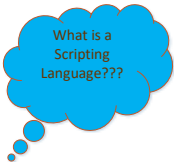
ABESIT
CRC-Training

JavaScript

ABESIT
CRC-Training

Introduction to JavaScripts

- What is JavaScript???
  - ✓ JavaScript is a scripting Language created by Netscape



- Scripting Language is a lightweight programming language.
- Scripting Languages are not needed to be compiled.
- The language is interpreted at runtime.

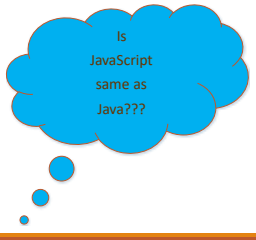
ABESIT
CRC-Training

Introduction to JavaScript (Contd.)

- A JavaScript is usually directly embedded in an HTML page.
- External JavaScripts can be created which can be used by HTML pages.
- JavaScript adds interactivity to HTML pages.
- JavaScript's are integrated into the browsing environment.

ABESIT
CRC-Training

Introduction to JavaScript (Contd.)



- Java is a programming language which requires compilation and interpretation.
- Java is used to make large scale applications.
- JavaScript is a scripting language which just requires interpretation. The script is some set of commands which the browser interprets.
- JavaScript is used to add interactivity in HTML Pages.

Introduction to JavaScript (Contd.)

- Bulb on off

Introduction to JavaScript (Contd.)

- Bulb on off

Introduction to JavaScript (Contd.)

- Bulb on off

```
document.getElementById()
```


Types of JavaScript

- **Client-Side JavaScript (CSJS)** -- an extended version of JavaScript that enables the enhancement and manipulation of web pages and client browsers.
- **Server-Side JavaScript (SSJS)** -- an extended version of JavaScript that enables back-end access to databases, file systems, and servers.
- **Core JavaScript** -- the base JavaScript language.

Core JavaScript

- Core JavaScript encompasses all of the statements, operators, objects, and functions that make up the basic JavaScript language.
- The following objects are part of core JavaScript:
  - ✓ array
  - ✓ date
  - ✓ math
  - ✓ number
  - ✓ string

Client Side Java Scripting

- CSJS is composed of core JavaScript and many additional objects, such as the following:
  - ✓ document
  - ✓ form
  - ✓ frame
  - ✓ window
  - ✓ Navigator
  - ✓ History

Server Side JavaScript

- SSJS is composed of core JavaScript and additional objects and functions for accessing databases and file systems, sending e-mail, and so on.

Uses of JavaScript (Client Side)

- Menus for Navigation
- Form Validation
- Popup Windows
- Password Protecting
- Math Functions
- Special effects with document and background
- Status bar manipulation
- Messages
- Mouse Cursor Effects
- Links

Test Your Understanding

_____ is an extended version of JavaScript that enables the enhancement and manipulation of web pages and client browsers

- Client Side JavaScript
- Server Side JavaScript
- Core JavaScript

Test Your Understanding

_____ is an extended version of JavaScript that enables the enhancement and manipulation of web pages and client browsers

- Client Side JavaScript
- Server Side JavaScript
- Core JavaScript

Syntax rules of JavaScript

- Statements may or may not contain a semicolon at the end.
- Multiple statements on one line must be separated by a semicolon.
- JavaScript is case sensitive.

Using <script> tag

- The HTML <script> tag is used to enter JavaScript into a HTML.
- The <script> tag can be embedded within
  - <head> tag.
  - <body> tag.
- JavaScript in the head section will be executed when called.
- JavaScript in the body section will be executed while the HTML page is loaded.
- Unlimited number of JavaScript's can be placed both in head and body section in a HTML document.

