languages, with JavaScript being the most prominent. ECMAScript specifies the syntax, semantics, and core features of scripting languages. JavaScript is often referred to as ECMAScript when discussing its standardized version. (International , *ECMA-262* 2023)

JavaScript fundamentals

Did You Know?

7. **JavaScript Standardization:** JavaScript was originally developed by Netscape, but to ensure its widespread adoption and interoperability across different web browsers, it needed to be standardized. ECMA International played a crucial role in this process by forming a committee to create a standardized specification for JavaScript, which became known as ECMAScript. The first edition of the ECMAScript standard was published in 1997.

8. **ECMA-262:** The ECMAScript standard is formally defined in the ECMA-262 specification. This specification outlines the syntax, semantics, and various components of the language, including objects, functions, and the Document Object Model (DOM).

9. **Updates and Versions:** Over the years, ECMAScript has gone through several versions and updates, each introducing new features and improvements to the language. Notable versions include ECMAScript 3, ECMAScript 5, ECMAScript 6 (also known as ES2015), and subsequent yearly updates (ES2016, ES2017, etc.). These updates have significantly enhanced the capabilities of JavaScript. (International , *ECMA-262* 2023)

JavaScript Identifiers

Like any other language, JS Identifiers are unique names given to variables, costs and functions. All JavaScript variables must be identified with unique names. The following are the rules governing the naming of identifiers.

1. Names can contain letters, digits, underscores, and dollar signs.
2. Names must begin with a letter.
3. Names can also begin with $ and _ (but we will not use it in this tutorial).
4. Names are case sensitive (y and Y are different variables).
5. Reserved words (like JavaScript keywords) cannot be used as names.
6. JavaScript identifiers are case-sensitive.

JavaScript Variables

JavaScript Variables

Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

1. Automatically
2. Using var
3. Using let
4. Using const

Declaring a JavaScript Variable

Creating a variable in JavaScript is called "declaring" a variable.

You declare a JavaScript variable with the var or the let keyword:In this first example, **x, y,** and **z** are defined variables.

They are automatically declared when first used:

Note! It is considered good programming practice to always declare variables before use.

JavaScript Variables

```
var carName;
```

or:

```
let carName;
```

Example

```
x = 5;
y = 6;
z = x + y;
```

In this first example, **x, y,** and **z** are defined variables.

They are automatically declared when first used:

Note! It is considered good programming practice to always declare variables before use.

Re-Declaring JavaScript Variables

If you re-declare a JavaScript variable declared with var, it will not lose its value.

The variable carName will still have the value "Volvo" after the execution of these statements:

Example

```
var carName = "Volvo";
var carName;
```

Note

You cannot re-declare a variable declared with let or const.

This will not work

```
let carName = "Volvo";
let carName;
```

JavaScript Variables+

Example using var

```
var x = 5;
var y = 6;
var z = x + y;
```

Note

The **var** keyword was used in all JavaScript code from 1995 to 2015.

The **let** and **const** keywords were added to JavaScript in 2015.

The **var** keyword should only be used in code written for older browsers.

Example using let

```
let x = 5;
let y = 6;
let z = x + y;
```

Example using const

```
const x = 5;
const y = 6;
const z = x + y;
```

JavaScript Variables+

Mixed Example

```
const price1 = 5;
const price2 = 6;
let total = price1 + price2;
```

Note! The two variables **price1** and **price2** are defined with the const keyword.

These are constant values and cannot be changed.

The variable total is declared with the **let** keyword.

The value total can be changed.

When to Use var, let, or const?

1. Always declare variables
2. Always use const if the value should not be changed
3. Always use const if the type should not be changed (Arrays and Objects)
4. Only use let if you can't use const
5. Only use var if you MUST support old browsers.

JavaScript Arithmetic

As with algebra, you can do arithmetic with JavaScript variables, using operators like = and +

Example

```
let x = 5 + 2 + 3;
```

You can also add strings, but strings will be concatenated:

Example

```
let x = "John" + " " + "Doe";
```

Note

If you put a number in quotes, the rest of the numbers will be treated as strings, and concatenated.

Declaring variable using Dollar Sign $

Since JavaScript treats a dollar sign as a letter, identifiers containing $ are valid variable names:

Example

```
let $ = "Hello World";
let $$$ = 2;
let $myMoney = 5;
```

Using the dollar sign is not very common in JavaScript, but professional programmers often use it as an alias for the main function in a JavaScript library.

