INSTITUT
TEKNOLOGI
HARAPAN
BANGSA
School of Telematics

Desain Web IF-1P03

JavaScript

Ken Ratri. MT

Introduction

- 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

Introduction

- JavaScript is *THE* scripting language of the Web.
- JavaScript is used in billions of Web pages to add functionality, validate forms, communicate with the server, and much more.
- A scripting language is a lightweight programming language, interpreted programming language
- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is programming code that can be inserted into HTML pages to be executed by the web browser.
- Open and cross-platform

Introduction

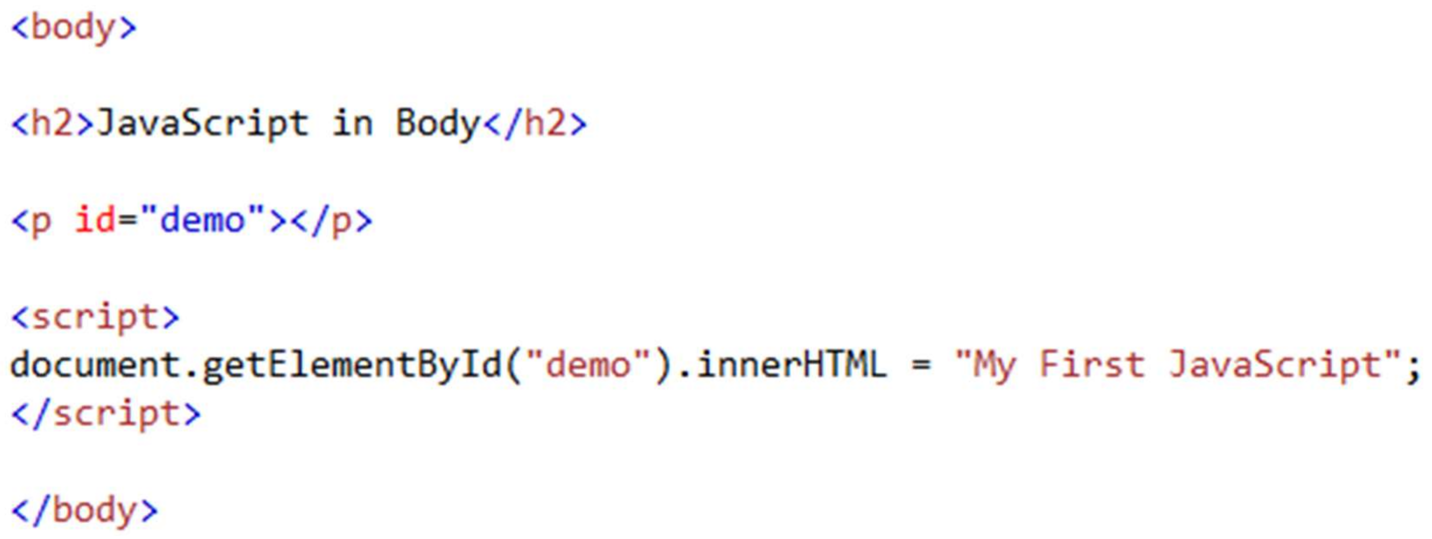
- JavaScript can:
  - You can modify the content of a web page by adding or removing elements.
  - You can change the style and position of the elements on a web page.
  - You can monitor events like mouse click, hover, etc. and react to it.
  - You can perform and control transitions and animations.
  - You can create alert pop-ups to display info or warning messages to the user.
  - You can perform operations based on user inputs and display the results.
  - You can validate user inputs before submitting it to the server.

Adding JavaScript to Your Web Pages

- There are typically three ways to add JavaScript to a web page:
  - Embedding the JavaScript code between a pair of <script> and </script> tag.
  - Creating an external JavaScript file with the .js extension and then load it within the page through the src attribute of the <script> tag.
  - Placing the JavaScript code directly inside an HTML tag using the special tag attributes such as onclick, onmouseover, onkeypress, onload, etc.

How To….

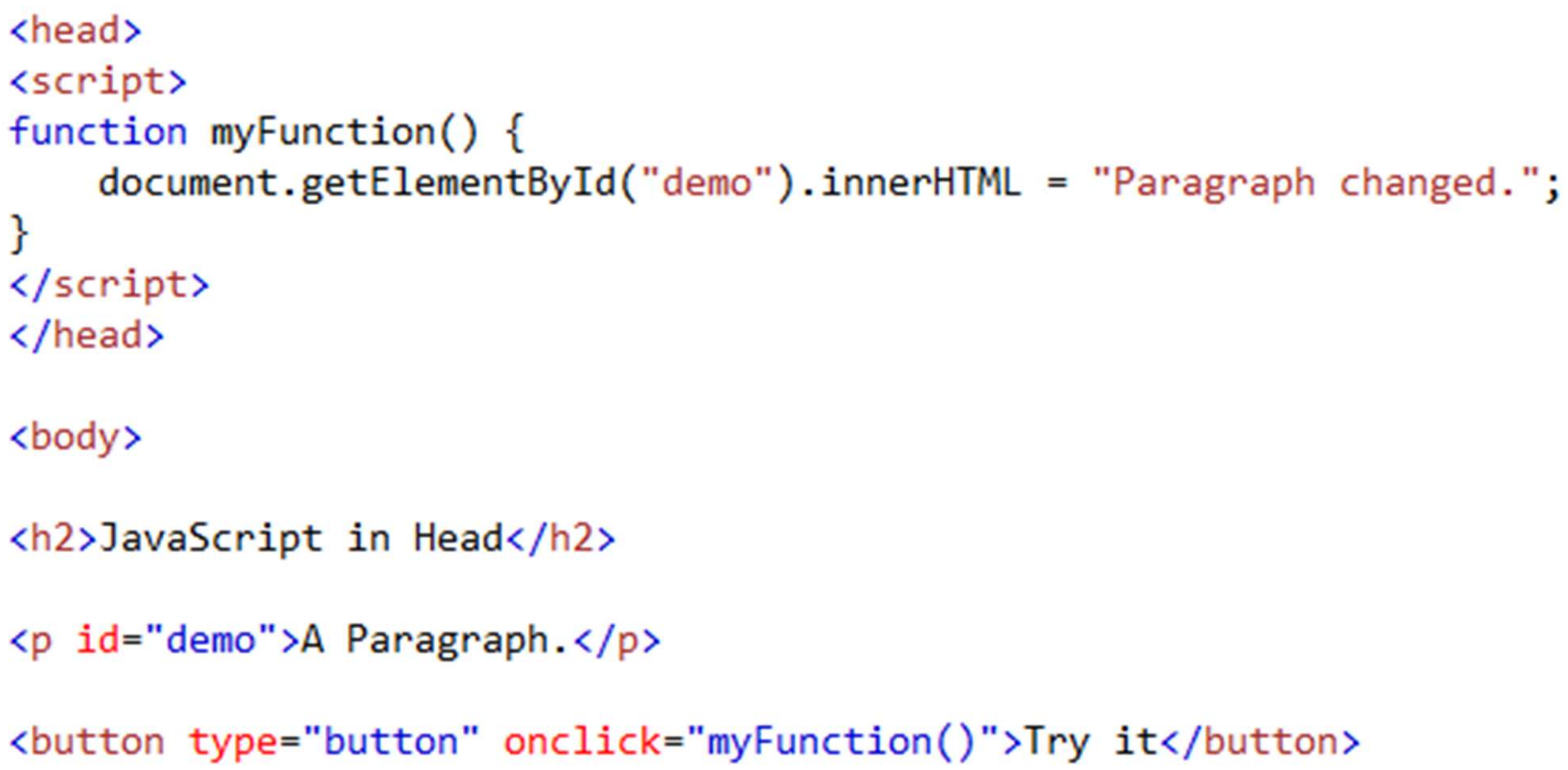
- Scripts in HTML must be inserted between <script> and </script> tags.



```
<body>

<h2>JavaScript in Body</h2>

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

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

</body>
```

