Variables JS

La mayoría de aplicaciones en Javascript necesitan trabajar con cierta información. Aquí dos ejemplos:

- Amazon - La información que contiene son los productos, usuarios, carrito...
- WhatsApp - La información que contiene son usuarios, mensajes...

Las variables se usan para guardar dicha información.

La Variable

La variable es el nombre de la "caja" en la que podemos guardar nuestra información, pueden contener productos, usuarios, y cualquier otro dato.

Para crear una variable en Javascript usamos la palabra ***let***. Después iría el nombre de nuestra variable, vamos a definir nuestra primera variable.

```
let product;
```

Ahora podemos usar esta variable y asignarle un valor con el operador =.

```
let product;
product = 'Tomatoes';
console.log(product); //muestra el contenido de la variable
```

> **var** o **let** → En scripts antiguos, antes de ES6, puedes encontrar la inicialización de las variables con **var**. En nuestro caso **let** es lo más parecido, pese tener diferencias importantes que veremos en otro momento.

Podemos ser aún más concisos a la hora de declarar una variable y hacerlo en una única línea.

```
let product = 'Tomatoes';
console.log(product); //muestra el conetenido de la variable
```

También podemos declarar varias variables en una única línea.

```
let product = 'Tomatoes', price = 5, color = 'red';
```

Personalmente me gusta declarar cada variable en una línea diferente por la facilidad que supone a la hora de leer el código.

```
let product = 'Tomatoes';
let price = 5;
let color = 'red';
```

Podemos seguir cambiando el valor de nuestras variables, vamos a ver como nuestros **tomatoes** pasan a ser **potatoes**.

```
let product;

product = 'Tomatoes';
```

```
product = 'Potatoes'; //Cambio de valor

console.log(product);
```

Además de cambiar el valor podemos copiar el valor de una variable a otra.

```
let product = 'Tomatoes';

let myProduct;

//Copiamos Tomatoes desde product a myProduct
product = myProduct;

//Ahora las dos variables tienen el mismo valor
console.log(product);
console.log(myProduct);
```

Las variables solo deberían declararse una vez, repetir la declaración de una variable es un error.

```
let product = 'Tomatoes'

//Repetir el 'let' produce un error
let product = 'Potatoes'// SyntaxError: 'product' has already be
```

Nombres de variables

A la hora de poner un nombre a nuestras variables solamente tenemos dos limitaciones:

- El nombre de la variable solo debe contener letras, dígitos y los símbolos $ y _.
- El primer carácter no debe ser un dígito.

```
let product;
let test1;
```

Nombres que no son válidos.

```
let 1product; // cannot start with a digit

let my-product; // hyphens '-' aren't allowed in the name
```

Nombres reservados.

```
let let = 'Hola'; // can't name a variable "let", error!
let return = 'Hola'; // also can't name it "return", error!
```

Siempre que declaremos una variable deberíamos usar nombres descriptivos que hagan referencia al valor que contiene dicha variable.

```
let userName = 'Alberto';

let userAge = 34;
```

Scope

Variables declaradas fuera de cualquier función se llaman *globales*. Las variables globales son visibles desde cualquier función (a menos que se oculten por variables locales).

Es una buena práctica reducir el uso de variables globales. El código moderno tiene pocas o ninguna variable global. La mayoría de las variables residen en sus

funciones. Aunque a veces puede ser útil almacenar datos a nivel de proyecto.

Variables locales

Una variable declarada dentro de una función solo es visible dentro de esa función. Dicho de otro modo la función genera un scope propio.

```
function showPokemon() {
  let pokemon = "Pikachu"; // variable local
  console.log(pokemon);
}

showPokemon(); // Pikachu

console.log(pokemon); // <-- Error! La variable es local para es
```

Variables externas

Una función también puede acceder a una variable externa. Puedes acceder a variables globales dentro del scope de una función.

```
let pokemonExt = 'Charmander';

function showPokemon() {
  console.log(pokemonExt);
}

showPokemon(); // Charmander
```

La función tiene acceso completo a la variable externa. Puede modificarlo también.

```
let pokemonExt = 'Charmander';

function showPokemon() {
  pokemonExt = "Squirtle"; // (1) Cambió la variable externa

  let message = 'Hola, ' + pokemonExt;
  console.log(message);
}

console.log( pokemonExt ); // Charmander antes de llamar la fun

showPokemon();

console.log( pokemonExt ); // Squirtle, el valor fué modificado
```