In the JavaScript library jQuery, for instance, the main function $ is used to select HTML elements. In jQuery $("p"); means "select all p elements".

Variable Declation using Underscore (_)

Since JavaScript treats underscore as a letter, identifiers containing _ are valid variable names

Example

```
let _lastName = "Johnson";
let _x = 2;
let _100 = 5;
```

JavaScript Data types

JavaScript Data types

JavaScript has several data types that are categorized into two main categories: primitive data types and reference data types.

Primitive Data Types:

Number,

String,

Boolean,

Undefined,

Null,

Symbol (ES6) and

BigInt (ES11)

JavaScript Data types

JavaScript has several data types that are categorized into two main categories: primitive data types and reference data types.

Primitive Data Types:

1. **Number:** Represents both integer and floating-point numbers. Example: let num = 42;
2. **String:** Represents a sequence of characters enclosed in single or double quotes. Example: **let text = "Hello, World!";**
3. **Boolean:** Represents a binary value, either true or false. Example: let isTrue = true;
4. **Undefined:** Represents a variable that has been declared but has not been assigned a value. Example: let undefinedVar;

JavaScript Data types

JavaScript has several data types that are categorized into two main categories: primitive data types and reference data types. (OpenAI. 2021).

Primitive Data Types:

5. **Null:** Represents an intentional absence of any object or value. Example: let emptyValue = null;
6. **Symbol (ES6):** Represents a unique and immutable value primarily used as object property keys. Example: const uniqueSymbol = Symbol("description");
7. **BigInt (ES11):** Represents large integers beyond the range of the Number type. Example: const bigIntValue = 1234567890123456789012345678901234567890n;

JavaScript Data types

Reference Data Types:

also known as objects, are data types that are not directly stored in a variable but are instead stored as references to their memory locations. They include;

1. **Object:** are one of the most versatile reference data types in JavaScript. They consist of key-value pairs, where the keys are strings (or Symbols) and the values can be of any data type, including other objects. Objects are used to represent complex data structures and are the basis for creating custom data types and classes.

```
const person = {
  name: "John",
  age: 30,
};
```

Sample.html

JavaScript Data types

2. **Array:** are specialized objects that store an ordered collection of values. Each value in an array is identified by its index, starting from zero. Arrays are commonly used for lists of items or collections of related data.

```
const fruits = ["apple", "banana", "cherry"];
```

3. **Function:** Functions are also objects in JavaScript. They can be assigned to variables, passed as arguments to other functions, and returned from functions. Functions are essential for defining behavior and performing actions in JavaScript programs.

```
function add(a, b) {
  return a + b;
}
```

JavaScript Data types

4. Date: Date objects are used for working with dates and times. They provide methods for manipulating and formatting dates and times.

```
const currentDate = new Date();
```

5. RegExp (Regular Expression): Regular expressions are objects used for pattern matching within strings. They are powerful tools for searching and manipulating text based on specific patterns.

```
const pattern = /[0-9]+/;
```

JavaScript Data types

7 **Map**: Maps are collections of key-value pairs where keys can be of any data type, including objects.

```
const myMap = new Map();
myMap.set("name", "Alice");
myMap.set("age", 25);
```

8. **Sets** are collections of unique values.

```
const mySet = new Set();
mySet.add(1);
mySet.add(2);
mySet.add(2);
```

JavaScript Data types

9. Symbol: Symbols are unique and immutable values, often used as object property keys.

```
const mySymbol = Symbol("description");
const obj = {
  [mySymbol]: "This is a symbol",
};
```

Reference data types are distinct from primitive data types (such as numbers, strings, and Booleans) in how they are stored and manipulated in memory. Reference data types are mutable (their values can be changed after they are created), whereas primitive data types are immutable. Understanding reference data types is crucial for effective JavaScript programming, as they allow for the creation of complex data structures and objects.