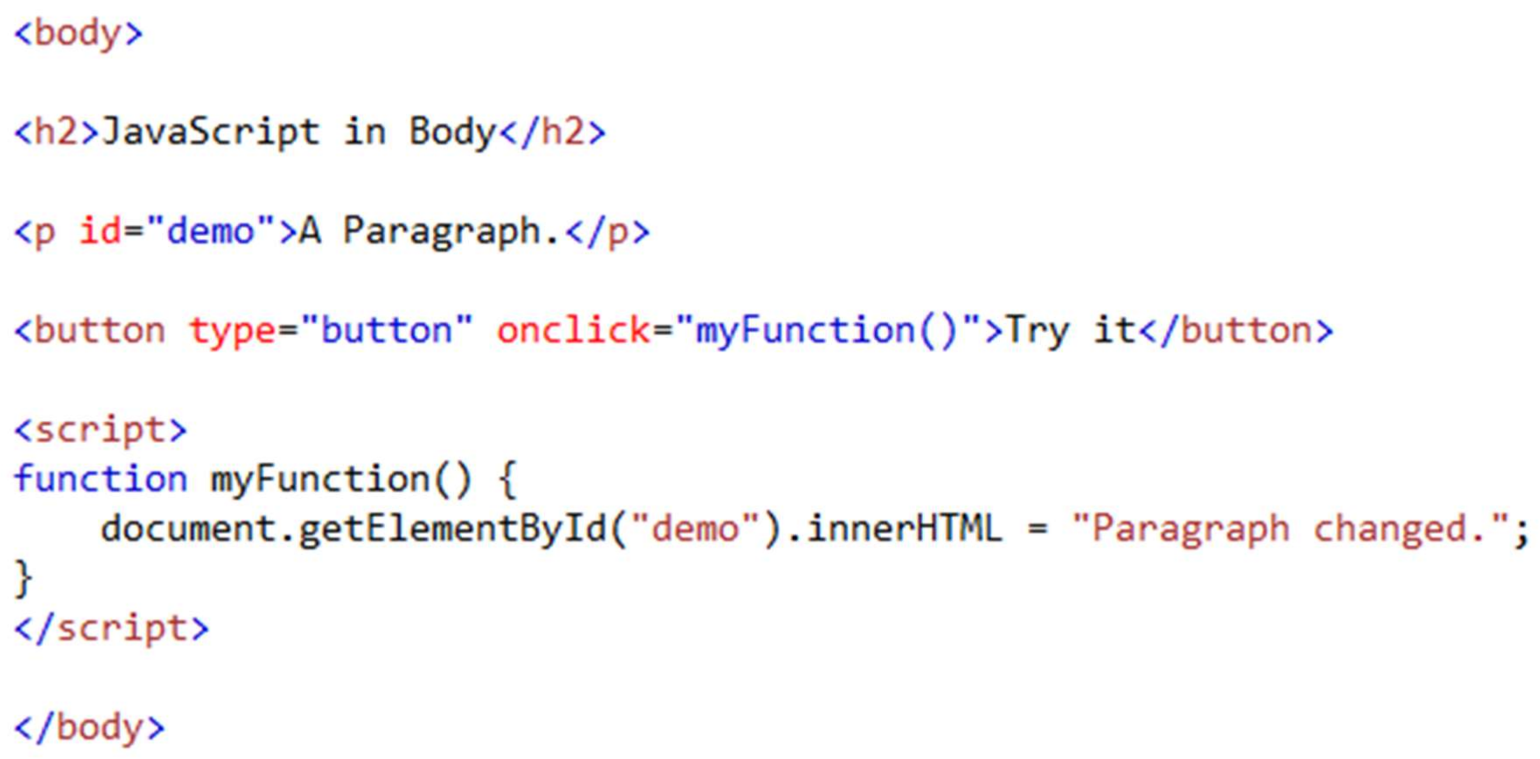
Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

JavaScript in <head>



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

<body>

<h2>JavaScript in Head</h2>

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

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

JavaScript in <body>



```
<body>

<h2>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>
```

External JS Advantages

- Placing JavaScripts 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 JS

```
<html>
<body>

<h2>External JavaScript</h2>

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

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

<p>(myFunction is stored in an external file called "myScript.js")</p>

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

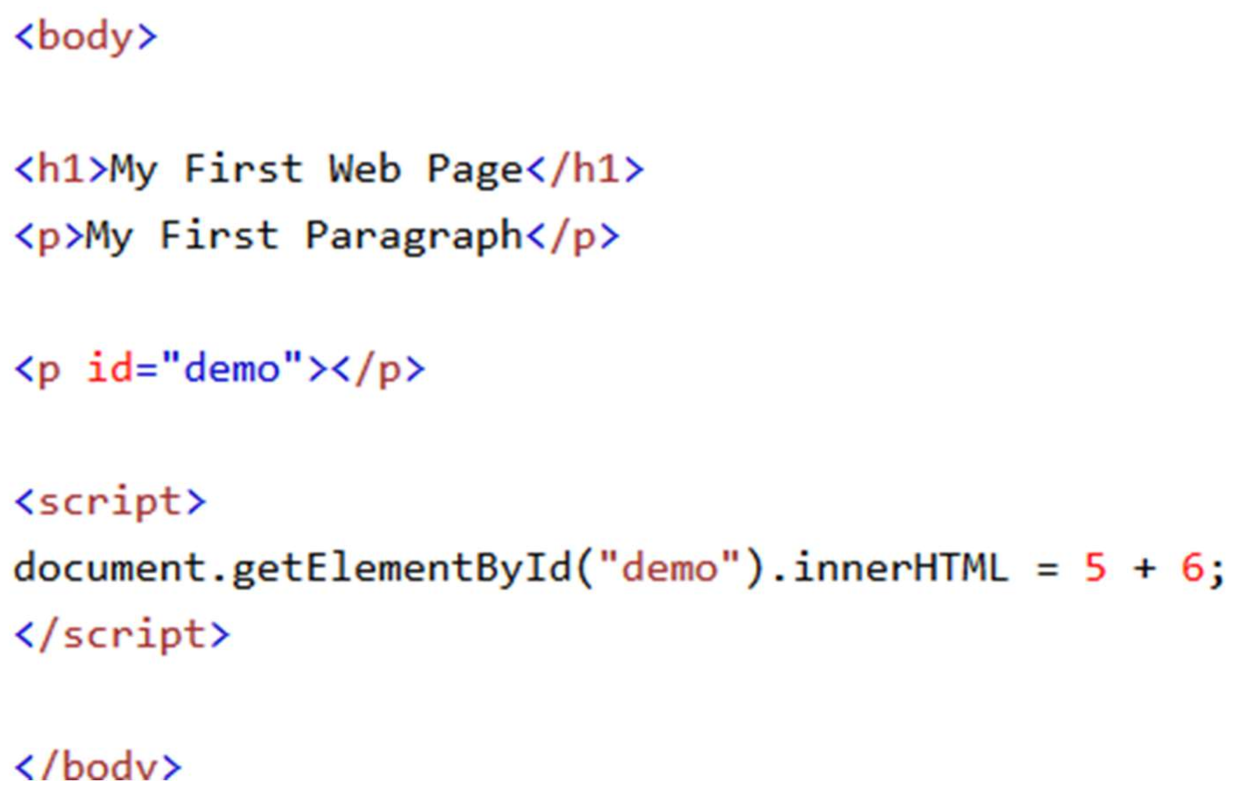
myScript .js

