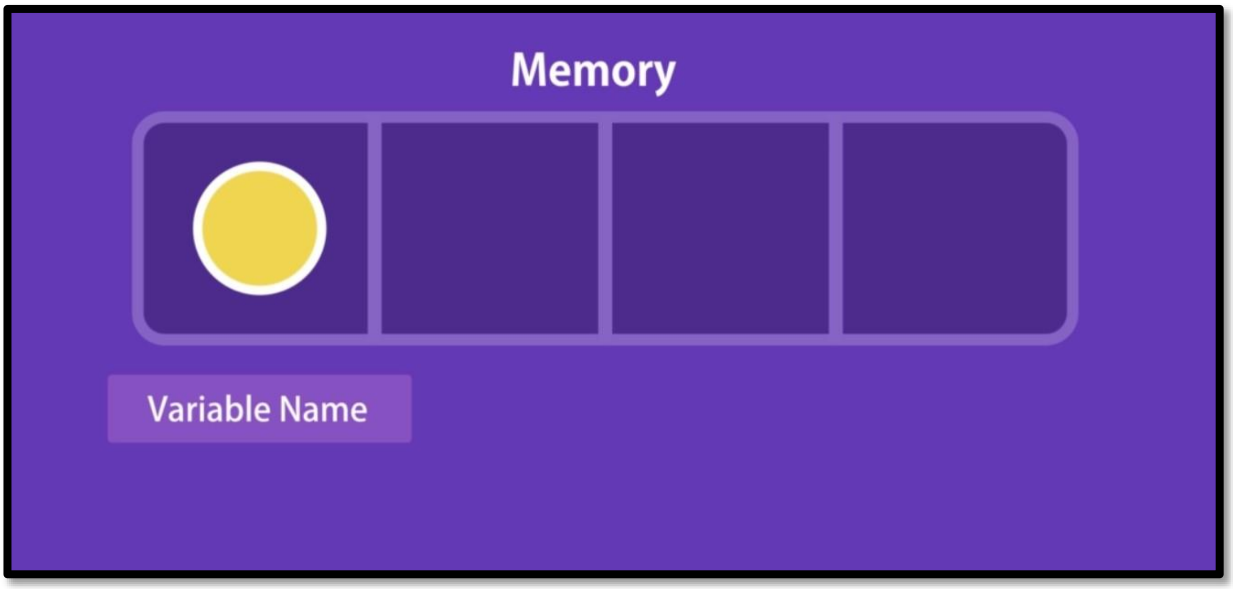
Values

Chunks of information we want to work with that information(data) can be of different types e.g.

```
"Tic Tac Toe", 9, "#board"
```

Variables



In programming language, we use variables to store data temporarily in a computer’s memory. So, we store the data somewhere and give that memory location a name. With this name we can read the data at a given location in the future.

A variable is like a box. What people keep inside the box is the value that we assign to a variable. The label that we put on the box is the name of the variable.

Before ES6 we used the var keyword to declare a variable but there are issues with the var keyword. So, the best practice is to use let keyword to declare a variable.

```
let name;
console.log(name); // undefined
```

By default the variable takes the value as undefined in Javascript.

```
let name = 'Anubhav';
console.log(name);
```

