Youtic's
Lic. Gabriel Chaldu

JavaScript – Arrays

En JavaScript, un "array" (también conocido como arreglo o matriz) es una estructura de datos que se utiliza para almacenar y organizar una colección de elementos relacionados. Los elementos en un array pueden ser de cualquier tipo de datos, como números, cadenas de texto, objetos, funciones u otros arrays. Los arrays se utilizan comúnmente para agrupar y manipular datos de manera eficiente.

Los elementos en un array están indexados, lo que significa que cada elemento tiene un número asociado llamado índice que indica su posición en el array. El primer elemento tiene un índice de 0, el segundo tiene un índice de 1 y así sucesivamente.

Aquí hay un ejemplo simple de cómo se puede crear y acceder a un array en JavaScript:

```
// Crear un array con algunos elementos
const miArray = [1, 2, 3, 4, 5];

// Acceder a un elemento por su índice
const primerElemento = miArray[0]; // Valor: 1
const segundoElemento = miArray[1]; // Valor: 2

// Modificar un elemento por su índice
miArray[2] = 10; // Ahora el array es [1, 2, 10, 4, 5]

// Obtener la longitud (cantidad de elementos) del array
const longitud = miArray.length; // Valor: 5

// Agregar un elemento al final del array
miArray.push(6); // Ahora el array es [1, 2, 10, 4, 5, 6]
```

AÑADIR ELEMENTOS A UN ARREGLO

```
const miArray2 = [1, 2, 3, 4, 5];
//agregar en una posición
miArray2[5]=6
//agrega al final
miArray2.push(10)
//agrega al inicio
miArray2.unshift(20)
//agregar spread operator
const newArray = [...miArray2, 1000]
```

Youtic's
Lic. Gabriel Chaldu

ELIMINAR ELEMENTOS DE UNA ARREGLO

```
const miArray = [1, 2, 3, 4, 5];
//elimina el primer elemento del arreglo
miArray.shift()
//elimina desde la posición actual la cantidad que querramos
miArray.splice(2,1);
//elimina desde la pos actual hasta el final TODOS
miArray.splice(2);

console.log(miArray)
```

SPREAD OPERATOR AÑADIR ELEMENTO

```
const miArray = [1, 2, 3, 4, 5];
const newArray = [...miArray, 1000]
```

SPREAD OPERATOR - ELIMINAR

Puedes utilizar el operador de propagación (spread operator) para eliminar elementos específicos de un arreglo en JavaScript. Sin embargo, es importante destacar que el operador de propagación no modifica el arreglo original, sino que crea un nuevo arreglo con los elementos deseados.

Aquí te muestro cómo puedes usar el operador de propagación para eliminar elementos de un arreglo:

```
let miArray = [1, 2, 3, 4, 5];
// Supongamos que queremos eliminar el elemento en el índice 2 (valor 3)
let indiceAEliminar = 2;
//Usamos el spread operator para crear un nuevo arreglo sin el elemento
en el índice indicado
let nuevoArray = [...miArray.slice(0, indiceAEliminar),
...miArray.slice(indiceAEliminar + 1)];
console.log(nuevoArray); // Resultado: [1, 2, 4, 5]
```

En este ejemplo, hemos utilizado el método `slice()` para dividir el arreglo en dos partes: una antes del elemento a eliminar y otra después del elemento a eliminar. Luego, usamos el operador de propagación para crear un nuevo arreglo combinando estas dos partes, lo que efectivamente elimina el elemento en el índice indicado.

Recuerda que el arreglo original `miArray` no se modifica en este proceso. Si deseas modificar directamente el arreglo original para eliminar elementos, puedes usar métodos como `splice()`, que alteran el arreglo existente.

Recorrer con foreach

```
let miArray = [1, 2, 3, 4, 5];
miArray.forEach(num=>{
    console.log(num)
})
```

Recorrer con un map

```
let miArray = [1, 2, 3, 4, 5];
const arrayFiltrado = miArray.map(numero => numero > 2)
console.log(arrayFiltrado)//retorna un array con false, false, true,
true,true
```

Recorrer y filtrar

```
let miArray = [1, 2, 3, 4, 5];
const arrayFiltrado = miArray.filter(numero => numero > 2)
console.log(arrayFiltrado)//retorna un array con 3,4,5
```

Math

La biblioteca "Math" en la mayoría de los lenguajes de programación proporciona una amplia gama de funciones matemáticas para realizar operaciones comunes y avanzadas.

