

## JavaScript Session 1

### JAVASCRIPT

JavaScript is the scripting language of the Web.

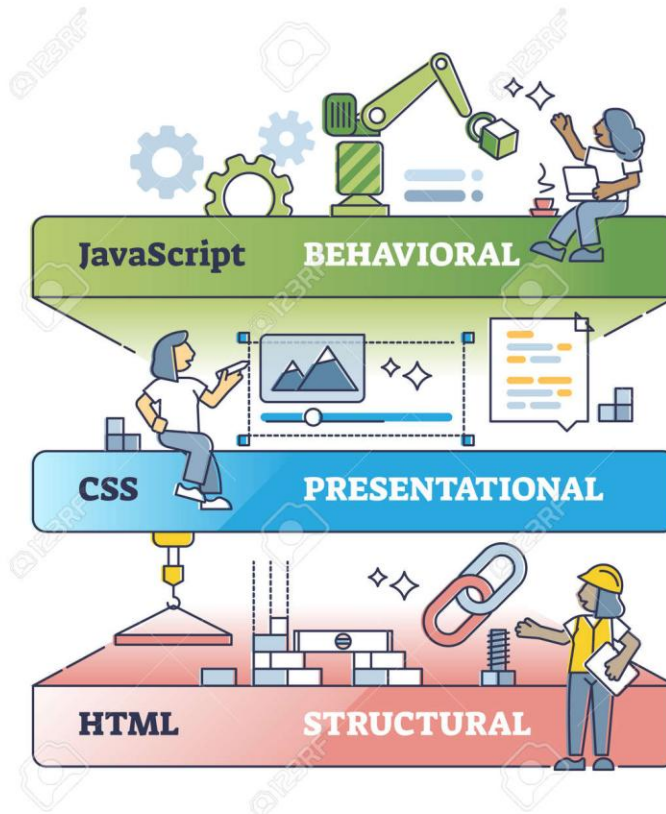
JavaScript is used in millions of Web pages to add functionality, validate forms, detect browsers, and much more.

#### Introduction to JavaScript

JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create

cookies, and much more.

JavaScript is the most popular scripting language on the Internet, and works in all major browsers, such as Internet Explorer, Mozilla Firefox, and Opera.



## What is JavaScript?

1. JavaScript was designed to add interactivity to HTML pages
2. JavaScript is a scripting language
3. A scripting language is a lightweight programming language
4. JavaScript is usually embedded directly into HTML pages
5. JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
6. Everyone can use JavaScript without purchasing a license

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language – in the same category as C and C++.

## JavaScript Syntax

**Theory:** JavaScript syntax dictates the structure and rules for writing valid JavaScript code. Think of it as the grammar of the language.

### Key Points:

- Case-Sensitive: myName is different from MyName.
- Statements: Instructions that perform actions, often terminated by semicolons (; - optional but highly recommended for clarity).
- Code Blocks: Enclosed in curly braces {} to group related statements.
- Comments: // for single-line, /\* ... \*/ for multi-line comments, used to explain code.

```
<script ...>
  JavaScript code
</script>
```

```
<script language = "javascript" type = "text/javascript">
  <!--
    // This is a comment. It is similar to comments in C++

    /*
     * This is a multi-line comment in JavaScript
     * It is very similar to comments in C Programming
     */
  //-->
</script>
```

## Example:

### Working on VS Code Editor

Code:

```
index.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Document</title>
7  </head>
8  <body>
9    <script>
10     console.log("hello world");
11     alert("welcome");
12   </script>
13 </body>
14 </html>
```

(OR)

```
index.html X JS first.js
index.html > html > head > meta
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Document</title>
7  </head>
8  <body>
9    <script src="first.js"></script>
10 </body>
11 </html>
```

```
<> index.html JS first.js X
JS first.js
1 console.log("hello world");
```

## Variables

Variables are used to store data values that can be changed during program execution.



## Variable Naming Rules

- JS has a few rules for naming variables:
  - Only letters, numbers, and the underscore (`_`) are valid characters in a variable name.
  - Variable names may not begin with a number. They must start with a letter or an underscore (`_`).
  - Variable names may not contain spaces.
  - Reserved words may not be used as variable names.
  - Pre-defined object, property, method, and built-in function names are also off limits, for example, `document`, `window`, `form`, `name`, `src`, `isNaN`, etc.