```
function myFunction() {
   document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

JavaScript Output

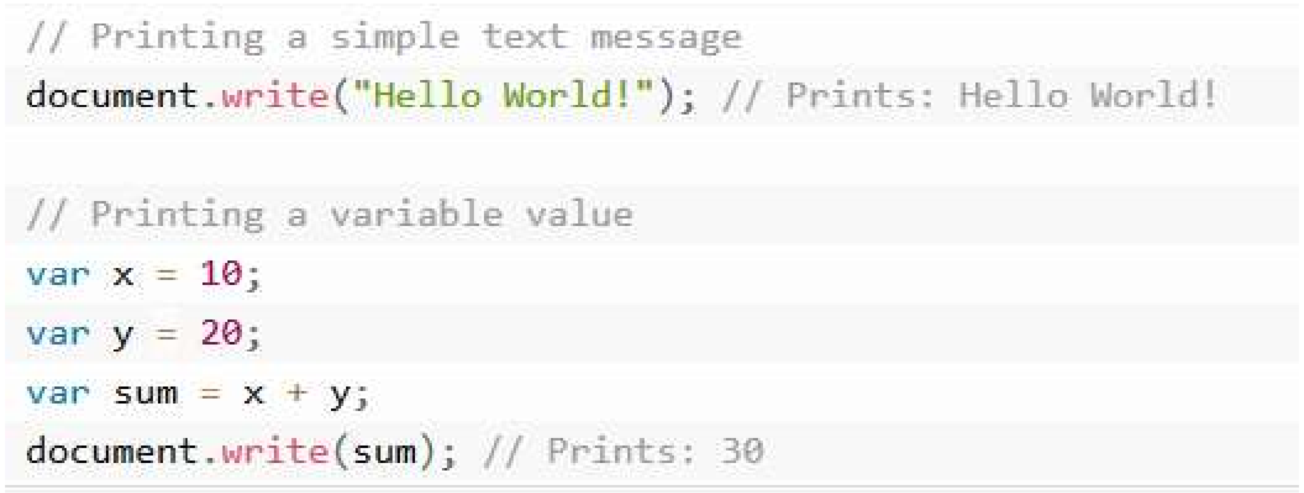
- JavaScript can "display" data in different ways:
  - Writing into an HTML element, using innerHTML.
  - Writing into the HTML output using document.write().
  - Writing into an alert box, using window.alert().
  - Writing into the browser console, using console.log().

innerHTML



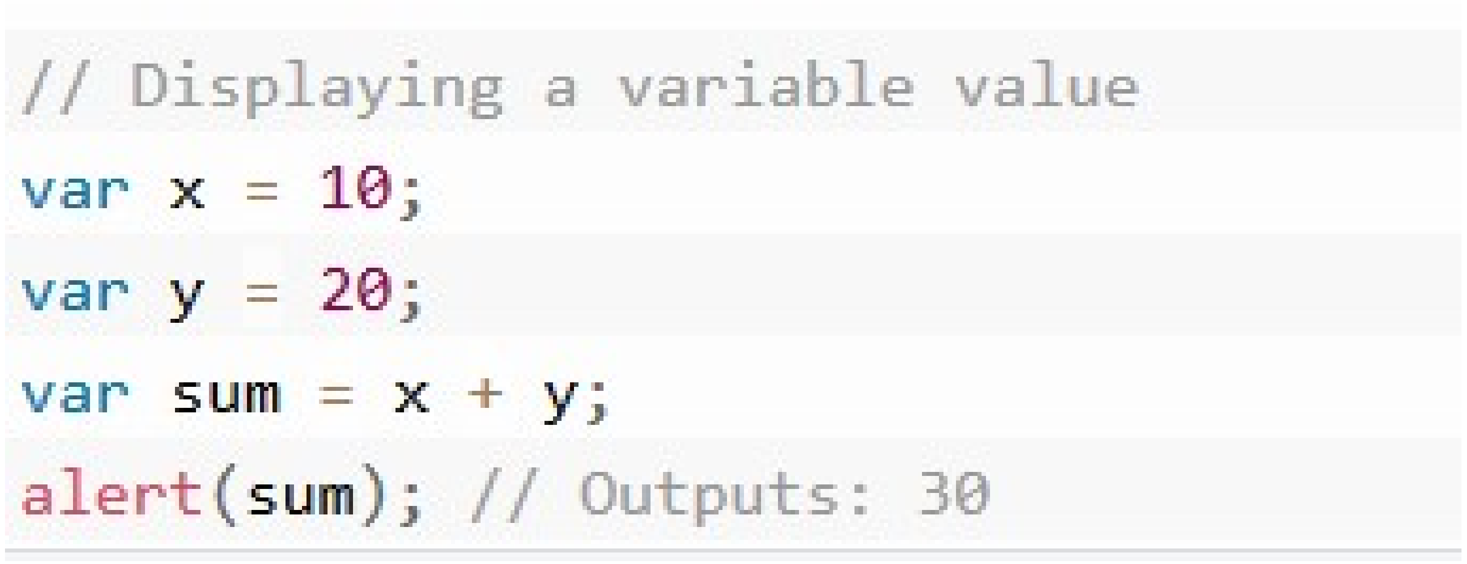
```
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

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

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

</body>
```

