JS

Docs_

Trucos y utilidades

1. **Uso de Array.isArray() para verificar si una variable es un arreglo:**

```
let miVariable = [1, 2, 3];
console.log(Array.isArray(miVariable)); // Imprime: true
```

2. **Expresiones regulares para validación de datos:**

```
let patronEmail = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
console.log(patronEmail.test("usuario@example.com")); // Imprime: true
```

3. **Usar setTimeout() y setInterval() para ejecutar código después de un retraso o repetidamente:**

```
setTimeout(() => console.log("Hola después de 2 segundos"), 2000);
let intervalo = setInterval(() => console.log("Repetir cada 3 segundos"),
3000);
// Para detener el intervalo: clearInterval(intervalo);
```

4. **Clonación de objetos o arrays usando el operador spread**:

```
const original = { a: 1, b: 2 };
const copia = { ...original };
console.log(copia); // Imprime: { a: 1, b: 2 }
```

Docs_ Generalidades Curiosidades

5. **Uso de etiquetas en bucles for y while para un control más detallado**:

```
loopExterno: for (let i = 0; i < 5; i++) {
  for (let j = 0; j < 5; j++) {
    if (i === 2 && j === 2) {
     break loopExterno;
    }
  console.log(i, j);
  }
}
```

6. **Uso de try...catch para manejar errores**:

```
try {
// Intenta ejecutar este código
} catch (error) {
// Maneja cualquier error que ocurra
  console.log(error);
}
```

7. **Usar el operador || para asignar valores predeterminados**:

```
function saludar(nombre) {
  nombre = nombre || 'Invitado';
  console.log('Hola ' + nombre);
}
saludar(); // Imprime: Hola Invitado
saludar('Pedro'); // Imprime: Hola Pedro
```

8. **Operadores de comparación estricta (=== y !==)**:

```

let valor1 = 5;
let valor2 = '5';
console.log(valor1 == valor2);  // true, comparación abstracta
console.log(valor1 === valor2); // false, comparación estricta
```

9. **Usar Function.prototype.apply() para pasar un arreglo de argumentos a una función**:

```
function suma(x, y, z) {
   return x + y + z;
}

let numeros = [1, 2, 3];
console.log(suma.apply(null, numeros)); // Imprime: 6
```

Docs_ Generalidades Curiosidades

10. **Usar document.querySelector() y document.querySelectorAll() para la manipulación del DOM**:

```
let primerDiv = document.querySelector('div'); // Selecciona el primer div
let todosLosDivs = document.querySelectorAll('div'); // Selecciona todos
los divs
```

11. **Usar document.createElement() y appendChild() para crear y agregar elementos al DOM**:

```
let nuevoDiv = document.createElement('div');
nuevoDiv.textContent = '¡Hola Mundo!';
document.body.appendChild(nuevoDiv);
```

12. **Usar localStorage y sessionStorage para almacenar datos localmente**:

```
localStorage.setItem('clave', 'valor'); // Persiste entre sesiones
sessionStorage.setItem('clave', 'valor'); // Persiste solo en la sesión
actual
```

13. **Desestructuración para acceder a propiedades de objetos y arrays:** La desestructuración te permite extraer múltiples propiedades de un objeto o elementos de un array en una sola línea. Por ejemplo, const {nombre, edad} = persona; para objetos, o const [primero, segundo] = array; para arrays.

```
let numero = 5;
console.log("El número es:", numero); // Imprime: El número es: 5
```

14. **Desestructuración:**

```
const persona = { nombre: "Juan", edad: 30 };
const { nombre, edad } = persona;
console.log(nombre); // Imprime: Juan
```

```
const numeros = [1, 2, 3];
const [primero, segundo] = numeros;
console.log(segundo); // Imprime: 2
```

15. **Valores predeterminados para parámetros**:

```
function saludar(nombre = 'Invitado') {
  console.log('Hola ' + nombre);
}
saludar(); // Imprime: Hola Invitado
saludar('Ana'); // Imprime: Hola Ana
```

Docs_ Generalidades Curiosidades

16. **Uso de spread y rest operators (...):** El operador spread es útil para expandir elementos de un array o propiedades de un objeto. El operador rest se utiliza para representar un número indefinido de argumentos como un array.

