Objetos predefinidos de JS

Arrays

Creación de Arrays

```
// Array vacío
const arrVacio = [];

// Array con elementos
const frutas = ['Manzana', 'Banana', 'Cereza'];
```

Métodos Comunes

Método	Descripción	Ejemplo
`push(elemento)`	Agrega uno o más elementos al final del array.	`frutas.push('Durazno');`
`pop()`	Elimina el último elemento del array y lo retorna.	`const ultima = frutas.pop();`
`shift()`	Elimina el primer elemento del array y lo retorna.	`const primero = frutas.shift();`
`unshift(elemento)`	Agrega uno o más elementos al inicio del array.	`frutas.unshift('Fresa');`
`map(función)`	Crea un nuevo array con los resultados de la función.	`const mayusculas = frutas.map(f => f.toUpperCase());`
`filter(función)`	Crea un nuevo array con los elementos que cumplen la condición.	`const conA = frutas.filter(f => f.includes('a'));`
`reduce(función, inicial)`	Aplica una función contra un acumulador y reduce el array a un solo valor.	`const total = numeros.reduce((acc, val) => acc + val, 0);`

`find(función)`	Retorna el primer elemento que cumple la condición.	`const banana = frutas.find(f => f === 'Banana');`
`includes(elemento)`	Verifica si el array incluye el elemento.	`frutas.includes('Manzana'); // true`
`slice(inicio, fin)`	Retorna una copia de una parte del array.	`const primeras = frutas.slice(0, 2);`
`splice(indice, cantidad, ...elementos)`	Cambia el contenido del array eliminando o agregando elementos.	`frutas.splice(1, 1, 'Kiwi'); // Remueve 'Banana' y agrega 'Kiwi'`

Iteración

```
// For clásico
for (let i = 0; i < frutas.length; i++) {
  console.log(frutas[i]);
}

// For...of
for (const fruta of frutas) {
  console.log(fruta);
}

// forEach
frutas.forEach(fruta => console.log(fruta));
```

- **Crear:** `const arr = [1, 2, 3];`
- **Agregar:** `arr.push(4);`
- **Iterar:** `arr.forEach(item => console.log(item));`
- **Transformar:** `const dobles = arr.map(x => x * 2);`

Map

Descripción

`Map` es una colección de pares clave-valor donde las claves pueden ser de cualquier tipo, incluidos objetos y funciones.

Usos Comunes

- Almacenar datos donde las claves no son necesariamente cadenas.
- Implementar cachés.
- Contar ocurrencias de elementos.

Creación y Métodos Comunes

```
// Creación de un Map
const mapa = new Map();

// Agregar elementos
mapa.set('nombre', 'Juan');
mapa.set(1, 'uno');
mapa.set({ a: 1 }, 'objeto');

// Obtener valores
console.log(mapa.get('nombre')); // 'Juan'

// Verificar existencia de una clave
console.log(mapa.has(1)); // true

// Tamaño del Map
console.log(mapa.size); // 3

// Eliminar un elemento
mapa.delete('nombre');

// Iteración
mapa.forEach((valor, clave) => {
  console.log(`${clave}: ${valor}`);
});

// Alternativa de iteración
for (const [clave, valor] of mapa) {
  console.log(`${clave}: ${valor}`);
}
```

```
// Limpiar el Map
mapa.clear();
```

- **Crear:** `const mapa = new Map();`
- **Agregar:** `mapa.set('clave', 'valor');`
- **Obtener:** `mapa.get('clave');`
- **Iterar:** `for (const [k, v] of mapa) { console.log(k, v); }`

Set

Descripción

`Set` es una colección de valores únicos, es decir, no permite elementos duplicados.

Usos Comunes

- Eliminar duplicados de un array.
- Verificar la presencia de un elemento de manera eficiente.
- Almacenar colecciones de valores únicos.