Using <script> tag

Eg:

```
<html>  <head><title>Example</title>
</head>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Is a standard command for writing output to a page

Using an External JavaScript

- A JavaScript can be written in an external file, which can be used by different HTML pages.
- The external script cannot contain the <script> tag.
- The external file needs to end with the .js extension.

Using External JavaScript (contd.)

External.js

```
document.write("This line has been
writen in the External JavaScript!!!")
```

JavaScript.html

```
<html>
<head><title>Example</title>
</head>
<body>
<script src="External.js">
</script>
<p >
This line has been written in the
html page!!!
</p>
</body>
</html>
```


Using External JavaScript (contd.)

	Internal Scripting	External Scripting
Loading time	Faster, as it is written within the HTML page	Slower, as it is loaded from server, whenever requested
When to use?	If number of lines of code is less	For large amount of code
Re-usable	No, you cannot re-use JavaScript code with any other HTML file	Yes, Same JavaScript file can be used in multiple HTML files
Maintainability	Difficult, as for every change request, each HTML pages containing JavaScript code has to be modified separately	Easy, as only 1 file needs to be modified

Test Your Understanding

Select the Correct Statement/s

- <script></script> is embedded within <head> </head>
- <script></script> is embedded within <body></body>
- <script></script> is embedded within <title></title>

Test Your Understanding

Select the Correct Statement/s

- <script></script> is embedded within <head> </head>
- <script></script > is embedded within <body></body>
- <script></script> is embedded within <title></title>

JavaScript Operators & Expressions

JavaScript Variables and expression

- **JavaScript Variables**
  - Variable:
    - A variable is a symbolic name that represents some data in the memory.
    - A variable value can change during the execution of the JavaScript.
    - A variable can be referred by its name to see or change its value.
    - Variables are name are case sensitive.
    - Must begin with a letter or underscore.

ABESIT
CRC-Training

JavaScript Variables Types

- The identifiers (Variables) in JavaScript can be categorized into three as shown below. They can be declared into specific type based on:
  - The data which an identifier will hold and
  - The scope of the identifier

ABESIT
CRC-Training

Rules of a Variable

- **Variable Declaration**
  - Variables can be declared using the var statement
    - var <variable name> = some value
    - let <variable name> = some value
    - const <variable name> = some value
  - Variables can also be created without using var statement
    - <variable name> = some value
    - Eg

    var firstname = "Samuel"
    OR
    let firstname = "Samuel"
    OR
    const firstname = "Samuel"
    OR
    firstname = "Samuel"

4 Ways to Declare a JavaScript Variable:
- ✓ Using **var**
- ✓ Using **let**
- ✓ Using **const**
- ✓ Using nothing

ABESIT
CRC-Training

Difference – let, var, const

- The Difference between let, const and var.

Keyword	Scope	Declaration	Assignment
let	Block	Redeclaration not allowed	Re-assigning allowed
const	Block	Redeclaration not allowed	Re-assigning not allowed
var	Function	Redeclaration allowed	Re-assigning allowed

ABESIT
CRC-Training

Problem Statement:

Observe the output to see the declaration of different identifiers i.e. var, let, and const.

- Create two more variables named celsiusScale using let and var,
- NORMAL_FAHREN using const for storing the normal temperature and calculated temperature in celsius.

```
<html>
<body>
        <h3>Body Temperature</h3>
        <script>
          var tempFahrenheit = 99;

          let TempFahrenheit = 99;

          const TEMP_CELSIUS = 38;

          document.write("Default temperature (var) is: " +
tempFahrenheit + "</span> <br/>");

          document.write("Default temperature (let) is: " +
TempFahrenheit + "</span> <br/>");

