

# Introduction

## Assignment Operators

An assignment operator assigns a value to its left operand based on the value of its right operand. Here are some of them:

- += addition assignment
- -= subtraction assignment
- \*= multiplication assignment
- /= division assignment

```
let number = 100;

// Both statements will add 10
number = number + 10;
number += 10;

console.log(number);
// Prints: 120
```

## String Interpolation

String interpolation is the process of evaluating string literals containing one or more placeholders (expressions, variables, etc).

It can be performed using template literals: `text ${expression} text` .

```
let age = 7;

// String concatenation
'Tommy is ' + age + ' years old.';

// String interpolation
`Tommy is ${age} years old.`;
```

## Variables

Variables are used whenever there's a need to store a piece of data. A variable contains data that can be used in the program elsewhere. Using variables also ensures code re-usability since it can be used to replace the same value in multiple places.

```
const currency = '$';
let userIncome = 85000;

console.log(currency + userIncome + ' is
more than the average income.');
```

// Prints: \$85000 is more than the average income.

## Undefined

`undefined` is a primitive JavaScript value that represents lack of defined value. Variables that are declared but not initialized to a value will have the value `undefined`.

```
var a;

console.log(a);
// Prints: undefined
```

## Learn Javascript: Variables

A variable is a container for data that is stored in computer memory. It is referenced by a descriptive name that a programmer can call to assign a specific value and retrieve it.

```
// Examples of variables
let name = "Tammy";
const found = false;
var age = 3;
console.log(name, found, age);
// Prints: Tammy false 3
```

## Declaring Variables

To declare a variable in JavaScript, any of these three keywords can be used along with a variable name:

- `var` is used in pre-ES6 versions of JavaScript.
- `let` is the preferred way to declare a variable when it can be reassigned.
- `const` is the preferred way to declare a variable with a constant value.

```
var age;
let weight;
const numberOfFingers = 20;
```

## Template Literals

Template literals are strings that allow embedded expressions, `${expression}`. While regular strings use single `'` or double `"` quotes, template literals use backticks instead.

```
let name = "Codecademy";
console.log(`Hello, ${name}`);
// Prints: Hello, Codecademy

console.log(`Billy is ${6+8} years old.`);
// Prints: Billy is 14 years old.
```

## let Keyword

`let` creates a local variable in JavaScript & can be re-assigned. Initialization during the declaration of a `let` variable is optional. A `let` variable will contain `undefined` if nothing is assigned to it.

```
let count;
console.log(count); // Prints: undefined
count = 10;
console.log(count); // Prints: 10
```

## const Keyword

A constant variable can be declared using the keyword `const`. It must have an assignment. Any attempt of re-assigning a `const` variable will result in JavaScript runtime error.

```
const numberOfColumns = 4;
numberOfColumns = 8;
// TypeError: Assignment to constant
variable.
```

## String Concatenation

In JavaScript, multiple strings can be concatenated together using the `+` operator. In the example, multiple strings and variables containing string values have been concatenated. After execution of the code block, the `displayText` variable will contain the concatenated string.

```
let service = 'credit card';
let month = 'May 30th';
let displayText = 'Your ' + service + '
bill is due on ' + month + '.';

console.log(displayText);
// Prints: Your credit card bill is due on
May 30th.
```

## console.log()

The `console.log()` method is used to log or print messages to the console. It can also be used to print objects and other info.

```
console.log('Hi there!');
// Prints: Hi there!
```

## JavaScript

JavaScript is a programming language that powers the dynamic behavior on most websites. Alongside HTML and CSS, it is a core technology that makes the web run.

## Methods

Methods return information about an object, and are called by appending an instance with a period `.`, the method name, and parentheses.

```
// Returns a number between 0 and 1  
Math.random();
```

## Built-in Objects

Built-in objects contain methods that can be called by appending the object name with a period `.`, the method name, and a set of parentheses.

```
Math.random();  
// 👉 Math is the built-in object
```

## Numbers

Numbers are a primitive data type. They include the set of all integers and floating point numbers.

```
let amount = 6;  
let price = 4.99;
```

## String `.length`

The `.length` property of a string returns the number of characters that make up the string.

```
let message = 'good nite';  
console.log(message.length);  
// Prints: 9  
  
console.log('howdy'.length);  
// Prints: 5
```

## Data Instances

When a new piece of data is introduced into a JavaScript program, the program keeps track of it in an instance of that data type. An instance is an individual case of a data type.

## Booleans

Booleans are a primitive data type. They can be either true or false .

```
let lateToWork = true;
```

### Math.random()

The `Math.random()` method returns a floating-point, random number in the range from 0 (inclusive) up to but not including 1.

```
console.log(Math.random());  
// Prints: 0 - 0.9999999999999999
```

### Math.floor()

The `Math.floor()` function returns the largest integer less than or equal to the given number.

```
console.log(Math.floor(5.95));  
// Prints: 5
```

## Single Line Comments

In JavaScript, single-line comments are created with two consecutive forward slashes `//` .

```
// This line will denote a comment
```

## Null

Null is a primitive data type. It represents the intentional absence of value. In code, it is represented as `null` .

```
let x = null;
```

## Strings

Strings are a primitive data type. They are any grouping of characters (letters, spaces, numbers, or symbols) surrounded by single quotes ' or double quotes " .

```
let single = 'Wheres my bandit hat?';  
let double = "Wheres my bandit hat?";
```

## Arithmetic Operators

JavaScript supports arithmetic operators for:

- + addition
- - subtraction
- \* multiplication
- / division
- % modulo

```
// Addition  
5 + 5  
  
// Subtraction  
10 - 5  
  
// Multiplication  
5 * 10  
  
// Division  
10 / 5  
  
// Modulo  
10 % 5
```

## Multi-line Comments

In JavaScript, multi-line comments are created by surrounding the lines with /\* at the beginning and \*/ at the end. Comments are good ways for a variety of reasons like explaining a code block or indicating some hints, etc.

```
/*  
The below configuration must be  
changed before deployment.  
*/  
  
let baseUrl =  
'localhost/taxwebapp/country';
```

## Remainder / Modulo Operator

The remainder operator, sometimes called modulo, returns the number that remains after the right-hand number divides into the left-hand number as many times as it evenly can.

```
// calculates # of weeks in a year, rounds  
down to nearest integer
```

```
const weeksInYear = Math.floor(365/7);
```

```
// calculates the number of days left over  
after 365 is divided by 7
```

```
const daysLeftOver = 365 % 7 ;
```

```
console.log("A year has " + weeksInYear +  
" weeks and " + daysLeftOver + " days");
```

 [Print](#)    [Share](#) ▼