Creación y Métodos Comunes

```
// Creación de un Set
const conjunto = new Set();

// Agregar elementos
conjunto.add(1);
conjunto.add('Hola');
conjunto.add({ a: 1 });
conjunto.add(1); // No se agrega, ya existe

// Verificar existencia
console.log(conjunto.has('Hola')); // true

// Tamaño del Set
console.log(conjunto.size); // 3
```

```
// Eliminar un elemento
conjunto.delete(1);

// Iteración
conjunto.forEach(valor => {
  console.log(valor);
});

// Alternativa de iteración
for (const valor of conjunto) {
  console.log(valor);
}

// Limpiar el Set
conjunto.clear();
```

Ejemplo: Eliminar Duplicados de un Array

```
const numeros = [1, 2, 3, 2, 4, 1, 5];
const sinDuplicados = [...new Set(numeros)];
console.log(sinDuplicados); // [1, 2, 3, 4, 5]
```

- **Crear:** `const conjunto = new Set();`
- **Agregar:** `conjunto.add(1);`
- **Verificar:** `conjunto.has(1);`
- **Eliminar Duplicados:** `const unico = [...new Set(arrayConDuplicados)];`

Math

Descripción

El objeto `Math` proporciona propiedades y métodos para constantes matemáticas y funciones.

Propiedades Comunes

Propiedad	Descripción

`Math.PI`	Valor de π (3.1416).
`Math.E`	Base de los logaritmos naturales.
`Math.LN2`	Logaritmo natural de 2.

Métodos Comunes

Método	Descripción	Ejemplo
`Math.abs(x)`	Valor absoluto de `x`.	`Math.abs(-5); // 5`
`Math.ceil(x)`	Redondea `x` hacia arriba.	`Math.ceil(4.2); // 5`
`Math.floor(x)`	Redondea `x` hacia abajo.	`Math.floor(4.8); // 4`
`Math.round(x)`	Redondea `x` al entero más cercano.	`Math.round(4.5); // 5`
`Math.max(...args)`	Retorna el valor máximo entre los argumentos.	`Math.max(1, 3, 2); // 3`
`Math.min(...args)`	Retorna el valor mínimo entre los argumentos.	`Math.min(1, 3, 2); // 1`
`Math.pow(x, y)`	`x` elevado a la potencia `y`.	`Math.pow(2, 3); // 8`
`Math.sqrt(x)`	Raíz cuadrada de `x`.	`Math.sqrt(16); // 4`
`Math.random()`	Número pseudoaleatorio entre 0 (inclusive) y 1 (exclusive).	`Math.random(); // e.g., 0.736`
`Math.floor(Math.random() * 10) + 1`	Número aleatorio entero entre 1 y 10.	`let num = Math.floor(Math.random() * 10) + 1;`

Ejemplos de Uso

```
// Generar un número aleatorio entre 0 y 100
const aleatorio = Math.random() * 100;

// Redondear un número
const redondeado = Math.round(aleatorio);
```

```
// Calcular la potencia
const potencia = Math.pow(2, 5); // 32

// Obtener la raíz cuadrada
const raiz = Math.sqrt(25); // 5
```

- **Número Aleatorio:** `Math.random();`
- **Redondear:** `Math.round(4.7); // 5`
- **Potencia:** `Math.pow(2, 3); // 8`
- **Máximo:** `Math.max(1, 2, 3); // 3`

Date

Descripción

El objeto `Date` maneja fechas y horas en JavaScript.

Creación de Objetos Date

```
const ahora = new Date(); // Fecha y hora actual
const fechaEspecifica = new Date('2024-05-15'); // Fecha es
pecífica
// Año, Mes (0-11), Día, Hora, Minuto, Segundo
const fechaConParametros = new Date(2024, 4, 15, 10, 30,
0);
```

Métodos Comunes

Método	Descripción	Ejemplo
`getFullYear()`	Retorna el año (e.g., 2024).	`fecha.getFullYear(); // 2024`
`getMonth()`	Retorna el mes (0-11).	`fecha.getMonth(); // 4` (Mayo)
`getDate()`	Retorna el día del mes (1-31).	`fecha.getDate(); // 15`
`getDay()`	Retorna el día de la semana (0-6).	`fecha.getDay(); // 3` (Miércoles)
`getHours()`	Retorna la hora (0-23).	`fecha.getHours(); // 10`

