ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Web Programming

Lec 3: JavaScript

References

- PHP 8 Basics, 2020, Springer
  - https://link.springer.com/chapter/10.1007/978-1-4842-8082-9_2
- The Absolute Beginner's Guide to HTML and CSS, 2023, Springer
  - https://link.springer.com/chapter/10.1007/978-1-4842-9250-1_7
- W3C Tutorial
  - https://www.w3schools.com/php
  - https://www.w3schools.com/html
  - https://www.w3schools.com/js
- Additional Topics
  - JQuery: https://www.w3schools.com/jquery
  - Bootstrap 5.0: https://www.w3schools.com/bootstrap5
  - Laravel/Blade Framework 11.0: https://www.w3schools.in/laravel

JavaScript

Why Study JavaScript?

JavaScript is one of the **3 languages** all web developers **must** learn:

1. **HTML** to define the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages

https://www.youtube.com/playlist?list=PL8q8h6vqfkSVRNnlbUk-O9JJ0c9B7mqCp

JavaScript

JavaScript Can Change HTML Content

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

<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>

</body>
</html>
```

JavaScript

JavaScript Can Change HTML Content

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

<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>

</body>
</html>
```

What Can JavaScript Do?

JavaScript can change HTML content.

Click Me!

What Can JavaScript Do?

Hello JavaScript!

Click Me!

JavaScript

JavaScript Can Change HTML Content

JavaScript in <body>

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

<h2>JavaScript in Body</h2>

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

<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>

</body>
</html>
```

JavaScript in Body

My First JavaScript

JavaScript

JavaScript Can Change HTML Content

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body><h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

JavaScript

JavaScript Can Change HTML Content

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

<h2>Demo JavaScript in Body</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

Demo JavaScript in Body

A Paragraph.

Try it

Demo JavaScript in Body

Paragraph changed.

Try it

JavaScript

JavaScript Can Change HTML Content
External JavaScript

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

<h2>Demo External JavaScript</h2>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p>This example links to "myScript.js".</p>
<p>(myFunction is stored in "myScript.js")</p>

<script src="myScript.js"></script>

</body>
</html>
```

Demo External JavaScript

A Paragraph.

Try it

This example links to "myScript.js".

(myFunction is stored in "myScript.js")

JavaScript

JavaScript Can Change HTML Content

External JavaScript Advantages

Placing scripts in external files has some advantages:
•It separates HTML and code
•It makes HTML and JavaScript easier to read and maintain
•Cached JavaScript files can speed up page loads

To add several script files to one page - use several script tags:

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

```
<script src="https://www.w3schools.com/js/myScript.js"></script>
```

JavaScript Output

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using `innerHTML`.
- Writing into the HTML output using `document.write()`.
- Writing into an alert box, using `window.alert()`.
- Writing into the browser console, using `console.log()`.

JavaScript

Using innerHTML

To access an HTML element, JavaScript can use the `document.getElementById(id)` method.
The `id` attribute defines the HTML element. The `innerHTML` property defines the HTML content:

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

<h2>My First Web Page</h2>
<p>My First Paragraph.</p>

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

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

My First Web Page

My First Paragraph.

11

JavaScript

Using document.write()

For testing purposes, it is convenient to use `document.write()`:

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

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<p>Never call document.write after the document has finished loading.
It will overwrite the whole document.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

My First Web Page

My first paragraph.

Never call document.write after the document has finished loading. It will overwrite the whole document.

11

JavaScript

Using document.write() after an HTML document is loaded, will **delete all existing HTML**:

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

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">Try it</button>

</body>
</html>
```

My First Web Page

My first paragraph.

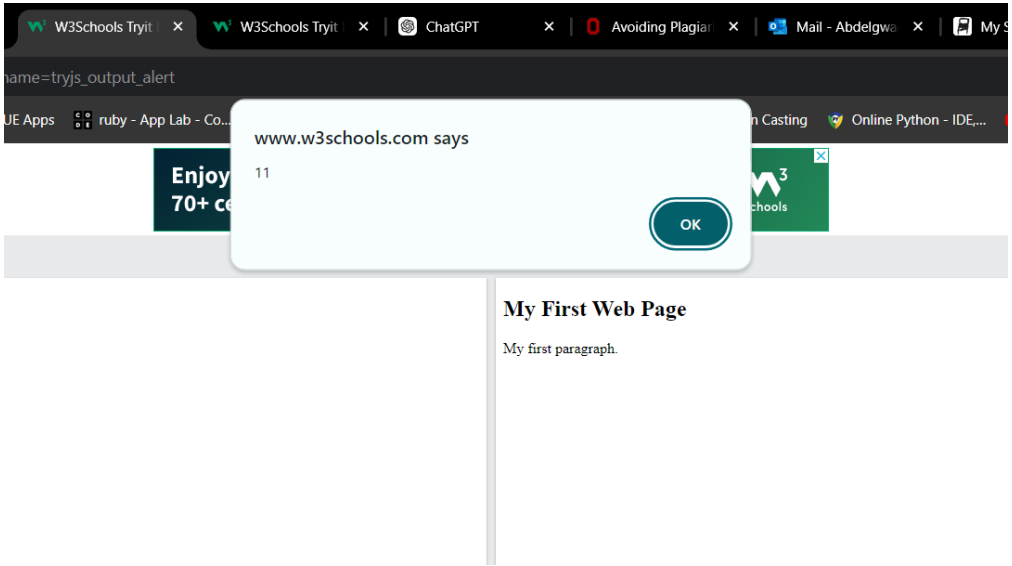
Try it

11

JavaScript

Using window.alert()

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

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```



JavaScript

Using console.log()

For debugging purposes, you can call the `console.log()` method in the browser to display data.

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

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

JavaScript

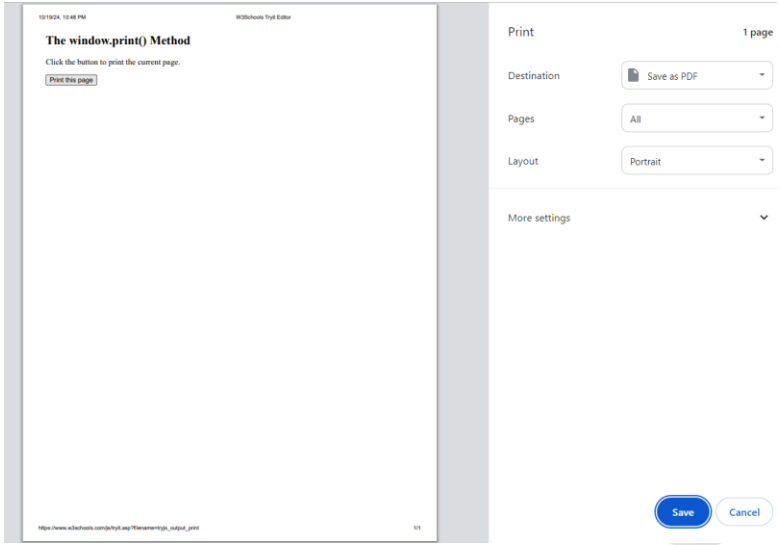
JavaScript Print

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

<h2>The window.print() Method</h2>

<p>Click the button to print the current page.</p>

<button onclick="window.print()">Print this page</button>

</body>
</html>
```

The window.print() Method

Click the button to print the current page.

Print this page



JavaScript

JavaScript Statements

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

<h2>JavaScript Statements</h2>

<p>A <b>JavaScript program</b> is a list of <b>statements</b> to be executed by a computer.</p>

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

<script>

let x, y, z;  // Statement 1
x = 5;        // Statement 2
y = 6;        // Statement 3
z = x + y;    // Statement 4

document.getElementById("demo").innerHTML =
"The value of z is " + z + ".";
</script>

</body>
</html>
```

JavaScript

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

<h2>JavaScript Statements</h2>

<p>A <b>JavaScript program</b> is a list of <b>statements</b> to be executed by a computer.</p>

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

<script>

let x, y, z;  // Statement 1
x = 5;        // Statement 2
y = 6;        // Statement 3
z = x + y;    // Statement 4

document.getElementById("demo").innerHTML =
"The value of z is " + z + ".";
</script>

</body>
</html>
```

JavaScript Statements

A **JavaScript program** is a list of **statements** to be executed by a computer.

The value of z is 11.