17. **Función de flecha:**

```
const sumar = (a, b) => a + b;
console.log(sumar(5, 3)); // Imprime: 8
```

18. **Template literals para concatenar strings:** En lugar de usar el operador + para concatenar cadenas, puedes usar los template literals para hacerlo de una manera más legible. Ejemplo: `Hola, mi nombre es ${nombre}`.

19. **Asincronía con async/await:** async/await hace que el trabajo con promesas sea más sencillo y legible. Puedes escribir código asincrónico que parece sincrónico y es más fácil de leer y mantener.

```
async function obtenerDatos() {
  let respuesta = await fetch('https://api.example.com/datos');
  let datos = await respuesta.json();
  console.log(datos);
}
obtenerDatos();
```

20. **Uso de map(), filter(), y reduce() en arrays:** Estas funciones de alto orden son extremadamente útiles para trabajar con arrays. map() te permite transformar cada elemento de un array, filter() te permite seleccionar elementos específicos de un array basados en una condición, y reduce() te ayuda a reducir un array a un solo valor.

```
const numeros = [1, 2, 3, 4, 5];
const dobles = numeros.map(n => n * 2);
console.log(dobles); // Imprime: [2, 4, 6, 8, 10]
```

```
const mayoresQueTres = numeros.filter(n => n > 3);
console.log(mayoresQueTres); // Imprime: [4, 5]
```

```
const suma = numeros.reduce((total, actual) => total + actual, 0);
console.log(suma); // Imprime: 15
```

Docs_ Generalidades Curiosidades

21. **Short-circuit evaluation para asignaciones condicionales**: Puedes usar && y || para asignar valores bajo ciertas condiciones de manera concisa. Por ejemplo, const a = b || 'default'; asigna a a el valor de b si b es "truthy", de lo contrario asigna 'default'.

```
let usuario = null;
let nombreUsuario = usuario || 'Invitado';
console.log(nombreUsuario); // Imprime: Invitado
```

22. **Funciones IIFE (Immediately Invoked Function Expression) para ejecutar funciones al instante:** Las IIFE son funciones que se ejecutan tan pronto como se definen. Son útiles para crear un alcance privado y evitar la contaminación del alcance global.

```
(function() {
  let mensaje = "Hola Mundo";
  console.log(mensaje); // Imprime: Hola Mundo
})();
```

23. **Object.freeze() para hacer inmutables los objetos:** Si quieres prevenir que los objetos sean modificados, puedes usar Object.freeze(objeto) para hacer que sus propiedades sean inalterables**.**

```
const objetoInmutable = Object.freeze({ nombre: "Pedro" });
objetoInmutable.nombre = "Juan"; // Esto no modificará el objeto
console.log(objetoInmutable.nombre); // Imprime: Pedro
```

24. **Uso de operador ternario**: El operador ternario en JavaScript es una forma abreviada de escribir una declaración if-else. En lugar de escribir una declaración if-else completa, puedes utilizar el operador ternario para hacer la misma comparación con menos código. Por ejemplo:

```
// Declaración if-else
if (a > b) {
  console.log("a es mayor que b");
} else {
  console.log("b es mayor que a");
}

// Operador ternario
a > b ? console.log("a es mayor que b") : console.log("b es mayor que a");
```

Docs_ Generalidades Curiosidades

25. **Conversión de tipos de datos**: JavaScript convierte automáticamente los tipos de datos según sea necesario, pero a veces puedes necesitar convertirlos explícitamente. Puedes convertir una cadena en un número entero con parseInt(), o en un número decimal con parseFloat(). Por ejemplo:

```
var cadena = "10";
var numero = parseInt(cadena); // numero es igual a 10
var decimal = parseFloat(cadena); // decimal es igual a 10.0
```