`getMinutes()`	Retorna los minutos (0-59).	`fecha.getMinutes(); // 30`
`getSeconds()`	Retorna los segundos (0-59).	`fecha.getSeconds(); // 0`
`setFullYear(año)`	Establece el año.	`fecha.setFullYear(2025);`
`setMonth(mes)`	Establece el mes (0-11).	`fecha.setMonth(5); // Junio`
`setDate(día)`	Establece el día del mes.	`fecha.setDate(20);`
`toISOString()`	Retorna la fecha en formato ISO.	`fecha.toISOString(); // '2024-05-15T10:30:00.000Z'`
`toLocaleDateString()`	Formatea la fecha según la localización.	`fecha.toLocaleDateString('es-ES'); // '15/05/2024'`
`getTime()`	Retorna el tiempo en milisegundos desde el Epoch.	`fecha.getTime();`

Ejemplos de Uso

```
const fecha = new Date();

// Obtener componentes de la fecha
const año = fecha.getFullYear();
const mes = fecha.getMonth() + 1; // +1 porque los meses so
n 0-11
const dia = fecha.getDate();

// Formatear la fecha
const formato = `${dia}/${mes}/${año}`;
console.log(formato); // e.g., '15/5/2024'

// Calcular la diferencia entre dos fechas
const otraFecha = new Date('2024-12-31');
const diferencia = otraFecha - fecha; // en milisegundos
const dias = diferencia / (1000 * 60 * 60 * 24);
console.log(`Faltan ${Math.floor(dias)} días para el 31 de
diciembre de 2024.`);
```

- **Fecha Actual:** `const ahora = new Date();`

- **Formato ISO:** `ahora.toISOString();`
- **Componentes:** `fecha.getFullYear(), fecha.getMonth(), fecha.getDate()`
- **Diferencia:** `otraFecha - fecha; // milisegundos`

String

Descripción

El objeto `String` representa cadenas de texto y proporciona múltiples métodos para manipularlas.

Métodos Comunes

Método	Descripción	Ejemplo
`length`	Retorna la longitud de la cadena.	`'Hola'.length; // 4`
`toUpperCase()`	Convierte la cadena a mayúsculas.	`'hola'.toUpperCase(); // 'HOLA'`
`toLowerCase()`	Convierte la cadena a minúsculas.	`'HOLA'.toLowerCase(); // 'hola'`
`charAt(indice)`	Retorna el carácter en el índice especificado.	`'Hola'.charAt(1); // 'o'`
`includes(subcadena)`	Verifica si incluye la subcadena.	`'Hola Mundo'.includes('Mundo'); // true`
`indexOf(subcadena)`	Retorna el índice de la primera ocurrencia.	`'Hola'.indexOf('o'); // 1`
`substring(inicio, fin)`	Retorna una parte de la cadena.	`'Hola'.substring(1, 3); // 'ol'`
`slice(inicio, fin)`	Similar a `substring`.	`'Hola'.slice(-2); // 'la'`
`split(separador)`	Divide la cadena en un array según el separador.	`'a,b,c'.split(','); // ['a', 'b', 'c']`
`replace(buscar, reemplazar)`	Reemplaza la primera ocurrencia.	`'Hola Mundo'.replace('Mundo', 'JS'); // 'Hola JS'`
`replaceAll(buscar, reemplazar)`	Reemplaza todas las ocurrencias (ES2021).	`'a a a'.replaceAll('a', 'b'); // 'b b b'`

`trim()`	Elimina espacios al inicio y al final.	`' Hola '.trim(); // 'Hola'`
`startsWith(subcadena)`	Verifica si comienza con la subcadena.	`'Hola'.startsWith('Ho'); // true`
`endsWith(subcadena)`	Verifica si termina con la subcadena.	`'Hola'.endsWith('la'); // true`
`padStart(longitud, relleno)`	Rellena al inicio hasta alcanzar la longitud.	`'5'.padStart(2, '0'); // '05'`
`padEnd(longitud, relleno)`	Rellena al final hasta alcanzar la longitud.	`'5'.padEnd(2, '0'); // '50'`

Interpolación de Cadenas (Template Literals)

```
const nombre = 'Juan';
const edad = 30;

// Uso de comillas invertidas y `${}` para insertar variabl
es
const saludo = `Hola, me llamo ${nombre} y tengo ${edad} añ
os.`;
console.log(saludo); // 'Hola, me llamo Juan y tengo 30 año
s.'
```