          document.write("Normal body temperature in Celsius
is: " + TEMP_CELSIUS + "</span>");
        </script>
</body>
```

Test Your Understanding

ABESIT CRC-Training

Dheeraj has written a simple program to calculate the area of various geometric shapes. How should he declare an identifier to store the value of 'Pi'?

- var pi = 3.14;
- let pi=3.14;
- const pi=3.14;

Test Your Understanding

ABESIT CRC-Training

Dheeraj has written a simple program to calculate the area of various geometric shapes. How should he declare an identifier to store the value of 'Pi'?

- var pi = 3.14;
- let pi=3.14;
- const pi=3.14;

Test Your Understanding

ABESIT CRC-Training

Which among the following is True regarding JavaScript Identifiers?

- Identifier declared using var keyword cannot be redeclared.
- Usage of let keyword in Identifier declaration will allow you re-declaration of same identifier later in the program.
- If an Identifier is declared using const keyword its mandatory to initialize it at the time of declaration.
- In JavaScript var and let keyword have same scope.

Test Your Understanding

ABESIT CRC-Training

Which among the following is True regarding JavaScript Identifiers?

- Identifier declared using var keyword cannot be redeclared.
- Usage of let keyword in Identifier declaration will allow you re-declaration of same identifier later in the program.
- If an Identifier is declared using const keyword its mandatory to initialize it at the time of declaration.
- In JavaScript var and let keyword have same scope.

Test Your Understanding

Observe the below code:

```
const rollNumber = 1; //line 1
var firstName = "Aaliya"; // line 2
let course ="JavaScript"; // line 3
rollNumber= 2; // line 4
console.log("Hi " + firstName +
", your roll number is " +
rollNumber + ". Hope you have
knowledge on HTML5");
```

Which line in the code is likely to throw an error?

- line 2
- line 1
- line 3
- line 4

Test Your Understanding

Observe the below code:

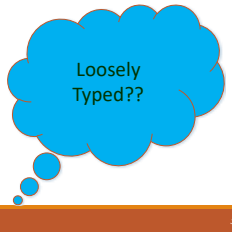
```
const rollNumber = 1; //line 1
var firstName = "Aaliya"; // line 2
let course ="JavaScript"; // line 3
rollNumber= 2; // line 4
console.log("Hi " + firstName +
", your roll number is " +
rollNumber + ". Hope you have
knowledge on HTML5");
```

Which line in the code is likely to throw an error?

- line 2
- line 1
- line 3
- line 4

Data Type in JavaScript

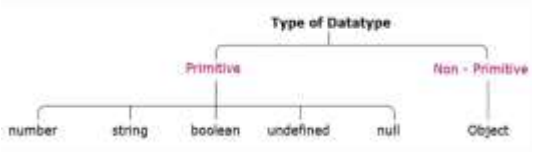
- JavaScript is a loosely typed language.



- Data Type of Variable need not be specified during declaration.
- Data types are automatically converted during script execution.

Data Type in JavaScript (contd.)

- JavaScript recognizes the following type of values:

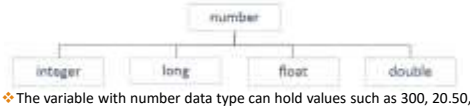


Data Type in JavaScript (contd.)

- The data is said to be **primitive** if it contains an individual value.

Number

- To store a variable that holds a numeric value, the primitive data type number is used.



- The variable with number data type can hold values such as 300, 20.50, 10001, and 13456.89.

Data Type in JavaScript (contd.)

- Constant of type number can be declared like this:

```
const pi = 3.14; // its value is 3.14
const smallestNaturalNumber = 0; // its value is 0
```

- In JavaScript, any other value that does not belong to the above-mentioned types is not considered as a legal number. Such values are represented as NaN (Not-a-Number).

```
let result = 0/0; // its value is NaN
let result = "Ten" * 5; //its value is NaN
```


Data Type in JavaScript (contd.)

String

- When a variable is used to store textual value, a primitive data type string is used. Thus, the string represents textual values. String values are written in quotes, either single or double.

```
let personName= “Ramesh”;   // let personName = ‘Ramesh’;

let ownership= "Ramesh's";   //    let ownership = 'Ramesh"s';

let ownership= "Ramesh"s";   //      let ownership = 'Ramesh's';
```

It is a syntax error.

Data Type in JavaScript (contd.)

Literals

- Literals can span multiple lines and interpolate expressions to include their results.

