UD2 – JS, Arrays y Elementos Predefinidos

DAW2-DWEC

Básicos Arrays

Permiten almacenar cualquier tipo de datos

Tamaño dinámico

- `let numeros = [];`
- `numeros = [1, 2, 3, 4, 5];`
- `numeros[0];`
- `numeros[2] = 33;`
- `numeros[numeros.length - 1];`
- `numeros.at(-1); // 5, sólo lectura`

Cuidado

- `numeros = new Array(4); // Instancio un array de 4 posiciones`
- `numeros = new Array(1, 2);`

Básicos Arrays - II

Arrays multidimensionales

- Sólo disponemos de arrays unidimensionales.
- => TRUCO: construir arrays de arrays.

```
let matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
console.log(matriz[1][1]); // 5
```

Copia de arrays

- El operador "..." crea una copia del array. Cuidado con los objetos que no se clonan.

```
let numeros = new Array(1, 2);
let numeros2 = [...numeros];
numeros2[0] = 11;
console.log(numeros); // [1, 2]
console.log(numeros2); // [11, 2]
```

Recorrer Arrays

Operadores

- **for** (inicialización; condición; actualización){}
- **for** (elemento **of** colección){}

Métodos

- .**forEach**(fn(elemento))
- .**map**(fn(elemento)): devuelve una copia con el array modificado

```
let resultado = numeros.map(numero => numero * 2);
console.log(resultado); // [2, 4]
console.log(numeros); // [1, 2]
```

Métodos en Arrays

Métodos mutables -> modifican el array

Métodos NO mutables -> no modifican el array (devuelven una copia)

Búsquedas

- .indexOf(valor, inicio=0): devuelve -1 si no encuentra el valor.
- .filter(función)

Ordenar

- .sort()
- .toSorted()

Admiten una función de búsqueda personalizada (a, b) => { return -1, 0, 1 }

Métodos comunes en Arrays

Añadir y eliminar elementos [<<, <, >, >>]

- << .shift():Object, .unshift(…elementos), .push(…elementos), .pop():Object >>

Juntar, partir, concatenar

- .join(separador)
- .split(separador)
- .concat(…arrays)

Aplanar

- .flat(profundidad=1)

```
[1, [2, [3, 4]]].flat(2);  // [1, 2, 3, 4]
```

Métodos comunes en Arrays - II

Invertir

- .reverse()
- .toReverse():[]

Inicializar a valor

- .fill(valor por defecto, inicio?, fin?)

Extraer una porción

- .slice(inicio, fin?)

Objetos predefinidos del lenguaje

El lenguaje JavaScript incluye una colección de elementos que facilitan la codificación.

Adicionalmente el entorno de ejecución o el navegador pueden ampliar esta colección.

Vamos a ver selección de los objetos predefinidos más comunes.

Propiedades

- globalThis
- Infinity
- NaN
- undefined

Funciones predefinidas

- **eval(cadena)**: ejecuta la cadena de entrada como código JS.
- **isFinite(valor)**: devuelve true, false o NaN
- **parseFloat(cadena)**
- **parseInt(cadena, baseEntrada=10)**
- **decodeURI(uri)**: remplaza los caracteres de escape “%xx” por los caracteres especiales equivalentes.
- **encodeURI(uri)**: remplaza ciertos caracteres por su equivalente de escape “%xx”.

```
let uri = "https://example.com?name=John%20Doe";
console.log(decodeURI(uri)); // "https://example.com?name=John Doe"

        uri = "https://example.com?name=John Doe";
        console.log(encodeURI(uri)); // "https://example.com?name=John%20Doe"
```

Temporizadores

NOTA: no son propias del lenguaje JS, las añaden los navegadores

- **setTimeout(función, tiempo):** ejecuta una función pasado un tiempo.
- clearTimeout(id): cancela el timeout.
- **setInterval(función, tiempo):** ejecuta una función periódicamente.
- clearInterval(id): cancela el interva.

```
let intervalo = setInterval(() => console.log("Hola cada 3 segundos"), 3000);

clearInterval(intervalo);
```

Objeto String

.charAt(posición)

.concat(...cadenas)

.indexOf(subcadena, posiciónInicio=0)

.slice(inicio, fin)

.replace(subcadena, nuevaSubcadena)

.split(separador)

.toLowerCase()

.toUpperCase()

.trim()

.localeCompare(cadena, locales?)