Rules for creating the variables

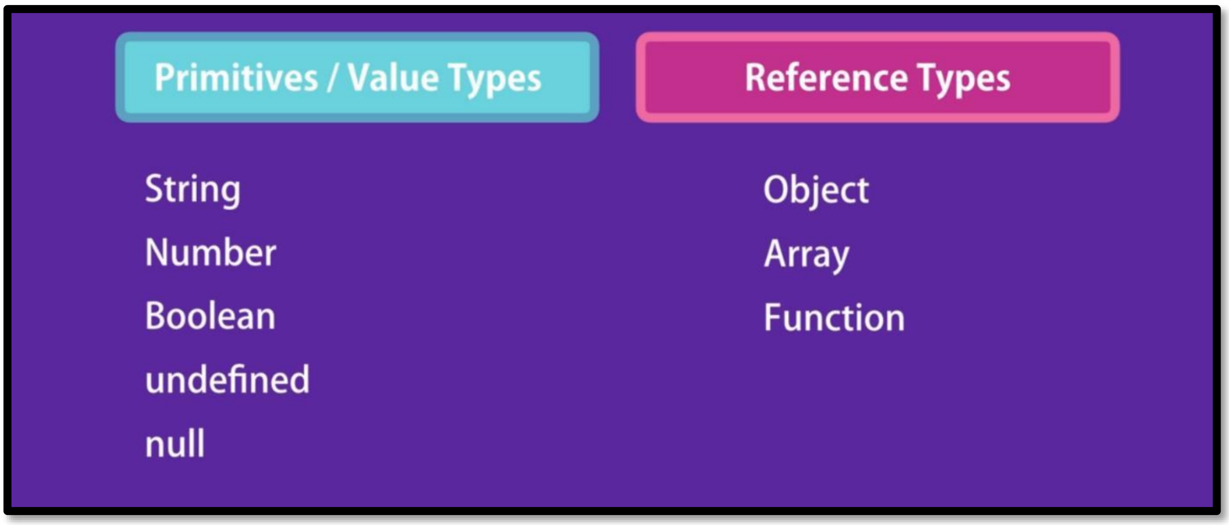
1. Variable name cannot be a reserved keyword.
2. Variable name should be a meaningful.
3. Variable name cannot start with a number e.g. 1name
4. Variable name cannot contain a space or a hyphen.
5. We can use the camel notation for multiple words e.g. firstName
6. Variable are case sensitive i.e. firstName is different from FirstName

Constants

The value of a constant can not be changed.

```
const interestRate = 4.5;
interestRate = 3;// error
```

In Javascript we have two categories of types



Primitives / Value Types	Reference Types
String	Object
Number	Array
Boolean	Function
undefined	
null	

Primitive Types

```
let name = 'Anubhav Gupta'; // String
let age = 15; Number
let isApproved = true; // Boolean
let firstName;
console.log(firstName); // undefined
let lastName = null; // null
```

Null vs Undefined

Null

Null means explicitly nothing or intentionally absence of any value e.g.

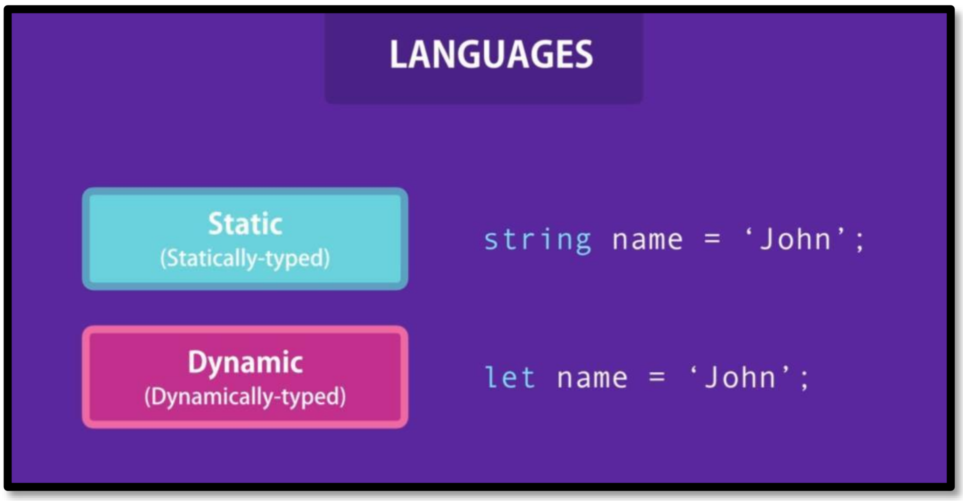
```
let user = null;
```

Undefined

Undefined is the data type that is not yet defined. Undefined variable is simply just we declared but without assigning a value. So, we can say undefined means empty value e.g.

```
let firstName;
```

Dynamic Typing


LANGUAGES

Static (Statically-typed)

```
string name = 'John';
```

Dynamic (Dynamically-typed)

```
let name = 'John';
```

When we create the new variable, we do not manually define the data type of the value that it contains. Javascript automatically determines the data type of its value when it is stored into a variable. In Javascript it is the value that has the data type not the variable.

