CodeWithHarry

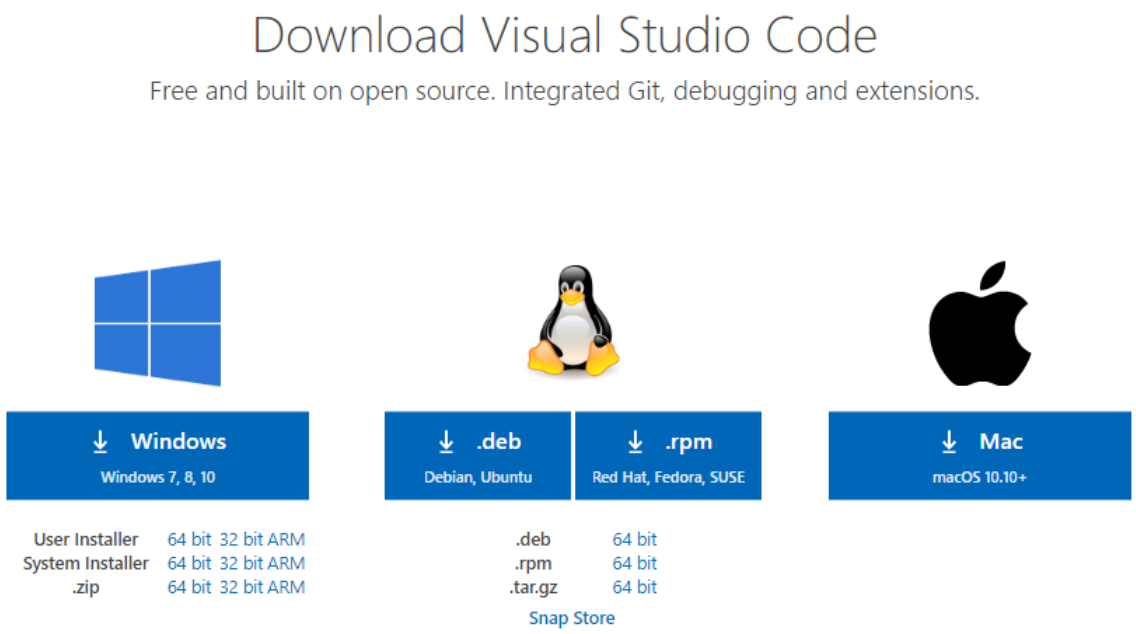
HTML CSS JS C C++ JAVA PYTHON PHP

Data Types in JavaScript (Primitive & Reference Types) | JavaScript Tutorial In Hindi #4

Overview Q&A Downloads Announcements

Data Types in JavaScript (Primitive & Reference Types) | JavaScript Tutorial In Hindi #4

To run a **JS program**, we need an IDE like Visual Studio Code or Codeblocks. For this series, we are using Virtual Studio Code (VS Code). ***Visual Studio Code*** is a fast source code editor and provides the tools that a developer needs for a quick code-build-debug cycle. To download VS Code, click on ***Download Virtual Studio Code***. For guidance, check the tutorial ***Installing VS Code, Extensions & Setup***



In today's tutorial, we will study about the ***primitive and reference datatype*** and their differences. In JavaScript, a variable stores two types of values: primitive and reference. JavaScript provides six primitive types like Boolean, number, string, undefined, null, symbol, and a reference type as object literals, array, functions, and dates. When we assign a value to a variable, the JavaScript engine will determine whether the value belongs to primitive or reference datatype. We do not have to specify the type of data that whether it is a string or an integer.

What are Primitive and Reference Types in JavaScript?

Primitive types:-

Primitive data types are numbers, Booleans, strings, undefined, null, and symbol. **Primitive data types** are the **basic** or **common** data types in JavaScript. Following is the table of primitive data types in JavaScript.

Primitive Data Type	Meaning
var x;	undefined
var x=undefined;	undefined
var x=null;	null type data
var x=3;	number.
var x=6.5	number
var x="5"	string
var x='5'	string
var x="Hello World"	string
var x=true	Boolean
var x=false;	Boolean

Here is an example:

```
let number = 20
```

The number variable stores a number value i.e. 20. Number values are called "primitive values" because they are simple building blocks of JavaScript apps.

```
let name = 'Harry'
let isMale = true
```

String and Booleans are also primitive types so we can also declare variables using var and const.

Reference Types

Reference in JavaScript are datatypes based on primitive. Like Objects, Arrays, and Functions. Everything is JavaScript is either an Object or Primitive datatype. When we create an object, that value is not directly assigned to the variable. Instead, a reference to that value is what gets set. Variable knows about the location of the object in memory, not the object itself. Following is the table of the reference data type.

Reference Data Type	Meaning
var y=["March", "April", "May"]	Array
var y={ name : "Harry", age : 22, gender : "male" }	Object
var y=function(){ }	Function on
var y=new Date();	Date

Here's an example:-

```
var student = {
  name: 'Harry',
  age: 20,
} //object
var sports= ['Tennis', 'Cricket']//array
```

Here, a student is an object and, therefore, a so-called reference type. The student object holds properties that have primitive values. This does not affect the object being a reference type, though.
The sports array is also a reference type - in this case, it holds a list of strings. A string is a primitive datatype, but it does not affect the array. Arrays are reference types.

What is the difference between primitive and reference datatype?

- JavaScript stores the primitive value on the stack because the size of a primitive value is fixed. On the other hand, JavaScript stores the reference value on the heap because the size of the reference value is dynamic.
- One of the most significant differences between primitive data and reference data is that, If the value is primitive, then we manipulate the **actual value**stored in that variable. Whereas, If the value is of reference data type, we can manipulate that object's reference, rather than the actual object. It means a variable that stores an object is **accessed by reference**.

Code index.html as described/written in the video

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Tutorial on Js</title>
</head>
```

```
<body>
    <h1>This is Js tutorial by Harry</h1>
</body>

<!-- <script src="js/tut2.js"></script> -->
<!-- <script src="js/tut3.js"></script> -->
<!-- <script src="js/tut4.js"></script> -->
<!-- <script src="js/tut5.js"></script> -->
<!-- <script src="js/tut6.js"></script> -->
<!-- <script src="js/tut7.js"></script> -->
<!-- <script src="js/tut8.js"></script> -->
<!-- <script src="js/tut9.js"></script> -->
<!-- <script src="js/tut10.js"></script> -->
<script src="js/tut11.js"></script>
</html>
```

Js code as described/written in the video

```
// Primitive data types

// String
let name = "harry";
console.log("My string is " + name);
console.log("Data type is " + (typeof name));

// Numbers
let marks = 34.4;
console.log("Data type is " + (typeof marks));

// Boolean
let isDriver = true;
console.log("Data type is " + (typeof isDriver));

// Null
let nullVar = null;
console.log("Data type is " + (typeof nullVar));

// Undefined
```

```
let undef = undefined;
console.log("Data type is " + (typeof undef));

// Reference Data Types

// Arrays
let myarr = [1, 2, 3, 4, false, "string"];
console.log("Data type is " + (typeof myarr));

// Object Literals
let stMarks = {
    harry: 89,
    Shubham: 36,
    Rohan: 34
}
console.log( typeof stMarks);

function findName() {

}
console.log( typeof findName);

let date = new Date();
console.log( typeof date);
```

Previous Next