Objeto Math

Propiedades

- Constantes: .PI, .E, .LN10, .LN2, .LOG10E, etc…

Métodos de redondeo

- Math.ceil(x)
- Math.floor(x)
- Math.round(x)

Otros métodos

- Math.random(): devuelve un valor entre [0, 1)
- .abs(x), .max(…), .min(…), .sqrt(x), .pow(b, e), etc….

Objeto Date

RECUERDA: las fechas en JS funcionan almacenando los milisegundos transcurridos desde el 1 de enero de 1979 UTC, el uso horario de España es UTC+1.

No debes preocuparte por la zona horaria porque los métodos básicos gestionan automáticamente la zona horaria.

Crear una fecha

- **new Date():** devuelve objeto con la fecha y hora actual.
- **new Date(YYYY-MM-DD HH:mm:ss:sss):** devuelve objeto con la fecha indicada.
- **new Date(año, mes=1, día=1, horas=0, minutos=0, segundos=0, milisegundos=0):** los mese van de 0 a 11.

Leer Date

.getFullYear()

.getMonth(): Devuelve el mes (0 = enero, 11 = diciembre).

.getDate(): Devuelve el día del mes (1-31).

.getDay(): Devuelve el día de la semana (0 = domingo, 6 = sábado).

.getHours(): Devuelve la hora (0-23).

.getMinutes(): Devuelve los minutos (0-59).

.getSeconds(): Devuelve los segundos (0-59).

.getTime(): Devuelve el número de milisegundos transcurridos desde el 1 de enero de 1970.

Escribir Date

.setFullYear(año)

.setMonth(mes): Establece el mes (0-11).

.setDate(día): Establece el día del mes (1-31).

.setHours(horas): Establece la hora (0-23).

.setMinutes(minutos): Establece los minutos (0-59).

Consideraciones Date

Puedo restar dos fechas obteniendo el número de milisegundos entre ellas.

```
let fecha1 = new Date('2023-09-01');
let fecha2 = new Date('2023-09-30');
let diferencia = fecha2 - fecha1;
let diasDiferencia = diferencia / (1000 * 60 * 60 * 24); // 29 días
```

Métodos estáticos

Sirven para calcular tiempos sobre la fecha de referencia. Todos devuelven los milisegundos transcurridos desde la fecha de referencia 1 de enero de 1970

- Date.now()
- Date.parse(cadenaFecha)
- Date.UTC(año, mes, día, horas, minutos, segundos, milisegundos)

Objeto JSON

"JavaScript Object Notation" es un formato de texto para el intercambio de datos.

Se basa en la sintaxis de JS por lo que se integra con facilidad en el lenguaje.

- **JSON.stringify(objeto):** Para convertir objetos de JavaScript en cadenas JSON.
- **JSON.parse(cadena, fn(propiedad, valor)?):** Para convertir cadenas JSON en objetos de JavaScript.

```
let objeto = { nombre: "Juan", edad: 30, activo: true };
let cadenaJSON = JSON.stringify(objeto);
console.log(cadenaJSON); // '{"nombre":"Juan","edad":30,"activo":true}'

cadenaJSON = '{"nombre":"Juan","edad":30,"activo":true}';
objeto = JSON.parse(cadenaJSON);
console.log(objeto.nombre); // "Juan"
```

CUIDADO CON LA SINTAXIS DE LAS CADENAS JSON, lo errores producen excepciones.

Colecciones Predefinidas

3 tipos: **Array, Map, Set**

El tipo Array se ha tratado de manera independiente debido a su importancia.

Map

Colección de pares clave-valor. Se suele usar JSON por las ventajas que aporta en comunicación y tratamiento.

Set

Colección que permite definir un conjunto. Los elementos del conjunto no se repiten.

```
let mapa = new Map();
mapa.set('nombre', 'Juan');
mapa.set('edad', 30);

for (let clave of mapa.keys()) {
  console.log(clave); // "nombre", "edad"
}
```

```
let conjunto = new Set();
conjunto.add(1);
conjunto.add(2);
conjunto.add(3);

console.log(conjunto.has(2)); // true
conjunto.delete(2);
console.log(conjunto.size); // 2
```

Preguntas