Si una variable con el mismo nombre se declara dentro de la función, le hace *sombra* [shadowing] a la externa. Por ejemplo, en el siguiente código, la función usa el pokemonExt local. El exterior se ignora.

```
let pokemonExt = 'Charmander';

function showPokemon() {
  let pokemonExt = "Squirtle"; // declara variable local

  console.log(pokemonExt); // Squirtle
}

// la función crea y utiliza su propia variable local pokemonExt
showPokemon();

console.log( pokemonExt );
// Charmander, se mantiene, la función no accedió a la variable
```

Crear variables `var` `let` `const`

En JavaScript, `var`, `let` y `const` son tres palabras clave que se utilizan para declarar variables. Aunque en un primer momento pueden parecer similares, hay algunas diferencias importantes entre ellas que debes tener en cuenta al elegir cuál utilizar.

Declaración con `var`

`var` es la forma más antigua de declarar variables en JavaScript y se utiliza desde la primera versión del lenguaje. Una variable declarada con `var` es accesible en todo el ámbito en el que se encuentra, ya sea una función o el ámbito global.

```
var name = 'John';

function sayHi() {
  console.log(`Hi, ${name}`);
}

sayHi(); // "Hi, John"
```

En este ejemplo, la variable `name` es accesible desde dentro de la función `sayHi`.

Declaración con `let`

`let` es una forma más reciente de declarar variables en JavaScript y fue introducida en la versión 6 de ECMAScript. A diferencia de `var`, una variable declarada con `let` tiene un ámbito de bloque, es decir, sólo es accesible dentro del bloque en el que se encuentra.

```
let name = 'John';

if (name === 'John') {
  let age = 25;
```

```
  console.log(`${name} is ${age} years old`);
}

console.log(age); // ReferenceError: age is not defined
```

En este ejemplo, la variable `age` sólo es accesible dentro del bloque `if`. Si se intenta acceder a ella fuera del bloque, se lanzará un error.

Declaración con `const`

La palabra clave `const` se utiliza para declarar variables y es similar a `let`, con algunas diferencias importantes. Una variable declarada con `const` tiene un alcance de bloque, lo que significa que está disponible sólo dentro del bloque en el que se declara. Además, `const` no permite ni reasignar ni redeclarar variables. Esto significa que una vez que se ha asignado un valor a una variable `const`, ésta no se puede cambiar.

Es importante tener en cuenta que, aunque una variable `const` no puede ser reasignada, eso no significa que su valor sea inmutable. Si la variable `const` es un objeto o un array, es posible modificar sus propiedades o elementos, pero no se puede reasignar a otro objeto o array.

Aquí tienes un ejemplo de una variable declarada con `const`:

```
const z = 5;
console.log(z); // 5

z = 10; // Uncaught TypeError: Assignment to constant variabl
e.

const z = 15; // Uncaught SyntaxError: Identifier 'z' has alr
```

```
eady been declared
```

En este caso, la variable `z` se declara con `const` y se inicializa con el valor 5. Luego, se intenta reasignar con el valor 10, pero esto produce un error ya que `const` no permite reasignar variables. Finalmente, se intenta redeclarar la variable `z` con el valor 15, pero esto también produce un error ya que `const` no permite redeclarar variables.

Reto coding

Crea una variable llamada `myFavoriteHero`, asigna el valor `Hulk` a ella.

Crea una variable llamada
`x`, asigna el valor `50` a ella.

Crea una variable llamada
`h` con el valor `5` y otra `y` con el valor `10`.

Crea una otra variable `z` y asignale el valor de `h + y`.

Dado el siguiente javascript, cambia el valor de la propiedad `age` a `250`.

```
const character = {name: 'Jack Frost', age: 10};
```

Declara 3 variables con los nombres y valores siguientes:

```
firstName = 'Jon';
lastName = 'Snow';
age = 24;
```

Guarda los valores en `sentence` con la siguiente estructura:

```
Soy Jon Snow, tengo 24 años y me gustan los lobos.
```

Dado el siguiente javascript, guarda en una variable la suma del precio de ambos juguetes.

```
const toy1 = {name: 'Buss myYear', price: 19};
const toy2 = {name: 'Rallo mcKing', price: 29};
```