JavaScript

Module 2 - JavaScript Basics

Chapter 4 - Data types

Aitrich

Data Types

JavaScript provides different data types to hold different types of values.

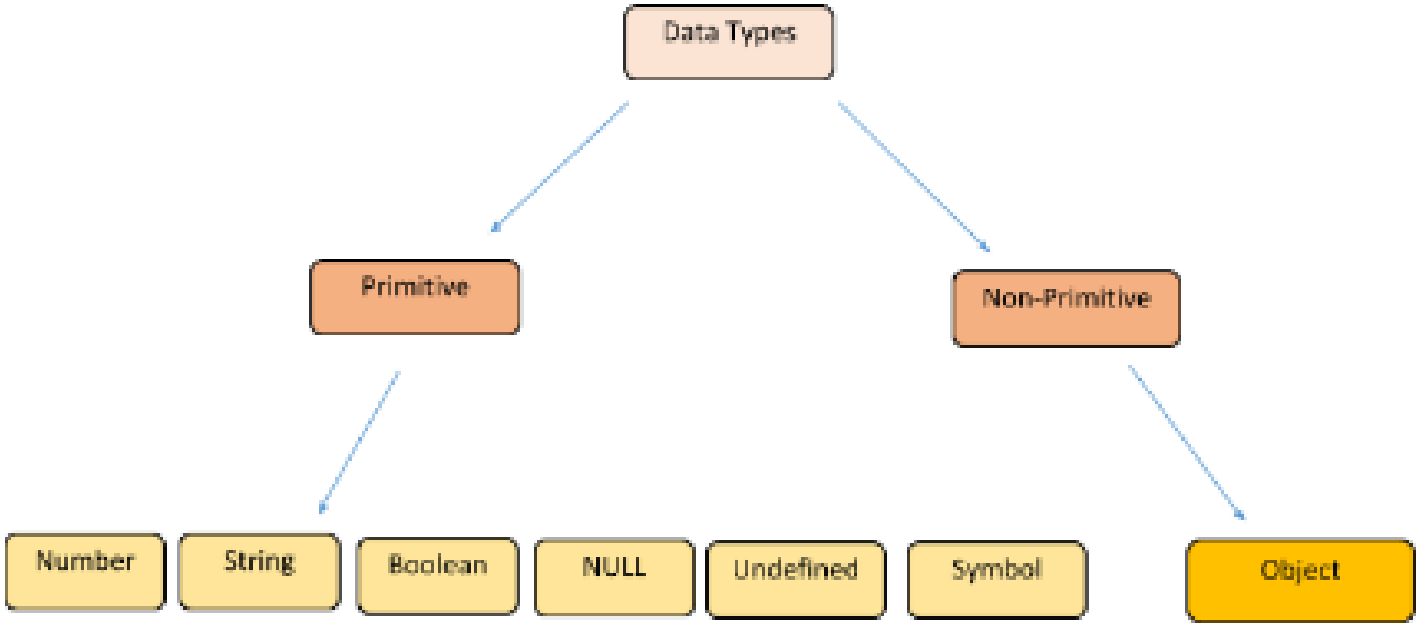
There are two types of data types in JavaScript.

- **Primitive data type**
- **Non-primitive (reference) data type**

- Primitive data types are data types that refer to a single value, in an address in memory. E.g. `var a =5;`
- Non-primitive data types refer to the 'address' in memory which contains single or multiple key-value pair/s. ie, The non-primitive type can store collections of data.

Note: A JavaScript variable can hold any type of data.

Data Types


Data Types
Primitive
Non-Primitive
Number
String
Boolean
NULL
Undefined
Symbol
Object

Data Types

The Object Datatype (Non-Primitive) contains

1. An object
2. An array
3. A date

Data Types - 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 institute = {firstName:"Aitrich", lastName:"Academy"};

// Array object:
const jobs= ["Manager", "Accountant", "Receptionist"];

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

DataType Concept

```
<!DOCTYPE html>
<html>
<body>
<p id="msg"></p>
<script>
let x = 16 + "Aitrich";
let y = "Aitrich" + 16 ;
let z = 16 + 4 + "Aitrich";
let u = "Aitrich" + 16 + 4;
document.getElementById("msg").innerHTML =
x+"<br>"+y+"<br>"+z+"<br>"+u ;
</script>
</body>
</html>
```

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

JavaScript evaluates expressions from left to right.Different sequences can produce different results.

Output

16Aitrich
Aitrich16
20Aitrich
Aitrich164

Aitrich

Strings

A string is a series of characters like "Aitrich Academy".

Strings are written with quotes. You can use single or double quotes.

```
// Using double quotes:
let jobName1 = "Manager";

// Using single quotes:
let jobName2 = 'Accountant';
```

Strings

Using Quotes inside a string:

The Quotes should not match the quotes surrounding the string.

```
// Single quote inside double quotes:
let msg1 = "It's alright";

// Single quotes inside double
quotes:
let msg2 = "Name is 'Aitrich'";

// Double quotes inside single
quotes:
let msg3 = 'Name is "Aitrich"';
```

Output

```
It's alright
He is called 'Johnny'
He is called "Johnny"
```

Aitrich

Numbers

Numbers are stored as decimal numbers (floating point).