```
let firstName="Kamal";
let lastName="Pratap";
console.log("Name: " + firstName + " " + lastName + " \n Email: " +
firstName + "_" + lastName + "@abesit.in");
```

- literals can also be used as:

```
let firstName="Kamal";
let lastName="Pratap";
console.log(`Name:${firstName} ${lastName}
Email: ${firstName}_${lastName}@abesit.in`);
```

Data Type in JavaScript (contd.)

Boolean

- When a variable is used to store a logical value that can always be true or false then, primitive data type Boolean is used.
- Thus, Boolean is a data type which represents only two values: true and false.
- Values such as 100, -5, “Cat”, 10<20, 1, 10*20+30, etc. evaluates to true.
- whereas 0, “”, NaN, undefined, null, etc. evaluates to false.

Data Type in JavaScript (contd.)

Undefined

- When the variable is used to store "no value", primitive data type undefined is used.
- Any variable that has not been assigned a value has the value undefined and such variable is of type undefined. The undefined value represents "no value".

```
let stuName;      //here value and the data type are undefined
```

```
let stuName = “Ram";      //here value is John and the data type is String
stuName = undefined;      //here value and the data type are undefined
```


Data Type in JavaScript (contd.)

null

- The null value represents "no object".
- Null data type is required as JavaScript variable intended to be assigned with the object at a later point in the program can be assigned null during the declaration.

```
let item = null;
```

- If required, the JavaScript variable can also be checked if it is pointing to a valid object or null.

```
document.write(item==null);
```


Non-Primitive Data Types

- The data type is said to be non-primitive if it is a collection of multiple values.
- The variables in JavaScript may not always hold only individual values which are with one of the primitive data types.
- There are times a group of values are stored inside a variable.
- JavaScript gives non-primitive data types named Object and Array, to implement this.

Non-Primitive Data Types

Objects

- Objects in JavaScript are a collection of properties and are represented in the form of [key-value pairs].
- The 'key' of a property is a string or a symbol and should be a legal identifier.
- The 'value' of a property can be any JavaScript value like Number, String, Boolean, or another object.

```
{
  key1 : value1,
  key2 : value2,
  key3 : value3
};
```

```
let mySmartPhone = {
    name: "iPhone",
    brand: "Apple",
    platform: "iOS",
    price: 50000
};
```


Non-Primitive Data Types

Array

- The Array is a special data structure that is used to store an ordered collection, which cannot be achieved using the objects.

- There are two ways of creating an array:

```
let dummyArr = new Array();

let dummyArr = [];

let digits =[1,2,3,"four"];
```


Test Your Understanding

Sachin wants to store the below-mentioned value:

i. 6745000
ii. Carlos
iii. 234.16789

What will be the recommended data type to be used for the above values?

- i – Double, ii – String, iii – Float
- i – Number, ii – Char, iii – Number
- i – Integer, ii – String, iii – Float
- i – Number, ii – String, iii – Number

Test Your Understanding

Sachin wants to store the below-mentioned value:

i. 6745000
ii. Carlos
iii. 234.16789

What will be the recommended data type to be used for the above values?

- i – Double, ii – String, iii – Float
- i – Number, ii – Char, iii – Number
- i – Integer, ii – String, iii – Float
- i – Number, ii – String, iii – Number

Test Your Understanding

Observe the below-mentioned code snippet:

```
let value1 = 12;
value1 = "Sample";
value1 = 34.67;
```

What will be the type of variable 'value1'?

- Float
- String
- Number
- Integer

Test Your Understanding

Observe the below-mentioned code snippet:

```
let value1 = 12;
value1 = "Sample";
value1 = 34.67;
```

What will be the type of variable 'value1'?

- Float
- String
- Number
- Integer

Test Your Understanding

What will be the output of the below-mentioned code snippet?

