JS (JavaScript)

JS

- JS Introduction
- JS Where To
- JS Output
- JS Syntax
- JS Variables
- JS Let
- JS Const
- JS Comments
- JS Statements
- JS Data Types
- JS Functions
- JS Objects
- JS Events
- JS Strings
- JS String Methods
- JS String Search
- JS String Templates
- JS Numbers
- JS Number Methods
- JS Conditions
- JS Switch
- JS Loop For
- JS Loop For In
- JS Loop For of
- JS Loop While
- JS Arrays
- JS Array Methods
- JS Array Sort
- JS Array Iteration
- JS Array Const
- JS Dates
- JS Date Formats
- JS Math
- JS Booleans
- JS Comparisons
- JS Type of and Type Conversion
- JS this Keyword & Arrow Function

JS Introduction

- JavaScript is the world's most popular programming language.
- JavaScript is The programming language of the Web.
- JavaScript is easy to learn.

One of many JavaScript HTML methods is getElementById().

E.g.

```
document.getElementById("hello").innerHTML = "Hello JavaScript";
```

Note: JavaScript can change HTML Content, HTML Attribute Values, HTML Styles, Show and Hide of HTML Elements .

JS Where To

In HTML, JavaScript code is inserted between <script> and </script> tags.

A JavaScript function is a block of JavaScript code, that can be executed when “called” for.

For example, a function can be called when an event occurs, like when the use clicks a button.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

Scripts can also be placed in external files.

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

External References

An external script can be referenced in 3 different ways:

- With a full URL (a full web address)
- With a file path (like /js/)
- Without any path

JS Output

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().

E.g.

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

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

E.g.

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

Note: document.write() method should only be used for testing.

E.g.

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

```
<script>
alert(5 + 6);
</script>
```

E.g.

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

F12 on your keybord will activate debugging.

Then select "Console" in the debugger menu.

JS Syntax

The JavaScript syntax defines two types of values :

- Fixed values
- Variable values

Fixed values are called Literals.

Variable values are called Variables.

JavaScript and Camel Case

Historically, programmers have used different ways of joining multiple words into one variable name:

- Hyphens : first-name, last-name (Hyphens are not allowed in JS.)
- Underscore: first_name, last_name
- Upper Camel Case (Pascal Case): FirstName, LastName
- Lower Camel Case : firstName, lastName

JavaScript Literals

The two most important syntax rules for fixed values are

1. Numbers are written with or without decimals.
2. Strings are text, written within double or single quotes.

E.g.

Numbers : 10.50 , 1001

Strings : “John Doe”, ‘John Doe’

JS Variables

There are 3 ways to declare a JavaScript variable:

- Using var
- Using let
- Using const

Variables

Variables are containers for storing data (values).

All JavaScript variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, total).

JS Let

The let keyword was introduced in ES6 (2015).

- Variables defined with let cannot be Redeclared.
- Variables defined with let must be Declared before use.
- Variable defined with let have Block Scope.

E.g.

```
let x = "John Doe";

let x = 0;

// SyntaxError: 'x' has already been declared
```

E.g.

```
//Block Scope

          let x = 10;

{
          let x = 2;

}
```

E.g.

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

ReferenceError : let must declare before use
```

JS Const

The const keyword was introduced in ES6 (2015).

- Variables defined with const cannot be Redeclared.
- Variables defined with const cannot be Reassigned.
- Variables defined with const have Block Scope.

E.g.

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

When to use JavaScript const ?

As a general rule, always declare a variable with const unless you know that the value will change.

Use const when user declare:

- A new Array
- A new Object
- A new Function
- A new RegExp

E.g.

```
const pi = 3.142

pi = 3.14 // Error
```

E.g.

```
var x = 2;       // Allowed
const x = 2;   // Not allowed

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

{
const x = 2;   // Allowed
const x = 2;   // Not allowed
}
```

JS Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable.

JavaScript comments can also be used to prevent execution , when testing alternative code.

Single Line Comments (//)

Multi-Line Comments (/* */)

JS Statements

A computer program is a list of “ instructions “ to be “ executed” by a computer.

In a programming language, these programming instructions are called statements.

Semicolon separate JavaScript statements.
Add a semicolon at the end of each executable statement.

JavaScript ignores multiple spaces. User can add white space to your script to make it more readable.

E.g.