Numbers can be written with, or without decimals:

```
// Without decimals:
let x2 = 34;

// With decimals:
let x1 = 34.00;
Let x3 = 3.14;
```

Output

```
34
34
3.14
```

Most programming languages have many number types: Ex:-Whole numbers (integers), byte (8-bit), short (16-bit), int (32-bit), long (64-bit),Real numbers (floating-point),float (32-bit), double (64-bit).

Javascript numbers are always one type: **double (64-bit floating point)**.

Numbers

Exponential Notation:-

Extra large or extra small numbers can be written with scientific (exponential) notation

```
let y = 123e5;     // The value of y will be 12300000
let z = 123e-5;    // The value of z will be 0.00123
```

Javascript Booleans

- Booleans can only have two values: true or false.
- Booleans are often used in conditional testing.
- The Boolean value of an expression is the basis for all JavaScript comparisons and conditions.

```
<!DOCTYPE html>
<html>
<body>
<p id="msg1"></p>
<p id="msg2"></p>
<p id="msg3"></p>
<script>
let x = 5;
let y = 6;
document.getElementById("msg1").innerHTML = Boolean(10 > 9);
document.getElementById("msg2").innerHTML = 3 > 5;
document.getElementById("msg3").innerHTML = x == y;
</script>
</body>
</html>
```

Output

true

false

false

Javascript Undefined

- In JavaScript, a variable without a value, has the value `undefined`. The type is also `undefined`.
- You can use the JavaScript `typeof` operator to find the type of a JavaScript variable. It returns the type of a variable or an expression.

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

<p id="msg"></p>

<script>
let job;  // Value is undefined, type is also undefined.


document.getElementById("msg").innerHTML = job + "<br>" + typeof job;
</script>

</body>
</html>
```

Output

```
undefined
undefined
```

Javascript Undefined

Also, any variable can be emptied, by setting the value to `undefined`. The type will also be `undefined`.

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

<p id="msg"></p>

<script>
let job = "Manager"; // Here, the Value is “Manager”
job = undefined;  // Now, the Value is undefined, type is also undefined

document.getElementById("msg").innerHTML = job + "<br>" + typeof job;
</script>

</body>
</html>
```

Output

```
undefined
undefined
```

Empty Value

An empty string has both a legal value and a type. It is not the same as `undefined`.

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

<p id="msg"></p>

<script>
let job = "";
document.getElementById("msg").innerHTML =
"The value is: " + job + "<br>" + "The type is: " + typeof job;
</script>

</body>
</html>
```

Output

The value is:
The type is: string

Javascript null

In JavaScript, `null` is a special value that represents empty or unknown value.

```
const number = null;

// This means that the number variable is empty.
```

Note: `null` is not the same as NULL or Null.

Javascript Symbol

This data type was introduced in a newer version of JavaScript (from ES2015).

A value having the data type `Symbol` can be referred to as a symbol value.Symbols are **immutable** (cannot be changed) primitive value that is **unique**.

If the same code snippet is used in various programs, then it is better to use `Symbols` in the object key. It's because you can use the same key name in different codes and avoid duplication issues.

Javascript Symbol

Though `id1`and `id2` both contain `'123'`, they are different as they are of the `Symbol` type.

```
// two symbols with the same description

const id1 = Symbol("123");

const id2 = Symbol('123');

  console.log(id1); // Symbol(123)

  console.log(id2.description); // 123
```

Every symbol is unique. Two Symbols even with the same key values are not same.

```
var occupation=Symbol('engineer');
var occupation=Symbol();
```

occupation===occupation // returns false.

Non-primitive/Object Datatypes

Objects

- JavaScript objects are written with curly braces {}.
- Object properties are written as name:value pairs, separated by commas.

```
<!DOCTYPE html>
<html>
<body>
<p id="msg"></p>
<script>
const applicant = {
  firstName : "John",
  lastName  : "Joseph",
  age     : 50,
  qualification  : "BTech"
};
document.getElementById("msg").innerHTML =
applicant.firstName + " is " + applicant.age + " years old.";
</script>
</body>
</html>
```

Output

John is 50 years old.

Arrays

JavaScript arrays are written with square brackets. Array items are separated by commas.

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

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

<p id="msg"></p>

<script>
const jobs = ["Manager","Accountant","Receptionist"];

document.getElementById("msg").innerHTML =
jobs[0]+"<br>"+jobs[1]+"<br>"+jobs[2];
</script>

</body>
</html>
```

Output

Manager
Accountant
Receptionist

Javascript Date

- JavaScript Date Objects let us work with dates:
- Date objects are created with the new Date() constructor.
- The JavaScript date object can be used to get year, month and day.

new Date() without arguments, creates a date object with the current date and time:

```
<!DOCTYPE html>
<html>
<body>
<p id="msg"></p>
<script>
const d = new Date();
document.getElementById("msg").innerHTML = d;
</script>
</body>
</html>
```

Output

Fri Dec 29 2023 17:48:05 GMT+0530 (India Standard Time)

Thank You