JavaScript Operators

JavaScript Operators

JavaScript operators are symbols or keywords that perform operations on operands, which can be values or variables. Operators enable you to perform various calculations, comparisons, and manipulations in your JavaScript code. Here are some of the most commonly used JavaScript operators:

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Ternary (Conditional) Operator
6. String Concatenation Operator
7. Typeof Operator
8. Other Operators

JavaScript Operators

1. **Arithmetic Operators**

+: Addition

-: Subtraction

*: Multiplication

/: Division

%: Modulus (remainder)

++: Increment

--: Decrement

```
let a = 5;
let b = 3;
let sum = a + b; // 8
let remainder = a % b; // 2
a++; // Increment a by 1
```

JavaScript Operators

2. **Assignment Operators**

= : Assignment

+= : Add and assign

-=: Subtract and assign

*=: Multiply and assign

/=: Divide and assign

%=: Modulus and assign

```
let x = 10;
x += 5; // x is now 15
x /=5; // x is now 7.5
```

JavaScript Operators

3. **Comparison Operators**

==: Equal to (loose equality)

===: Equal to (strict equality)

!=: Not equal to (loose inequality)

!==: Not equal to (strict inequality)

<: Less than

>: Greater than

<=: Less than or equal to

>=: Greater than or equal to

```
let num1 = 5;
let num2 = "5";
let isEqualLoose = num1 == num2; // true (loose equality)
let isEqualStrict = num1 === num2; // false (strict equality)
```

JavaScript Operators

4. Logical Operators

&&: Logical AND

||: Logical OR

!: Logical NOT

```
let isTrue = true;
let isFalse = false;
let result = isTrue && isFalse; // false (AND)
```

JavaScript Operators

5. Ternary (Conditional) Operator

A shorthand way of writing an if-else statement.

condition ? expr1 : expr2:

```
let age = 18;
let canVote = age >= 18 ? "Yes" : "No";// Yes
```

JavaScript Operators

6. **String Concatenation Operator**

+: Combines (concatenates) strings to variables or other data types.

```
let firstName = "John";
let lastName = "Doe";
let fullName = firstName + " " + lastName; // "John Doe"
```

JavaScript Operators

7. Typeof Operator

typeof: Returns a string indicating the type of a variable or value.

```
let value = 42;
let type = typeof value; // "number"
```

JavaScript Operators

8. Other Operators

- **in**: Checks if an object has a specified property.
- **instanceof**: Checks if an object is an instance of a particular class or constructor function.

```
let person = { name: "Alice" };
let hasName = "name" in person; // true
```

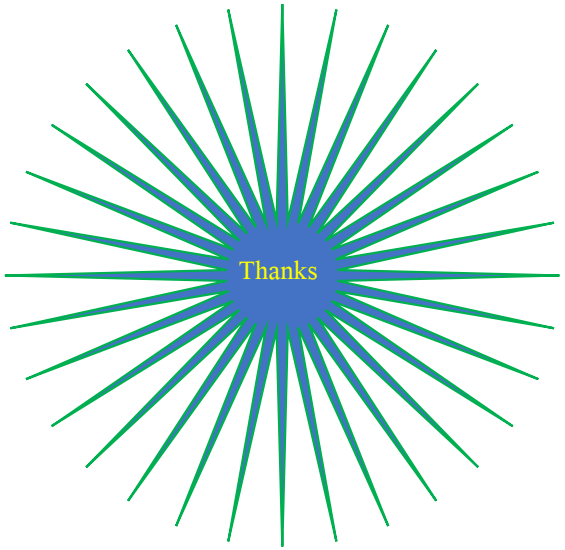
These operators are essential tools for performing various tasks in JavaScript, including mathematical calculations, data comparisons, and logical operations. Understanding how to use them effectively is crucial for writing JavaScript code.

Summary

Summary

1. Summary of previous lecture
2. JavaScript fundamentals
3. variables(looked declarations of varriables; Automatically, Using var, Using let, Using const and initialization of the same),
4. Data types(**Primitive Data Types:** Number, String, Boolean, Undefined, Null, Symbol (ES6) and BigInt (ES11), and **Reference Data Types:** Objects, Date, Arrays, functions etc),
5. Operators(Arithmetic, Assignment, Comparison, Logical Operators etc.), conditional statements, loops etc. and

Thank you for Listening



References

Ecma International , E. (2023). *ECMA-262*. https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

Web app development in 2023 Brewster, C. (2023): Everything you need to know. Trio. https://www.trio.dev/front-end/resources/web-app-development

JavaScript Data Types. OpenAI. (2021). OpenAI Knowledge Base.