```
let x, y, z;          // statement 1 ( Declare 3 variables )

x = 5;                // statement 2 ( Assign the value 5 to x )
y = 6;                // statement 3 ( Assign the value 6 to y )
z = x + y;            // statement 4 ( Assign the sum of x and y to z )
```

JS Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	**Description**
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division 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 comparison Operators

Operator	**Description**
==	Equal to
===	Equal value and equal type
!=	Not equal
!==	Not equal value or not equal type
>	Greater than
<	Less than
>=	Greater than equal to
<=	Less than or equal to
?	Ternary operator

JavaScript Logical Operators

Operator	**Description**
&&	Logical and
||	Logical or
!	Logical not

JavaScript Type Operators

Operator	**Description**
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

JS Data Types

JavaScript variables can hold different data types: numbers, stings, objects and more.

In Programming, data types is an important concept.
To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve.

E.g.

```
let x                                                   // Undefined
let y = null;                                           // Null
let z = true;                                           // Booleans
let length = 16;                                        // Number
let lastName = “Johnson”;                               // String
let x = { firstName: “John”, lastName : “Doe” };        // Object
```

JS Functions

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ( ).

Function names can contain letters, digits, underscores and dollar signs (same rules as variables).

There have four types of Functions (method)

- no return function (method)
- return function (method)
- parameter function (method)
- return parameter function (method)

no return function (method)

E.g.

```
function myfunction ( ) {

        alert ( " Hello World " ) ;

}
```

return function (method)

E.g.

```
function myfunction ( ) {

        return 1 + 2 ;
}
```

parameter function (method)

E.g.

```
function myfunction (a , b ) {

        alert ( a + b ) ;

}
```

return parameter function (method)

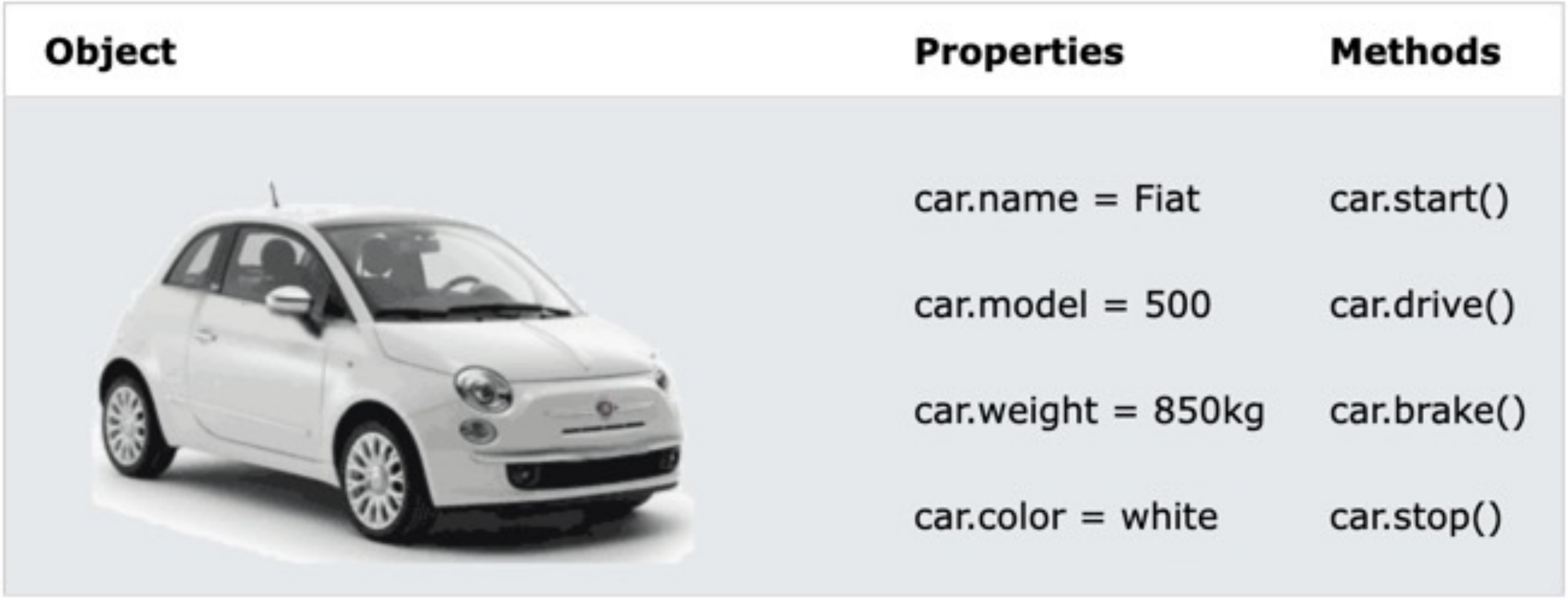
E.g.

```
function myfunction (a , b ) {

        return a + b ;

}
```

JS Objects

In real life, a car is an object.

A car has properties like weight and color , and methods like drive.
All cars have the same properties, but the property values differ from car to car.
All cars have the same methods, but the methods are performed at different times.



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