using **document.write()**



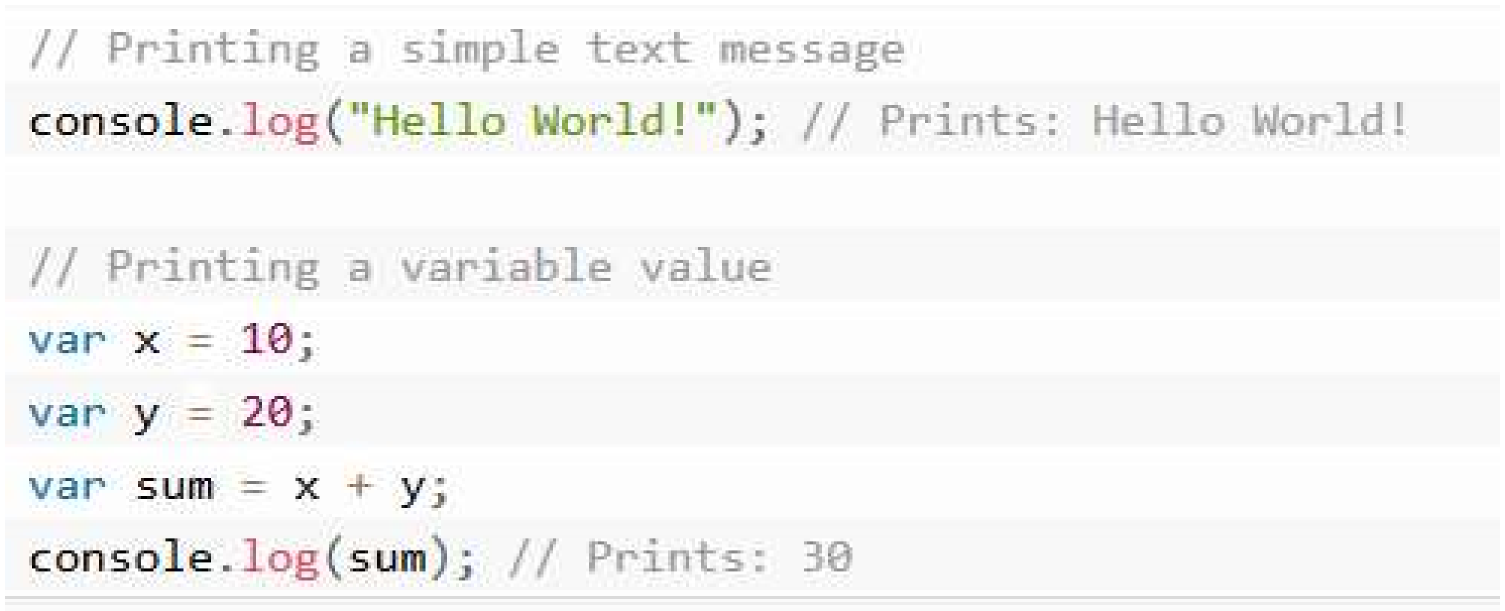
```
// Printing a simple text message
document.write("Hello World!"); // Prints: Hello World!

// Printing a variable value
var x = 10;
var y = 20;
var sum = x + y;
document.write(sum); // Prints: 30
```

Using window.alert()



```
// Displaying a variable value
var x = 10;
var y = 20;
var sum = x + y;
alert(sum); // Outputs: 30
```

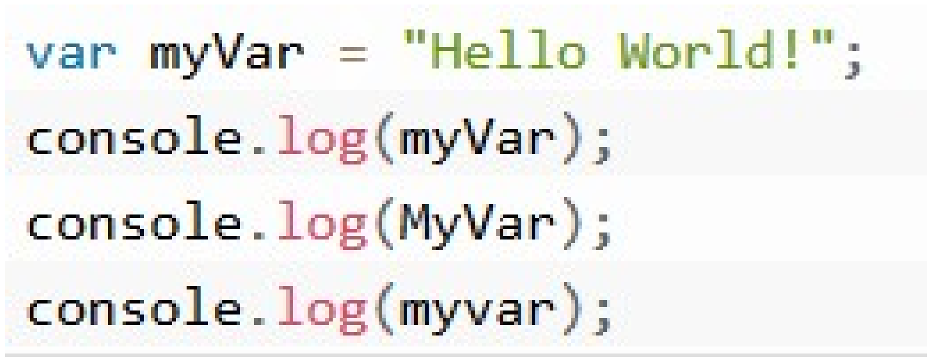
using **console.log()**



```
// Printing a simple text message
console.log("Hello World!"); // Prints: Hello World!

// Printing a variable value
var x = 10;
var y = 20;
var sum = x + y;
console.log(sum); // Prints: 30
```

JavaScript Statements

- JavaScript is case sensitive.



```
var myVar = "Hello World!";
console.log(myVar);
console.log(MyVar);
console.log(myvar);
```

- JavaScript statements can be grouped together in blocks.

```
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
}
</script>
```

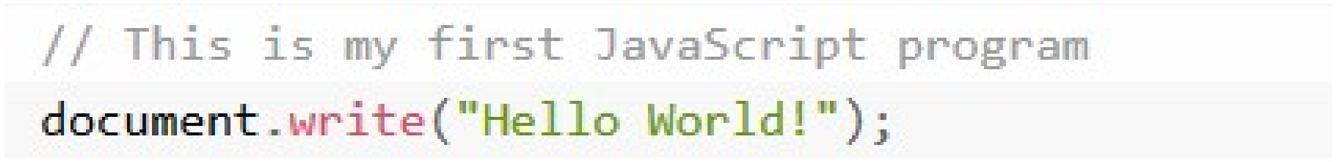
JavaScript Statements

- JavaScript ignores extra spaces. You can add white space to your script to make it more readable. The following lines are equivalent:

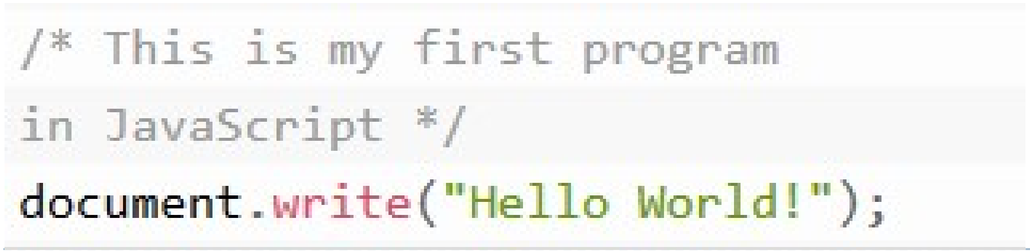
```
var name="Hege";
var name = "Hege";
```

Comments

- A comment is simply a line of text that is completely ignored by the JavaScript interpreter. Comments are usually added with the purpose of providing extra information pertaining to source code.
- It will not only help you understand your code when you look after a period of time but also others who are working with you on the same project.