### Example:

```
index.html JS first.js X
JS first.js
1  fullName= "abc";
2  age=12;
3  console.log(fullName);
4  console.log(age);
```

## Variables Types

### let, const & var

var : Variable can be re-declared & updated. A global scope variable.

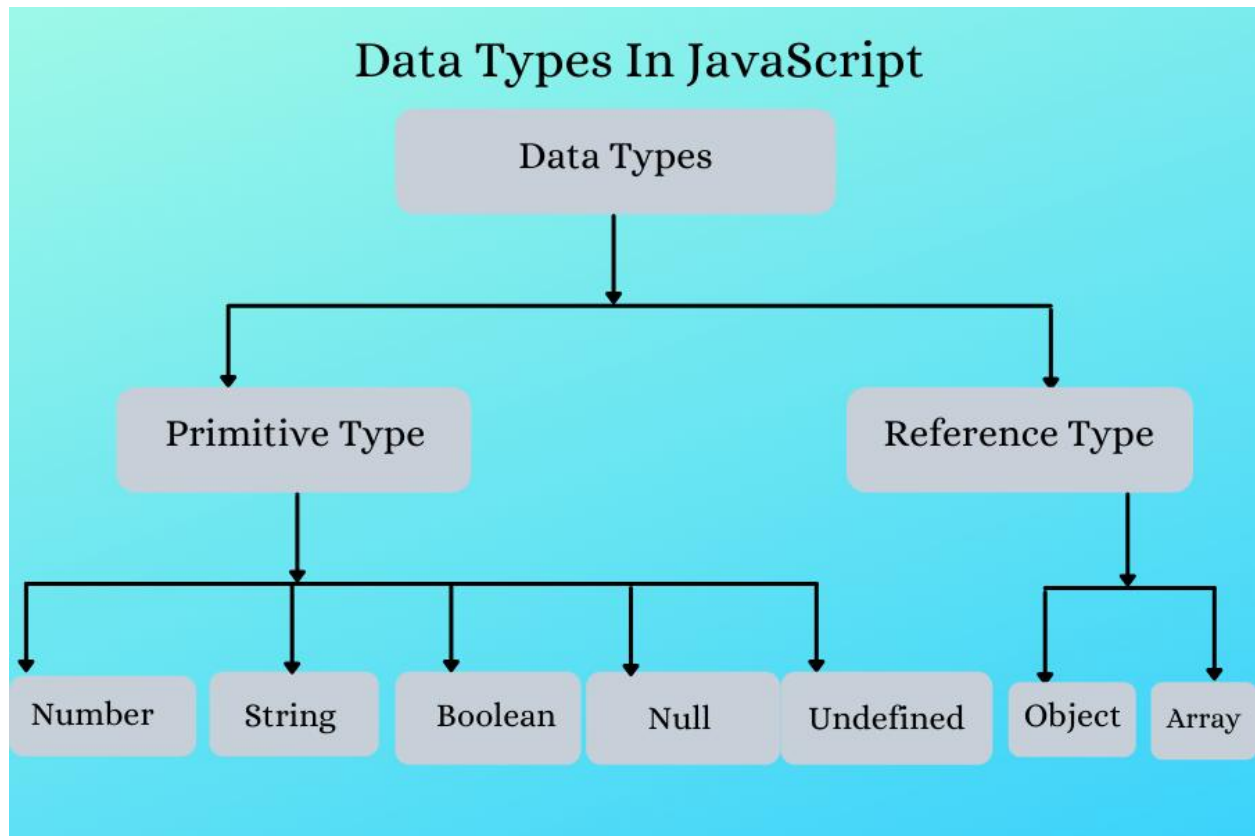
let : Variable cannot be re-declared but can be updated. A block scope variable.

const : Variable cannot be re-declared or updated. A block scope variable.