JavaScript

JavaScript Syntax

```
// How to create variables:
var x;
let y;

// How to use variables:
x = 5;
y = 6;
let z = x + y;
```

JavaScript

JavaScript Values

The JavaScript syntax defines two types of values:

•Fixed values
•Variable values

Fixed values are called **Literals**.

```
<script>
document.getElementById("demo").innerHTML = 10.50;
</script>
```

```
<script>
document.getElementById("demo").innerHTML = 'John Doe';
</script>
```

Variable values are called **Variables**

```
<script>
let x;
x = 6;
document.getElementById("demo").innerHTML = x;
</script>
```

JavaScript

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

<h2>JavaScript Expressions</h2>

<p>Expressions compute to values.</p>

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

<script>
document.getElementById("demo").innerHTML = "John" + " "  + "Doe";
</script>

</body>
</html>
```

JavaScript Expressions

Expressions compute to values.

John Doe

JavaScript

JavaScript Keywords

The `let` keyword tells the browser to create variables:

```
let x, y;
x = 5 + 6;
y = x * 10;
```

The `var` keyword also tells the browser to create variables:

```
var x, y;
x = 5 + 6;
y = x * 10;
```

JavaScript

JavaScript Comments

Not all JavaScript statements are "executed".
Code after double slashes // or between /* and */ is treated as a **comment**.
Comments are ignored, and will not be executed:

```
let x = 5;   // I will be executed

// x = 6;   I will NOT be executed
```

JavaScript

JavaScript Identifiers / Names

A JavaScript name must begin with:

- A letter (A-Z or a-z)
- A dollar sign ($)
- Or an underscore (_)

JavaScript is Case Sensitive

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

The variables `lastName` and `lastname`, are two different variables:

JavaScript

JavaScript Variables

Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

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

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

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

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

JavaScript

Mixed Example

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

The two variables `price1` and `price2` are declared 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.

JavaScript

```html
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Variables</h1>

<p>You can declare many variables in one statement.</p>

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

<script>
let person = "John Doe", carName = "Volvo", price = 200;
document.getElementById("demo").innerHTML = carName;
</script>

</body>
</html>
```

JavaScript Variables

You can declare many variables in one statement.

Volvo

JavaScript

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

<h1>JavaScript Variables</h1>

<p>The result of adding "5" + 2 + 3 is:</p>
<p id="demo"></p>

<script>
let x = "5" + 2 + 3;
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

JavaScript Variables

The result of adding "5" + 2 + 3 is:

523

JavaScript

JavaScript Let

Block Scope

Variables declared with `let` have **Block Scope**

Example

Variables declared inside a { } block cannot be accessed from outside the block:

```
{
  let x = 2;
}
// x can NOT be used here
```

JavaScript

Global Scope

Variables declared with the `var` always have **Global Scope**.

Variables declared with the `var` keyword can NOT have block scope:

Example

Variables declared with `var` inside a { } block can be accessed from outside th

```
{
  var x = 2;
}
// x CAN be used here
```

JavaScript

Cannot be Redeclared

Variables defined with `let` **can not** be redeclared.

You can not accidentally redeclare a variable declared with `let`.

With `let` you **can not** do this:

```
let x = "John Doe";

let x = 0;
```

Variables defined with `var` **can** be redeclared.

With `var` you **can** do this:

```
var x = "John Doe";

var x = 0;
```

Redeclaring Variables

Redeclaring a variable using the `var` keyword can impose problems.

Redeclaring a variable inside a block will also redeclare the variable outside the block:

Example

```
var x = 10;
// Here x is 10

{
var x = 2;
// Here x is 2
}

// Here x is 2
```

JavaScript

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

<h2>Redeclaring a Variable Using var</h2>

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

<script>
var  x = 10;
// Here x is 10

{
var x = 2;
// Here x is 2
}

// Here x is 2
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

Redeclaring a Variable Using

2

JavaScript

Redeclaring a variable using the `let` keyword can solve this problem.

Redeclaring a variable inside a block will not redeclare the variable outside the block:

Example

```
let x = 10;
// Here x is 10

{
let x = 2;
// Here x is 2
}

// Here x is 10
```

JavaScript

With `let`, redeclaring a variable in the same block is NOT allowed:

Example

```
var x = 2;   // Allowed
let x = 3;   // Not allowed

{
let x = 2;   // Allowed
let x = 3;   // Not allowed
}

{
let x = 2;   // Allowed
var x = 3;   // Not allowed
}
```

JavaScript

Redeclaring a variable with `let`, in another block, IS allowed:

Example