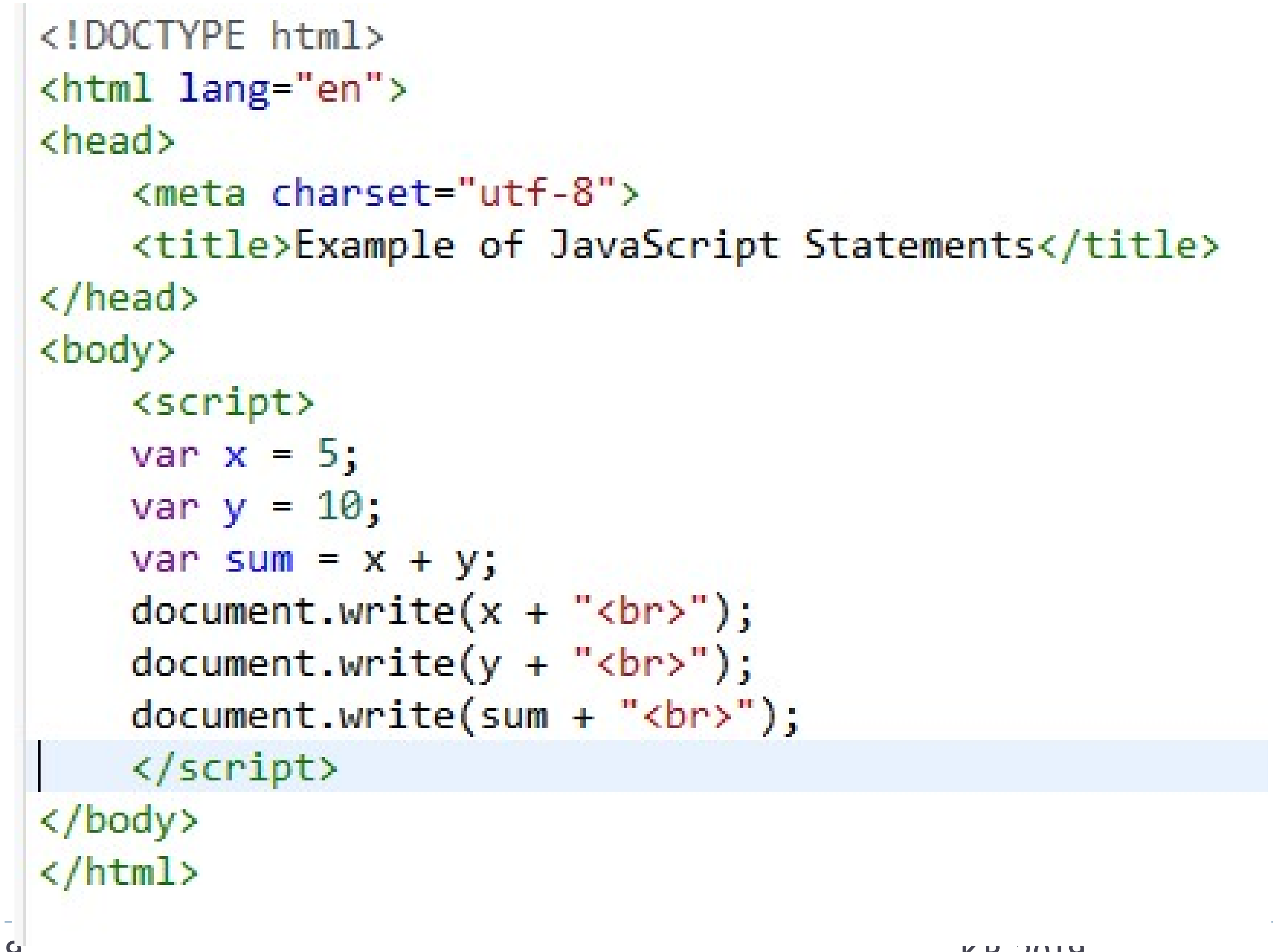
```
// This is my first JavaScript program
document.write("Hello World!");
```



```
/* This is my first program
in JavaScript */
document.write("Hello World!");
```

Variables &Operators

- JavaScript variables are "containers" for storing information:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Example of JavaScript Statements</title>
</head>
<body>
    <script>
    var x = 5;
    var y = 10;
    var sum = x + y;
    document.write(x + "<br>");
    document.write(y + "<br>");
    document.write(sum + "<br>");
    </script>
</body>
</html>
```

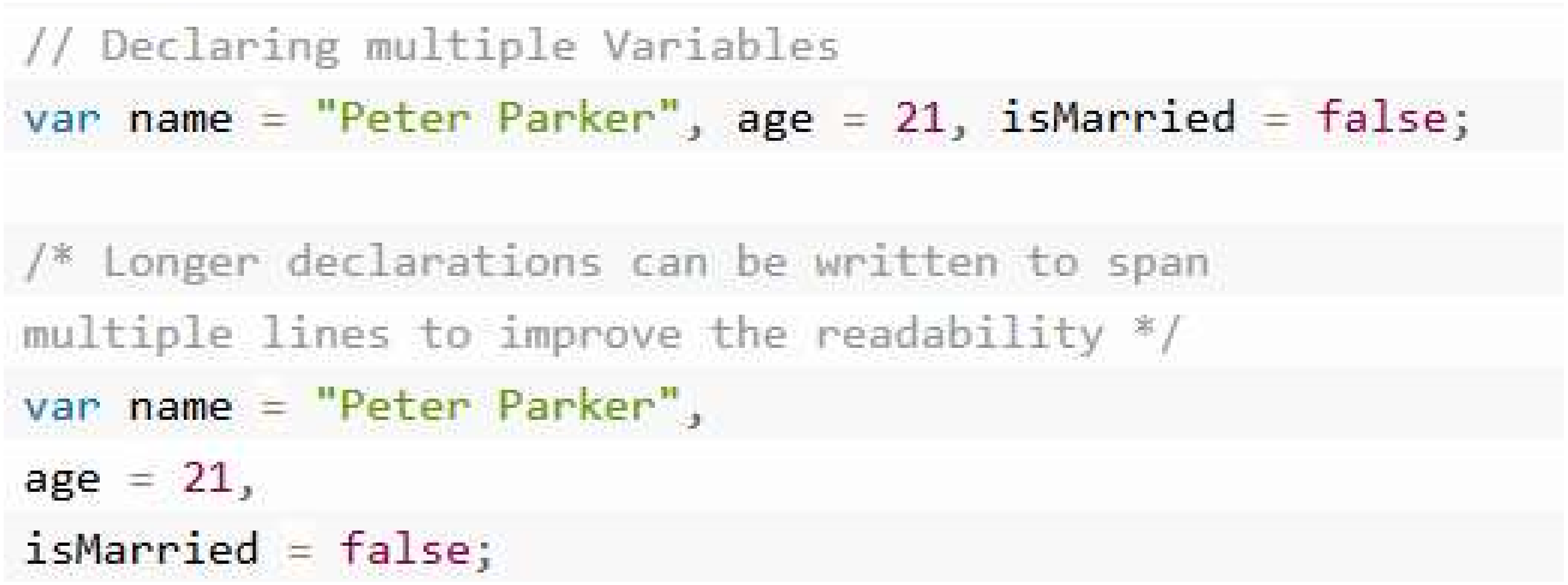
Variables &Operators

- Variable can have short names (like x and y) or more descriptive names (age, sum, totalvolume).
- Variable names must begin with a letter
- Variable names can also begin with $ and _ (but we will not use it)
- Variable names are case sensitive (y and Y are different variables)

Variable	note
Contoh_3.1	True
_contoh3.2	True
2001_angkatan	False
$sql	False

One Statement, Many Variables

- You can declare many variables in one statement. Just start the statement with **var** and separate the variables by comma:



```
// Declaring multiple Variables
var name = "Peter Parker", age = 21, isMarried = false;