### Example:

```
index.html JS first.js X
JS first.js > [⌘] isFollow
1  let fullName = "abc";
2  let age =12;
3  let isFollow = true;
4  let a;
5  let b = null;
6  let x = BigInt("123");
7  let y = Symbol("hello");
```

## ***JavaScript Data Types***



## Primitive Data Types

Primitive	
<b>null</b>	let example = null;
<b>undefined</b>	let example = undefined;
<b>Boolean</b>	let example = true;
<b>Number</b>	let example = 33;
<b>String</b>	let example = "hello";
<b>BigInt</b>	let example = 33n;
<b>Symbol</b>	let example = Symbol("hello");

### Example:

```
<> index.html JS first.js X
JS first.js > [?] isFollow
1 let fullName = "abc";
2 let age = 12;
3 let isFollow = true;
4 let a;
5 let b = null;
6 let x = BigInt("123");
7 let y = Symbol("hello");
```

### Output:

```
Elements Console Sources Network >>
top Filter Default levels No Issues
> fullName
< 'abc'
> typeof fullName
< 'string'
> age
< 12
> typeof age
< 'number'
> isFollow
```

```
> isFollow
< true
> typeof isFollow
< 'boolean'
> a
< undefined
> typeof a
< 'undefined'
> b
< null
> typeof b
< 'object'
> x
< 123n
> typeof x
< 'bigint'
> y
< Symbol(hello)
> typeof y
< 'symbol'
>
```

**Non- Primitive(Reference Type) Data Type**



Reference	
Object	let example = {hello: "world"};
Function	let example = () => 2 + 2;

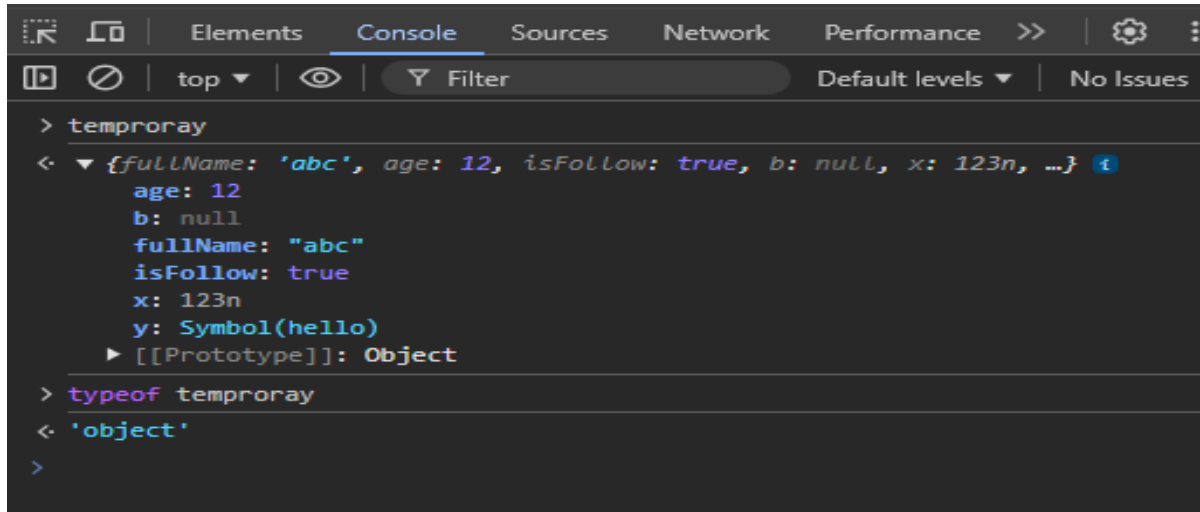
### Example:

```
<> index.html JS first.js X
JS first.js > [0] temproray
1  const temproray = {
2    fullName: "abc",
3    age: 12,
4    isFollow: true,
5    b: null,
6    x: BigInt("123"),
7    y: Symbol("hello")
8  }
```

JS first.js > ...

```
1  const temproray = {  
2    fullName: "abc",  
3    age: 12,  
4    isFollow: true,  
5    b: null,  
6    x: BigInt("123"),  
7    y: Symbol("hello")  
8  }  
9  
10 temproray["age"] = temproray["age"] + 1;  
11 console.log(temproray["age"]);  
12 console.log(temproray.isFollow);
```

## Output:



```
> temporaray
< {fullName: 'abc', age: 12, isFollow: true, b: null, x: 123n, ...}
  age: 12
  b: null
  fullName: "abc"
  isFollow: true
  x: 123n
  y: Symbol(hello)
  [[Prototype]]: Object
> typeof temporaray
< 'object'
>
```

## Operators:

Operators perform operations on values.

