JAVASCRIPT DATA TYPES

JavaScript has 8 Datatypes

1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

The Object Datatype

The object data type can contain:

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

```
// Numbers:
let length = 16;
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];
```

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

Undefined

The type Undefined has one value, undefined. This is the value given to variables declared without initializers.

```
let numberOfPets = 5
let age
numberOfPets
age
```

Null

The type Null has one value, null. Technically you can use this value for whatever you want, but conventionally people use it as a way to emphatically and explicitly mean there's really no value.

```
let supervisor = "None"
let supervisor = null
let supervisor = undefined
```

Boolean

The two boolean values are true and false. Operators producing boolean values are && ("and also"), || ("or else"), and ! ("not"). Try:

```
let x = 42
let y = -1
let bothPositive = x > 0 && y > 0
let atLeastOneNegative = x < 0 || y < 0
```

Number

```
42              // Forty-two, and example of a "whole number" or "integer"
3.15            // Three and 15 one-hundredths
1_000_000_000   // It's fine to use underscores in your numbers
2.4434634E9     // 2.443634 times 10 to the 9th
7.538E-23       // 7.538 times 10 to the -23rd
388E12          // 388 times 10 to the 12th

0x3F82          // A hexadecimal literal (base-16), equal to 16258
0b111011010101  // A binary literal (base-2), equal to 3797
```

```
0o3571          // An octal literal (base-8), equal to 1913
```

BigInt

All JavaScript numbers are stored in a a 64-bit floating-point format.

JavaScript BigInt is a new datatype (ES2020) that can be used to store integer values that are too big to be represented by a normal JavaScript Number.

Example

```
let x = BigInt("123456789012345678901234567890")J;
```

JavaScript Arrays

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p>Array indexes are zero-based, which means the first item is [0].</p>
<p id="demo"></p>
<script>
const cars = ["Saab","Volvo","BMW"];
document.getElementById("demo").innerHTML = cars[0];
</script>
</body>
</html>
```

JavaScript Objects

JavaScript objects are written with curly braces {}.

Object properties are written as name:value pairs, separated by commas.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>

<script>
const person = {
  firstName : "John",
  lastName  : "Doe",
  age     : 50,
  eyeColor  : "blue"
};

document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
</body>
</html>
```

The typeof Operator

You can use the JavaScript typeof operator to find the type of a JavaScript variable.

The typeof operator returns the type of a variable or an expression:

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

<h1>JavaScript Operators</h1>
<h2>The typeof Operator</h2>
<p>The typeof operator returns the type of a variable or an expression.</p>

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

<script>
document.getElementById("demo").innerHTML =
typeof "" + "<br>" +
```

```
typeof "John" + "<br>" +
typeof "John Doe";
</script>
</body>
</html>
```