/* Longer declarations can be written to span
multiple lines to improve the readability */
var name = "Peter Parker",
age = 21,
isMarried = false;
```

JavaScript primitive data types

- There are six basic data types in JavaScript which can be divided into three main categories: primitive (or *primary*), *composite* (or *reference*), and *special* data types.
- five types of primitive (or primary)data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript primitive data types

- A string can be any text inside quotes. You can use simple or double quotes:

```
var carname="Volvo XC60";
var carname='Volvo XC60';
```

```
var answer="It's alright";
var answer="He is called 'Johnny'";
var answer='He is called "Johnny"';
```

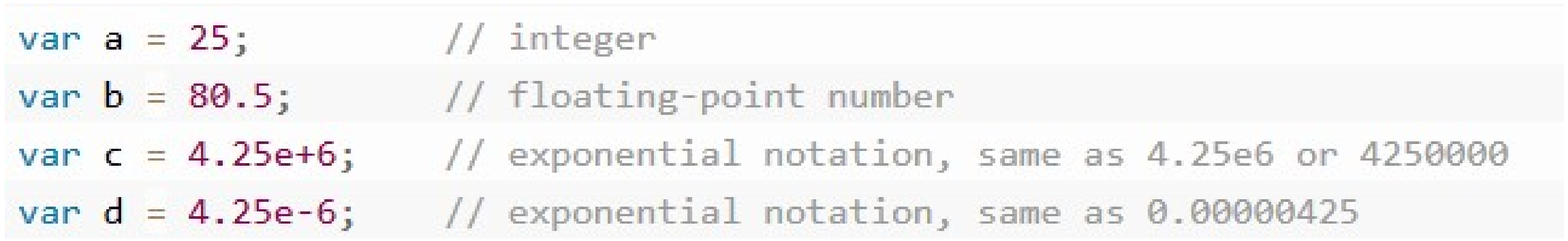
JavaScript Data Types

- JavaScript variables can also hold other types of data, like text values (name="Andrew Benjamin").
- In JavaScript a text like " Andrew Benjamin " is called a string.
- When you assign a text value to a variable, put double or single quotes around the value.
- When you assign a numeric value to a variable, do not put quotes around the value. If you put quotes around a numeric value, it will be treated as text.

```
var pi=3.14;
var name="Andrew Benjamin";
var answer='Yes I am!';
```

Data Types

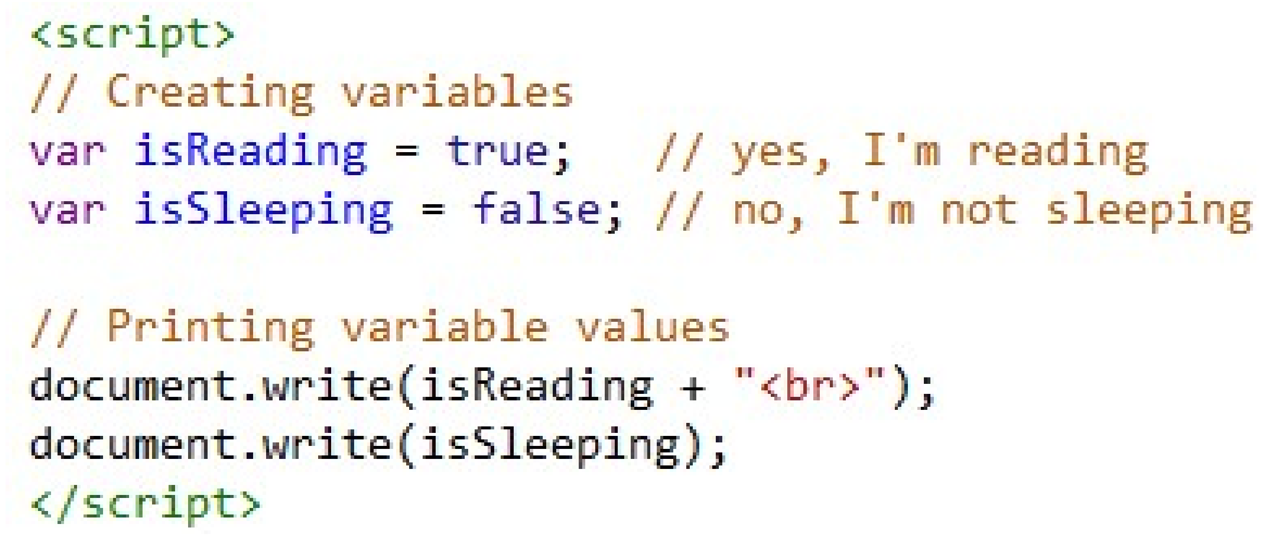
- **Numbers**
- JavaScript has only one type of numbers. Numbers can be written with, or without decimals:



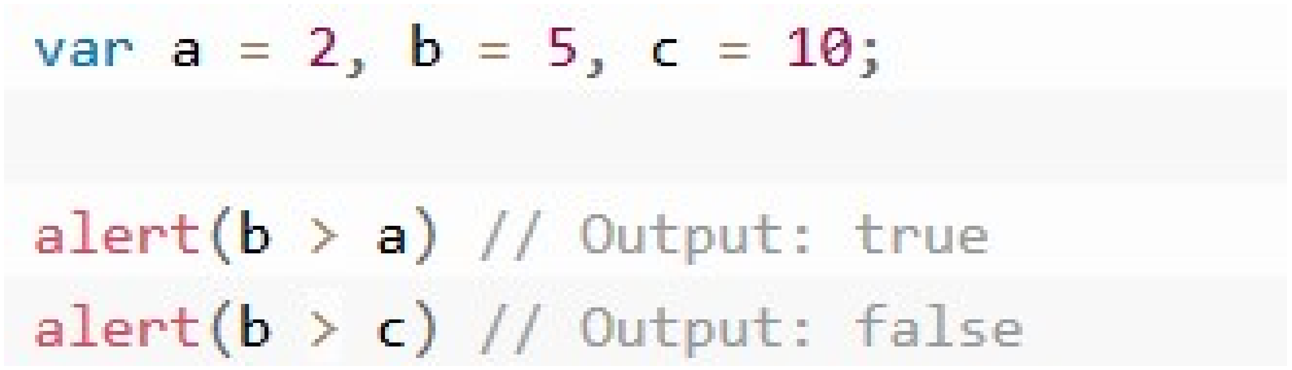
```
var a = 25;            // integer
var b = 80.5;          // floating-point number
var c = 4.25e+6;       // exponential notation, same as 4.25e6 or 4250000
var d = 4.25e-6;       // exponential notation, same as 0.00000425
```

Data Types

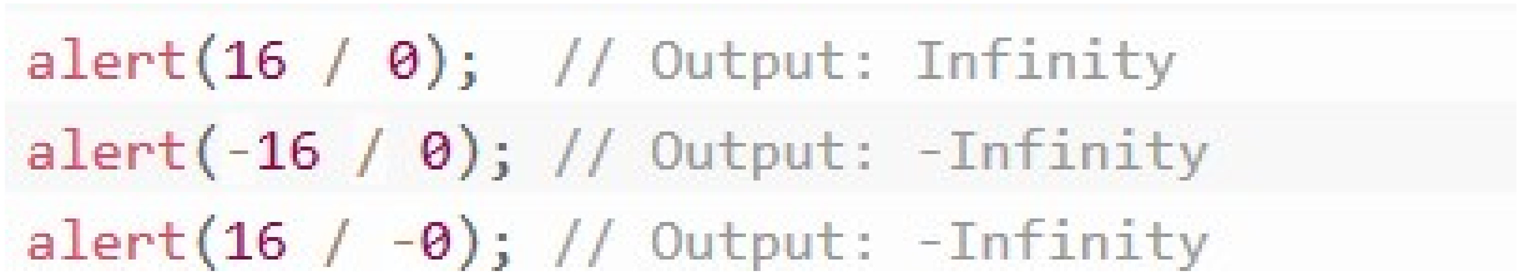
- **Booleans**
- true or false.
- Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.



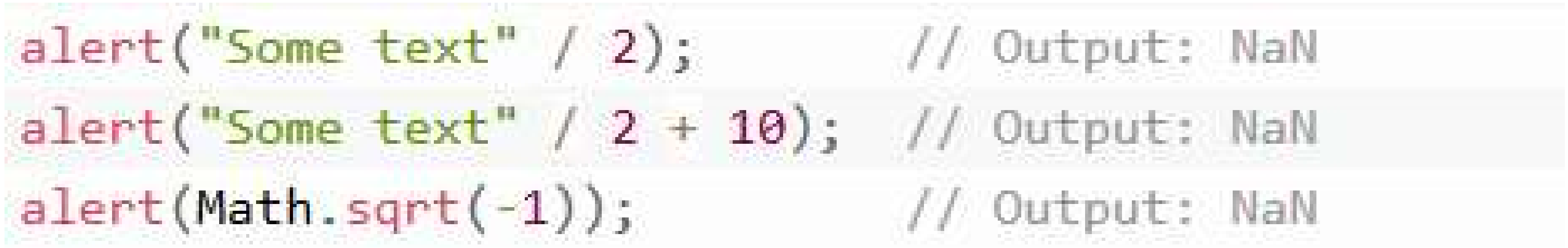
```
<script>
// Creating variables
var isReading = true;    // yes, I'm reading
var isSleeping = false; // no, I'm not sleeping