Ejemplos de Uso

```
const texto = '  JavaScript es genial!  ';

// Eliminar espacios
const limpio = texto.trim();
console.log(limpio); // 'JavaScript es genial!'

// Convertir a mayúsculas
const mayus = limpio.toUpperCase();
console.log(mayus); // 'JAVASCRIPT ES GENIAL!'

// Reemplazar una palabra
const reemplazo = limpio.replace('genial', 'asombroso');
console.log(reemplazo); // 'JavaScript es asombroso!'
```

```
// Dividir la cadena en palabras
const palabras = limpio.split(' ');
console.log(palabras); // ['JavaScript', 'es', 'genial!']
```

- **Longitud:** `'Hola'.length; // 4`
- **Mayúsculas:** `'hola'.toUpperCase(); // 'HOLA'`
- **Reemplazar:** `'Hola Mundo'.replace('Mundo', 'JS'); // 'Hola JS'`
- **Dividir:** `'a,b,c'.split(','); // ['a', 'b', 'c']`

Expresiones Regulares (RegExp)

Descripción

Las expresiones regulares son patrones utilizados para buscar, validar y manipular cadenas de texto.

Creación de Expresiones Regulares

```
// Notación literal
const regex1 = /hola/i; // 'i' para insensible a mayúsculas

// Constructor
const regex2 = new RegExp('hola', 'i');
```

Métodos Comunes

Método	Descripción	Ejemplo
`test(cadena)`	Retorna `true` si la cadena cumple el patrón.	`/abc/.test('abcdef'); // true`
`exec(cadena)`	Retorna un array con la información de la coincidencia o `null`.	`/abc/.exec('abcdef'); // ['abc']`
`match(regex)`	Retorna las coincidencias de la regex en la cadena.	`'abcdef'.match(/abc/); // ['abc']`
`search(regex)`	Retorna el índice de la primera coincidencia o `-1`.	`'abcdef'.search(/d/); // 3`

`replace(regex, reemplazo)`	Reemplaza las coincidencias con el reemplazo.	`'hola mundo'.replace(/mundo/, 'JS'); // 'hola JS'`
`split(regex)`	Divide la cadena según el patrón.	`'a,b,c'.split(/,/); // ['a', 'b', 'c']`

Características Comunes

Característica	Descripción
`.`	Cualquier carácter excepto nueva línea.
`^`	Inicio de la cadena.
`$`	Fin de la cadena.
`*`	Cero o más repeticiones.
`+`	Una o más repeticiones.
`?`	Cero o una repetición.
`{n}`	Exactamente `n` repeticiones.
`{n,}`	Al menos `n` repeticiones.
`{n,m}`	Entre `n` y `m` repeticiones.
`[]`	Conjunto de caracteres.
`	`
`()`	Agrupación.
`\\d`	Dígito (0-9).
`\\w`	Caracter de palabra (alfanumérico y _).
`\\s`	Espacio en blanco.

Ejemplos de Uso

```
const correo = 'usuario@example.com';
const regexCorreo = /^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+$/;

// Validar correo electrónico
const esValido = regexCorreo.test(correo);
console.log(esValido); // true

// Extraer palabras de una cadena
```

```
const frase = 'Aprendiendo JavaScript con expresiones regul
ares.';
const palabras = frase.match(/\\b\\w+\\b/g);
console.log(palabras); // ['Aprendiendo', 'JavaScript', 'co
n', 'expresiones', 'regulares']

// Reemplazar espacios por guiones
const url = '<https://www.ejemplo.com/mi> pagina';
const urlFormateada = url.replace(/\\s+/g, '-');
console.log(urlFormateada); // '<https://www.ejemplo.com/mi
-pagina>'
```

Modificadores Comunes

Modificador	Descripción
`i`	Insensible a mayúsculas y minúsculas.
`g`	Búsqueda global (todas las coincidencias).
`m`	Búsqueda multilínea.
`s`	Permite que `.` coincida con caracteres de nueva línea.
`u`	Tratamiento de Unicode completo.
`y`	Búsqueda "sticky" que coincide desde el último índice.

- **Validar Correo:**

```
const regexCorreo = /^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.
[a-zA-Z0-9-.]+$/;
regexCorreo.test('usuario@example.com'); // true
```

- **Extraer Palabras:**

```
'Hola Mundo'.match(/\\b\\w+\\b/g); // ['Hola', 'Mundo']
```

- **Reemplazar Texto:**

```
'a b c'.replace(/\\s/g, '-'); // 'a-b-c'
```