Hojas de trucos / **Aprende JavaScript: Fundamentos**

codecademy

Introducción

Operadores de asignación

Un operador de asignación asigna un valor a su operando izquierdo en función del valor de su operando derecho. Algunos ejemplos son:

- += asignación de adición
- -= asignación de resta
- *= tarea de multiplicación
- /= asignación de división

```
sea número = 100 ;

// Ambas expresiones sumarán 10
número = número + 10 ;
número += 10 ;

console.log ( número ) ;
// Impresiones: 120
```

Interpolación de cadenas

La interpolación de cadenas es el proceso de evaluar literales de cadena que contienen uno o más marcadores de posición (expresiones, variables, etc.).
Puede realizarse utilizando plantillas literales: text ${expression} text .

```
sea edad = 7 ;

// Concatenación de cadenas
'Tommy tiene ' + edad + ' años.' ;

// Interpolación de cadenas
` Tommy tiene ${ age } años. ` ;
```

Variables

Las variables se utilizan cuando se necesita almacenar un dato. Una variable contiene datos que se pueden usar en otras partes del programa. El uso de variables también garantiza la reutilización del código, ya que se puede usar para reemplazar el mismo valor en varios lugares.

```
constante moneda = '$' ;
sea userIncome = 85000 ;

console.log ( currency + userIncome + ' es
mayor que el ingreso promedio. ' ) ;
// Impresiones: 85.000 dólares es más que
el ingreso promedio.
```

Indefinido

`undefined` es un valor primitivo de JavaScript que representa la falta de un valor definido. Las variables que se declaran pero no se inicializan con un valor tendrán el valor `undefined` .

```
variable a ;

console.log ( a ) ;
// Imprime: indefinido
```

Aprende 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);

// calcuates the number of days left over after 365 is divded by 7
const daysLeftOver = 365 % 7 ;

console.log ( " Un año tiene " + weeksInYear + " semanas y " + daysLeftOver + " días" ) ;
```

Imprimir Compartir