A continuación, se presentan algunas de las funciones más importantes y comunes que suelen estar disponibles en la biblioteca Math:

1. **Funciones Trigonométricas**: Estas funciones incluyen seno, coseno y tangente, junto con sus inversas (arcoseno, arcocoseno y arcotangente). Son utilizadas para realizar cálculos relacionados con ángulos y triángulos.

2. **Funciones Exponenciales y Logarítmicas**: Estas funciones incluyen exponente, logaritmo natural y logaritmo en base 10. Son útiles para cálculos que involucran crecimiento exponencial y análisis de datos logarítmicos

3. **Funciones de Redondeo y Truncamiento**: Incluyen funciones como `Math.round()` para redondear al entero más cercano, `Math.floor()` para truncar hacia abajo y `Math.ceil()` para truncar hacia arriba.

4. **Funciones de Valor Absoluto**: La función `Math.abs()` devuelve el valor absoluto de un número, es decir, su distancia respecto a cero en la recta numérica.

5. **Funciones de Potenciación y Raíces**: Incluyen `Math.pow()` para calcular potencias y `Math.sqrt()` para calcular raíces cuadradas.

6. **Funciones de Aleatoriedad**: `Math.random()` genera un número pseudoaleatorio en el rango [0, 1). Esto es útil para generar números aleatorios para simulaciones y juegos

7. **Funciones de Mínimo y Máximo**: `Math.min()` y `Math.max()` se utilizan para encontrar el valor mínimo y máximo de un conjunto de números.

8. **Función de Número PI**: La constante `Math.PI` proporciona el valor numérico de π (pi), que es una constante matemática fundamental en geometría y trigonometría.

9. **Función de Redondeo Decimal**: `Math.round()` puede ser utilizada con un segundo argumento para especificar la cantidad de decimales a la que se debe redondear un número.

10. **Funciones Trigonometría Hiperbólica**: Además de las funciones trigonométricas estándar, la biblioteca Math también puede proporcionar funciones trigonométricas hiperbólicas como seno hiperbólico, coseno hiperbólico y tangente hiperbólica.

Ejemplos:

```
const base = 2;
const exponent = 3;
const powerResult = Math.pow(base, exponent);

const number = 100;
const naturalLog = Math.log(number);
const logBase10 = Math.log10(number);

console.log(`2^3 = ${powerResult}`);
```

```
console.log(`Logaritmo natural de 100: ${naturalLog}`);
console.log(`Logaritmo base 10 de 100: ${logBase10}`);
```

Funciones de Redondeo y Truncamiento:

```
const decimalNumber = 3.7;
const roundedNumber = Math.round(decimalNumber);
const floorNumber = Math.floor(decimalNumber);
const ceilNumber = Math.ceil(decimalNumber);

console.log(`Número redondeado: ${roundedNumber}`);
console.log(`Número truncado hacia abajo: ${floorNumber}`);
console.log(`Número truncado hacia arriba: ${ceilNumber}`);
```

Función de Valor Absoluto:

```
const negativeNumber = -5;
const absoluteValue = Math.abs(negativeNumber);

console.log(`Valor absoluto de ${negativeNumber}: ${absoluteValue}`);
```

Funciones de Aleatoriedad:

```
const randomValue = Math.random(); // Genera un número entre 0 (incluido)
y 1 (excluido)

console.log(`Número aleatorio: ${randomValue}`);
```

Función Math.min() y Math.max() con Números:

```
const num1 = 10;
const num2 = 5;
const num3 = 8;

const minValue = Math.min(num1, num2, num3);
const maxValue = Math.max(num1, num2, num3);

console.log(`El valor mínimo es: ${minValue}`);
console.log(`El valor máximo es: ${maxValue}`);
```

```
const numbers = [15, 3, 9, 7, 21, 1];

const minFromArray = Math.min(...numbers);
const maxFromArray = Math.max(...numbers);

console.log(`El valor mínimo del array es: ${minFromArray}`);
console.log(`El valor máximo del array es: ${maxFromArray}`);
```