Language can be static or it can be dynamic but Javascript is dynamic. In static language, when we declare a variable the type of that variable can't be changed but in dynamic language type can be changed even after it's declaration e.g.

```
let name = 'Anubhav';
name = 36;
console.log(name);
```

Reference Types

Objects

When we are dealing with multiple related variable, we can put those variables inside an object.

```
let person = { name: 'Anubhav', age: 30 }

console.log(person); // { name: 'Anubhav', age: 30 }
```

To update the values we can use

1. Dot Notation `person.name = 'Nanu';`

2. Bracket Notation `person['name'] = 'Nanu';`

Dot Notation vs Bracket Notation

Dot notation is concise. So, default choice should be Dot Notation. Sometimes we do not know the name of the target property until the runtime. At that time, we use the Bracket Notation.

Arrays

Sometimes our application deals with the list of the objects e.g. the list of products in the shopping cart. In such situations we use the arrays to store the list of objects.

```
let selectedColors = []; // Empty Array
let selectedColors = ['Red', 'Orange', 'Pink'];
console.log(selectedColors);
```

```
(3) ['Red', 'Orange', 'Pink']
0: "Red"
1: "Orange"
2: "Pink"
```

Note that each element has an index value which determines the position of that element in the array e.g.

```
console.log(selectedColors[0]); // Red
```

As we know, Javascript is a dynamic typed language. We can expand the array.

```
selectedColors[3] = 'Green';
```

Unlike other languages, we can heterogenous type of elements in an array. e.g.

```
let selectedColors = ['Red', 'Orange', 'Pink'];
selectedColors[3] = 123;
selectedColors[4] = true;
console.log(selectedColors); // [ 'Red', 'Orange', 'Pink', 123, true ]
```

Functions

Functions are the building blocks of the Javascript. Function is a set of statements that performs the particular task.

```
function greet()
{
      console.log('Hello World');
}

greet(); //Hello World
```

```
function greet(name) // parameter
{
	console.log('Hello ' + name);
}

greet('John'); //Hello John // argument

function greet(name, lastName)
{
	console.log('Hello ' + name + ' ' + lastName);
}

greet('John'); //Hello John undefined

function square(num)
{
	return num * num;
}

let ans = square(2);
console.log(ans); //4
```

Arrays as Objects

Technically, Arrays are objects.

```
console.log(typeof selectedColors); //object
```

We can access all the properties of an object using the dot notation.

Type Coercion

Type coercion in JavaScript refers to the automatic conversion of one data type to another when an operation requires it.

```
console.log('4' + 2 ) // String 42
console.log('5' * 2); // Number 10
console.log('5' - 2); // Number 3
console.log('25' / 5); // Number 5
```

Type Conversion

```
console.log(String(100)); // string 100

console.log(String(null)); // string null

console.log(String(undefined)); // string undefined

console.log(String(true)); // string true

console.log((123).toString()); // string 123

console.log(null.toString()); // string 100

console.log(Number(false)); // Number 0

console.log(Number(true)); // Number 1
console.log(Number('value')); // NaN

console.log(Number('25')); // Number 25

console.log(parseInt('12345falak')); // Number  12345

console.log(parseInt('   12345falak')); // Number  12345

console.log(parseInt('chandni12345falak')); // NaN

console.log(parseInt('123.45falak')); // Number  123

console.log(parseFloat('123.45falak')); // Number  123.45

let a = '12345falak';
let b = +a;
console.log(b); // NaN

console.log(Boolean(TRUTHY_VALUE)) // true

console.log(Boolean(FALSY_VALUE)) // false
```