// Printing variable values
document.write(isReading + "<br>");
document.write(isSleeping);
</script>
```



```
var a = 2, b = 5, c = 10;

alert(b > a) // Output: true
alert(b > c) // Output: false
```



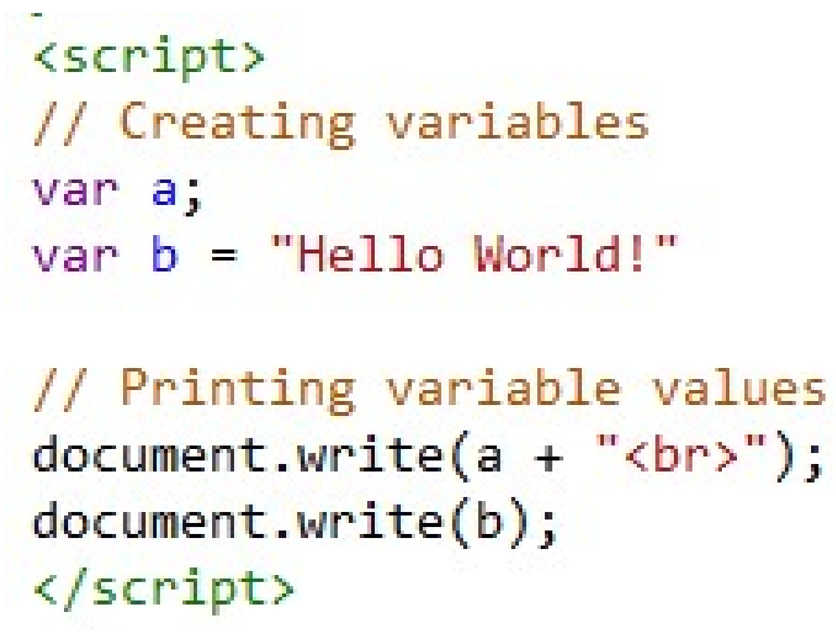
```
alert(16 / 0);  // Output: Infinity
alert(-16 / 0); // Output: -Infinity
alert(16 / -0); // Output: -Infinity
```



```
alert("Some text" / 2);       // Output: NaN
alert("Some text" / 2 + 10);  // Output: NaN
alert(Math.sqrt(-1));         // Output: NaN
```

Data Types

- **Undefined** is the value of a variable with no value.
- Variables can be emptied by setting the value to **null**;



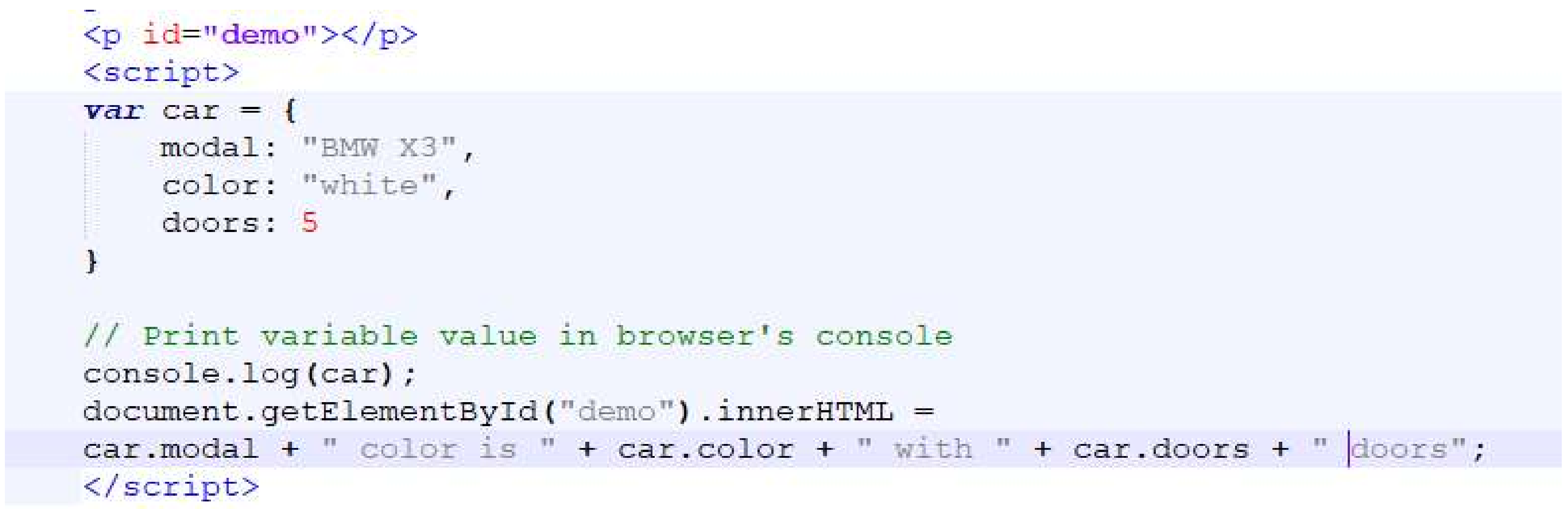
```
<script>
// Creating variables
var a;
var b = "Hello World!"