### Arithmetic Operators

- + Addition
- - Subtraction
- \* Multiplication
- / Division
- % Modulus
- \*\* Exponentiation

### Example:

```
JS first.js > ...
1  //Airthmetic Operators
2  let a= 5;
3  let b=2;
4  console.log("a = ",a, "b = ", b);
5  console.log("a + b = ",a + b );
6  console.log("a - b = ", a - b);
7  console.log("a * b = ", a * b);
8  console.log("a / b = ", a / b);
9  console.log("a % b = ", a % b);
10 console.log("a ** b = ", a ** b);
```

## Unary Operators

- ++ Increment
- -- Decrement

### Example:

```
//Airthmetic Operators
let a= 5;
let b=2;
console.log("a = ",a, "b = ", b);
console.log("a + b = ",a + b );
console.log("a - b = ", a - b);
console.log("a * b = ", a * b);
console.log("a / b = ", a / b);
console.log("a % b = ", a % b);
console.log("a ** b = ", a ** b);

//unary operator
a++;
b--;
console.log("a++ = ", a );
console.log("b-- = ", b);
```

## ***Conditional Statements***

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

**In JavaScript we have the following conditional statements:**

- if statement - use this statement if you want to execute some code only if a specified condition is true
- if...else statement - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- if...else if....else statement - use this statement if you want to select one of many blocks of code to be executed
- switch statement - use this statement if you want to select one of many blocks of code to be executed

### **If Statement**

You should use the if statement if you want to execute some code only if a specified condition is true.

#### **Syntax**

```
if (condition)
{
code to be executed if condition is true
}
```

Note that if is written in lowercase letters

### Example:

```
JS first.js > ...
1  let age = 18;
2  let eligible;
3  if(age>=18){
4      eligible = "yes eligible";
5  }
6  if(age<18){
7      eligible = "not eligible";
8  }
9  console.log("Eligible or not?",eligible);
```

### **If...else Statement**

If you want to execute some code if a condition is true and another code if the condition is not true, use the if....else statement.

#### **Syntax**

```
if (condition)
{
code to be executed if condition is true
}
else
{
code to be executed if condition is not true
}
```

### Example:

```
JS first.js > ...
1  let age = 17;
2  let eligible;
3  if(age>=18){
4      eligible = "yes";
5  }
6  else{
7      eligible = "no";
8  }
9  console.log("eligible or not?", eligible);
```

### **If...else if...else Statement**

You should use the if....else if...else statement if you want to select one of many sets of lines to execute.

#### **Syntax**

```
if (condition1)
{
code to be executed if condition1 is true
}
else if (condition2)
{
code to be executed if condition2 is true
}
else
{
code to be executed if condition1 and
condition2 are not true
}
```

### Example:

```
JS first.js > [?] age
1  let age = 18;
2  let vote;
3  if(age<18){
4      vote = "no";
5  }else if(age=18){
6      vote = "yes";
7  }
8  else{
9      vote = "yes";
10 }
11 console.log("vote or not?",vote);
```

## **The JavaScript Switch Statement**

You should use the switch statement if you want to select one of many blocks of code to be executed.

### Syntax

```
switch(n)
{
case 1:
    execute code block 1
    break;
case 2:
    execute code block 2
    break;
default:
    code to be executed if n is
    different from case 1 and 2
}
```



This is how it works: First we have a single expression  $n$  (most often a variable), that is evaluated once.

The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use `break` to prevent the code from running into the next case automatically.

#### Example:

```
JS first.js > [?] color
1  const color = "white";
2  switch(color){
3      case 'blue':
4          console.log("color is blue");
5          break;
6      case 'white':
7          console.log("color is white");
8          break;
9      default:
10         console.log("sorry color not matched");
11 }
```