Introduction to JavaScript: Introduction

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 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?";
```

JavaScript 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 incomeCurrency = '$';
let userIncome = 85000;

console.log(incomeCurrency + userIncome + ' is more
than the average income.');
// Prints: $85000 is more than the average income.
```

JavaScript Numbers

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

```
let x = 2;
let y = 2.00;
```

JavaScript Booleans

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

```
let lateToWork = true;
```

JavaScript Libraries

JavaScript libraries contain methods that you can call by appending the library name with a period `.`, the method name, and a set of parentheses.

```
Math.random();
// ☝ Math is the library
```

Math.random()

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

```
console.log(Math.random());
// Output: 0 - 0.9
```

String.length

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

```
let x = 'good nite~';
console.log(x.length);
// Expected output: 10

console.log('howdy'.length);
// Expected output: 5
```

JavaScript 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();
```

JavaScript 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);
// output: Your credit card bill is due on May 30th.
```

Math.floor()

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

```
let number = 15.95;
console.log(Math.floor(number));  // 15

number = -99.5 ;
console.log(Math.floor(number));  // -100
```

Single Line Comments

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

```
let score; // This is a comment.
```

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';
```

`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.
```

JavaScript String Interpolation

String interpolation is the process of evaluating string literals containing one or more placeholders (expressions, variables, etc). The two main ways to interpolate strings are with:

- String concatenation: `"string" + expression + "string"`
- Template literals: `` `text ${expression} text` ``

```
let age = 7;

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

// Template literal with interpolation
`Tommy is ${age} years old.`;
```

JavaScript Null

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

```
let x = null;
```

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
```

`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); // Logs undefined in console
count = 10;
console.log(count); // Logs 10 in console
```

JavaScript 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); // undefined
let b;
console.log(b); // undefined
```

Assignment Operators

In JavaScript, the addition assignment operator (`+=`) can be used to add the value on the right hand side to the existing value & assign it to the variable. The addition assignment operator is a shorthand for `variableName = variableName + value`. Here are all of them:

- `+=` addition assignment
- `-=` Subtraction assignment
- `*=` multiplication assignment
- `/=` division assignment

```
let number = 100;

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

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

JavaScript String Interpolation

String interpolation is the process of evaluating string literals containing one or more placeholders (expressions, variables, etc). The two main ways to interpolate strings are with:

- String concatenation: `"string" + expression + "string"`
- Template literals: `` `text ${expression} text` ``

```
let age = 7;

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

// Template literal with interpolation
`Tommy is ${age} years old.`;
```

JavaScript Template Literals

In JavaScript, template literals are strings that allow embedded expressions (`${expression}`). While regular strings use single (`'`) or double (`"`) quotes, template literals use backticks instead. Take a look at the code block for examples.

```
// Syntax:
`string text ${expression} more string text`

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

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