26. **Operador spread**: El operador spread te permite combinar dos arreglos o expandir los elementos de un arreglo en los argumentos de una función. Por ejemplo:

```
var arreglo1 = [1, 2, 3];
var arreglo2 = [4, 5, 6];
var arreglo3 = [...arreglo1, ...arreglo2]; // arreglo3 es igual a [1, 2, 3, 4, 5, 6]
```

```
function sumar(a, b, c) {
  return a + b + c;
}
```

```
var arreglo4 = [1, 2, 3];
var resultado = sumar(...arreglo4); // resultado es igual a 6
```

27. **Operador de cortocircuito**: El operador de cortocircuito te permite simplificar la evaluación de expresiones booleanas. Puedes utilizar el operador && para evaluar una serie de condiciones, y detener la evaluación si una de las condiciones es falsa. Puedes utilizar el operador || para asignar un valor predeterminado a una variable si su valor es nulo o indefinido. Por ejemplo:

```
var resultado = a && b && c; // resultado es igual a c si a, b y c son verdaderos, o de lo contrario es igual a false
var valor = a || "predeterminado"; // valor es igual a a si a tiene un valor distinto de nulo o indefinido, o de lo contrario es igual a "predeterminado"
```

28. Usar Array.from() para crear arreglos desde objetos iterables:

```
let set = new Set([1, 2, 3]);
let array = Array.from(set);
console.log(array); // Imprime: [1, 2, 3]
```

Docs_ Generalidades Curiosidades

29. Uso de Object.assign() para fusionar y clonar objetos:

```
const objeto1 = { a: 1 };
const objeto2 = { b: 2 };
const combinado = Object.assign({}, objeto1, objeto2);
console.log(combinado); // Imprime: { a: 1, b: 2 }
```

30. Utilizar Array.prototype.some() y Array.prototype.every() para pruebas condicionales:

```
const array = [1, 2, 3, 4, 5];
const alMenosUnoEsPar = array.some(num => num % 2 === 0);
console.log(alMenosUnoEsPar); // Imprime: true
```

```
const todosSonPares = array.every(num => num % 2 === 0);
console.log(todosSonPares); // Imprime: false
```

31. Uso de Math para cálculos comunes:

```
console.log(Math.max(1, 3, 5)); // Imprime: 5
console.log(Math.min(1, 3, 5)); // Imprime: 1
console.log(Math.round(1.5)); // Imprime: 2
```

32. Truco para convertir strings a números usando el operador +:

```
let numero = "123";
console.log(+numero); // Imprime: 123
```

33. Encadenamiento opcional (?.) para acceder a propiedades de objetos que pueden no existir:

```
const usuario = { nombre: 'Juan', detalles: { edad: 30 } };
console.log(usuario.detalles?.edad); // Imprime: 30
console.log(usuario.direccion?.calle); // No produce un error, imprime:
undefined
```

34. Uso de fetch para realizar solicitudes HTTP:

```
fetch('https://api.example.com/datos')
.then(response => response.json())
.then(datos => console.log(datos));
```

Docs_ Generalidades Curiosidades

35. Usar new Set() para eliminar elementos duplicados de un array:

const numeros = [1, 2, 2, 3, 4, 4, 5];

const unicos = [...new Set(numeros)];

console.log(unicos); // Imprime: [1, 2, 3, 4, 5]

36. Uso de etiquetas de plantilla para crear funciones de procesamiento de strings:

```
function etiqueta(literales, ...expresiones) {
  let resultado = '';
  literales.forEach((literal, i) => {
  resultado += literal + (expresiones[i] || '');
  });
  return resultado;
}
let cantidad = 10, unidad = 'kg';
console.log(etiqueta`Tengo ${cantidad} ${unidad} de arroz`); // Imprime:
Tengo 10 kg de arroz
```