// Printing variable values
document.write(a + "<br>");
document.write(b);
</script>
```

Data Types

- The Null Data Type
- A null value means that there is no value. It is not equivalent to an empty string ("") or 0, it is simply nothing.
- A variable can be explicitly emptied of its current contents by assigning it the null value.

The Object Data Type

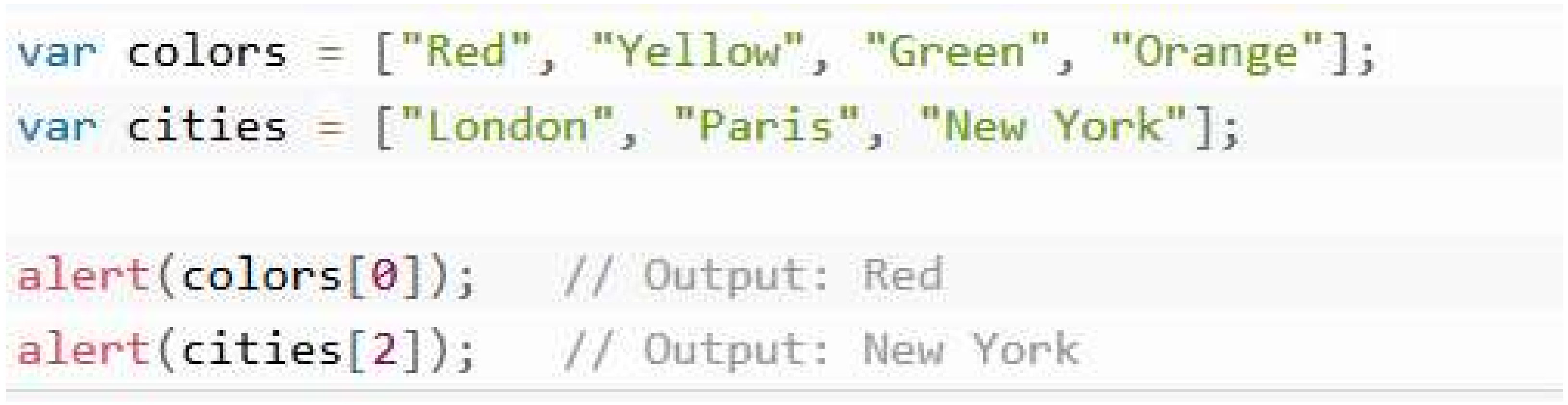
- The object is a complex data type that allows you to store collections of data.
- An object contains properties, defined as a key-value pair. A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects.
- JavaScript objects are written with curly braces {}.
- Object properties are written as name:value pairs, separated by commas.



```
<p id="demo"></p>
<script>
var car = {
    modal: "BMW X3",
    color: "white",
    doors: 5
}

// Print variable value in browser's console
console.log(car);
document.getElementById("demo").innerHTML =
car.modal + " color is " + car.color + " with " + car.doors + " doors";
</script>
```

The Array Data Type

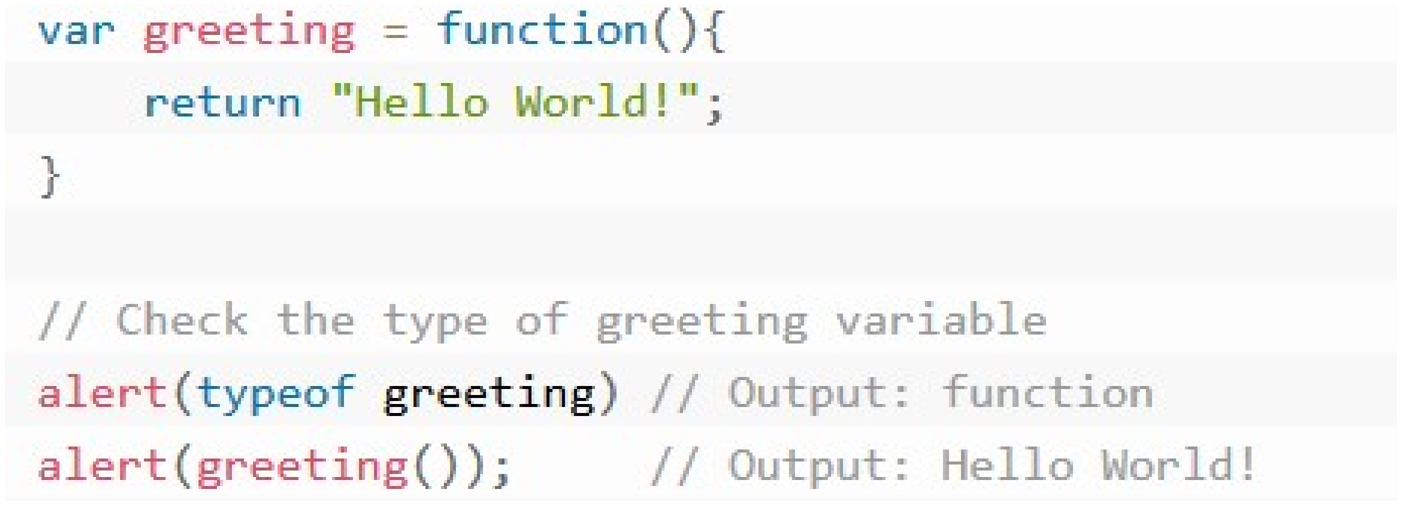
- An array is a type of object used for storing multiple values in single variable.
- Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays.
- The array index starts from 0, so that the first array element is arr[0] not arr[1].



```
var colors = ["Red", "Yellow", "Green", "Orange"];
var cities = ["London", "Paris", "New York"];

alert(colors[0]);    // Output: Red
alert(cities[2]);    // Output: New York
```

The Function Data Type

- The function is callable object that executes a block of code. Since functions are objects, so it is possible to assign them to variables, as shown in the example below:



```
var greeting = function(){
    return "Hello World!";
}

// Check the type of greeting variable
alert(typeof greeting) // Output: function
alert(greeting());     // Output: Hello World!
```

JavaScript Operators

- JavaScript uses an **assignment operator** (=) to **assign** values to variables:

  (5 + 6) * 10

- Keyword :

```
<script>
	var x = 5 + 6;
	var y = x * 10;
	document.getElementById("demo").innerHTML = y;
</script>
```

The Concept of Data Types

- When adding a number and a string, JavaScript will treat the number as a string.

```
var x = 16 + "Volvo";   //16Volvo
var x = "Volvo" + 16;   //Volvo16
var x = 16 + 4 + "Volvo";   //20Volvo
var x = "Volvo" + 16 +4;  //Volvo164
```

The typeof Operator

- You can use the JavaScript **typeof** operator to find the type of a JavaScript variable:

```
typeof "John"                // Returns "string"
typeof 3.14                 // Returns "number"
typeof false              // Returns "boolean"
```

Operators

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

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

Assignment Operators

Assignment operators are used to assign values to JavaScript variables.
Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.
If x=5,

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true