```
let x = 2;   // Allowed

{
let x = 3;   // Allowed
}

{
let x = 4;    // Allowed
}
```

JavaScript

JavaScript Const

Cannot be Reassigned

A variable defined with the `const` keyword cannot be reassigned:

Example

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

Must be Assigned

JavaScript `const` variables must be assigned a value when they are declared:

Correct

```
const PI = 3.14159265359;
```

Incorrect

```
const PI;
PI = 3.14159265359;
```

JavaScript Const

Constant Objects and Arrays

The keyword `const` is a little misleading.

It does not define a constant value. It defines a constant reference to a value.

Because of this you can NOT:

- Reassign a constant value
- Reassign a constant array
- Reassign a constant object

But you CAN:

- Change the elements of constant array
- Change the properties of constant object

Constant Objects and Arrays

The keyword `const` is a little misleading.

It does not define a constant value. It defines a constant reference to a value.

Because of this you can NOT:

- Reassign a constant value
- Reassign a constant array
- Reassign a constant object

But you CAN:

- Change the elements of constant array
- Change the properties of constant object

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

<h2>JavaScript const</h2>

<p>Declaring a constant array does NOT make the elements unchangeable:</p>

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

<script>
// Create an Array:
const cars = ["Saab", "Volvo", "BMW"];

// Change an element:
cars[0] = "Toyota";

// Add an element:
cars.push("Audi");

// Display the Array:
document.getElementById("demo").innerHTML = cars;
</script>

</body>
</html>
```

JavaScript const

Declaring a constant array does NOT make the elements unchangeable:

Toyota,Volvo,BMW,Audi

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

<h2>JavaScript const</h2>

<p>Declaring a constant object does NOT make the objects properties unchangeable:</p>

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

<script>
// Create an object:
const car = {type:"Fiat", model:"500", color:"white"};

// Change a property:
car.color = "red";

// Add a property:
car.owner = "Johnson";

// Display the property:
document.getElementById("demo").innerHTML = "Car owner is " + car.owner;
</script>

</body>
</html>
```

JavaScript const

Declaring a constant object does NOT make the objects properties unchangeable:

Car owner is Johnson

JavaScript

JavaScript Operators

Javascript operators are used to perform different types of mathematical and logical computations.

Examples:

The **Assignment Operator** = assigns values

The **Addition Operator** + adds values

The **Multiplication Operator** * multiplies values

The **Comparison Operator** > compares values

JavaScript Arithmetic Operators

Arithmetic operators perform arithmetic on numbers (literals or variables).

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

JavaScript

JavaScript Data Types

```
x = 16 + "Volvo";
```

Note”=:
When adding a number and a string, JavaScript will treat the number as a string.

JavaScript has 8 Datatypes

String
Number
Bigint
Boolean
Undefined
Null
Symbol
Object

The Object Datatype

The object data type can contain both **built-in objects**, and **user defined objects**:

Built-in object types can be:

objects, arrays, dates, maps, sets, intarrays, floatarrays, promises, and more.

Examples

```
// Numbers:
let length = 16;
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
```

JavaScript:

```
let x = 16 + 4 + "Volvo";
```

Result:

```
20Volvo
```

Try it Yourself »

JavaScript:

```
let x = "Volvo" + 16 + 4;
```

Result:

```
Volvo164
```

Try it Yourself »

JavaScript

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example

```
let x;        // Now x is undefined
x = 5;        // Now x is a Number
x = "John";   // Now x is a String
```

JavaScript

JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task.
A JavaScript function is executed when "something" invokes it (calls it).

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Functions</h1>

<p>Call a function which performs a calculation and returns the result:</p>

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

<script>
function myFunction(p1, p2) {
  return p1 * p2;
}

let result = myFunction(4, 3);
document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

JavaScript

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code

Local Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.

Local variables can only be accessed from within the function.

Example

```
// code here can NOT use carName

function myFunction() {
  let carName = "Volvo";
  // code here CAN use carName
}

// code here can NOT use carName
```

JavaScript

JavaScript Objects

Real Life Objects

In real life, **objects** are things like: houses, cars, people, animals, or any other subjects.

Here is a **car object** example:

Car Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()