```
let person = {
   name: "Joe Camel",
   age: 42,
    status: "dead"
 }
 console.log(typeof person);
```

- Number
- Array
- Boolean
- Object

Test Your Understanding

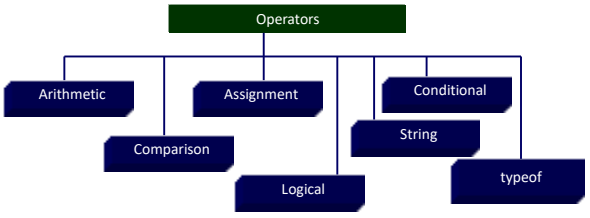
What will be the output of the below-mentioned code snippet?

```
let person = {
   name: "Joe Camel",
   age: 42,
    status: "dead"
 }
 console.log(typeof person);
```

- Number
- Array
- Boolean
- Object

ABESIT
CRC-Training

JavaScript Operators



ABESIT
CRC-Training

JavaScript Operator (contd.)

Arithmetic

ABESIT
CRC-Training

JavaScript Operator (contd.)

Comparison

ABESIT
CRC-Training

JavaScript Operator (contd.)

Comparison

Strict equality (===) and strict inequality (!==) operators consider only values of the same type to be equal.

ABESIT
CRC-Training

JavaScript Operator (contd.)

Assignment

Operator	Example	Same as	Result
=	x=10		10
+=	x=10 x+=10	x=x+10	20
-=	x=10 x-=5	x=x-10	5
*=	x=10 x*=5	x=x*10	50
/=	x=10 x/=5	x=x/5	2
%=	x=10 x%=5	x=x%5	0

ABESIT
CRC-Training

JavaScript Operator (contd.)

Logical

Operator	Description	Example	Result
&&	and	x=5 y=3 x<5 && x%y>4	false
||	or	x=5 y=1 x<3 || x%y>4	true
!	not	x=5 y=3 !(x<3 && !(x%y)>4	true

ABESIT
CRC-Training

JavaScript Operator (contd.)

String

Operator	Description	Example	Result
+	Concatenation	x="Java" y="Script" z=x+y	JavaScript

ABESIT
CRC-Training

JavaScript Operator (contd.)

Conditional

Description	Syntax	Example	Result
Conditional	variablename=(condition)?value1:value2	x=5 y=2 z=(x>y)?x:y	5

JavaScript Operator (contd.)

typeof

Description	Example	Result
Checks the data type of a variable	x=5 typeof(x)	number
	x="Samuel" typeof(x)	string
	x typeof(x)	undefined

Test Your Understanding

x = 20
x = "Test"
typeof(x) evaluates to

- **number**
- **string**
- **null**

Test Your Understanding

x = 20
x = "Test"
typeof(x) evaluates to

- **number**
- **string**
- **null**

Test Your Understanding

Which of the given conditions will return TRUE?

- null==undefined;
- null===undefined;
- null=="undefined";
- null==="undefined";

Test Your Understanding

ABESIT CRC-Training

Which of the given conditions will return TRUE?

- null==undefined;
- null===undefined;
- null=="undefined";
- null==="undefined";

Test Your Understanding

ABESIT CRC-Training

Choose the correct output of the given code:

```
let num1 = 20;
let num2 = "20";

console.log(num1==num2);
console.log(num1===num2);
```

- true;true;
- true;false;
- false;true;
- false;false;

Test Your Understanding

ABESIT CRC-Training

Choose the correct output of the given code:

```
let num1 = 20;
let num2 = "20";

console.log(num1==num2);
console.log(num1===num2);
```

- true;true;
- true;false;
- false;true;
- false;false;

Test Your Understanding

ABESIT CRC-Training

```
<html>
  <script>
        let x, y=5
      console.log(x);
      console.log(y);
  </script>
  <body>
  </body>
</html>
```

What values will be logged to the console when the above web page is rendered on the browser.

- undefined and 5
- 5 and undefined
- 5 and 5
- undefined and undefined