FORMACIÓN PROFESIONAL
MONTECASTELO

Objetos nativos

Desarrollo Web en Entorno Cliente

Ciclo Superior de Desarrollo de Aplicaciones Web

2023/2024

Tipos de objetos

- Los objetos nativos son aquellos proporcionados por el entorno de JS de manera predeterminada.
- Ejemplos de objetos nativos son String, Number, Array, Date, Math, etc.
- En cambio, los objetos definidos por el usuario son aquellos creados por el programador.

Objetos nativos

- Los objetos nativos son accesibles en cualquier lugar de un programa y funcionan del mismo modo en cualquier navegador de modo independiente al sistema operativo.
- Son objetos nativos:
  - Number
  - Boolean
  - Date
  - Math
  - String
  - Array

Objetos nativos

- Es importante no confundir los tipos de datos primitivos `number`, `string` y `boolean`, con los correspondientes objetos nativos `Number`, `String` y `Boolean`.
- Los objetos nativos envuelven a datos primitivos y proporcionan funcionalidades auxiliares para trabajar con tipos primitivos.
- Por ejemplo, el objeto `String` proporciona operaciones comunes sobre cadenas como concatenar, buscar, reemplazar, etc.

Objeto nativos

- El objeto `Number` permite trabajar con valores numéricos.
- El objeto `String` representa una serie de caracteres que forman una cadena.
- El objeto `Boolean` es un objeto envoltorio para un dato que puede tomar los valores `true` o `false`.

Objetos nativos

- El objeto `Date` permite trabajar con fechas y horas.
- El objeto nativo `Math` proporciona constantes matemáticas de uso común, como el número PI o el número E, así como métodos que implementan funciones matemáticas que no cuentan con un operador propio en JS: raíz cuadrada, funciones trigonométricas, etc.

Creación y manejo de objeto nativo

- Los objetos nativos se crean invocando su constructor con la palabra reservada `new`:

```
let num = new Number(9.8765);
```

- Los métodos de los objetos nativos se invocan del siguiente modo:

```
let num2 = num.toFixed(2); // 9.88
```

- Los propiedades de los objetos nativos se invocan del siguiente modo:

```
let num3 = Number.MAX_VALUE;
```

Objetos nativos

- Si se invoca el constructor de un objeto nativo sin la palabra reservada `new`, simplemente se convierte el valor pasado al correspondiente valor primitivo.

```
let value = "25"; // cadena
let num = Number(value); // conversión a número
typeof num; // devuelve "number" (primitivo)
```

- Si se emplea `new`, se convierte el valor al correspondiente valor primitivo y además se genera un objeto que envuelve a dicho valor primitivo.

```
let value = "25";
let obj = new Number(value); //constructor
typeof obj; // devuelve "object"
```

Objetos nativos

- En ocasiones se invocan métodos de objetos nativos sobre tipos de datos primitivos.
- Aunque a primera vista pueda parecer una contradicción, esto es posible debido a que JS convierte el dato primitivo en el correspondiente objeto nativo para poder invocar el método.

```
let x = 25;
typeof x; // devuelve number (primitivo)
x.toString();
/* la variable x no es un objeto, JS lo
convierte en objeto de tipo Number para
poder invocar toString() */
```

Arrays

- El objeto `Array` permite almacenar varios valores en una única variable.
- Un array es simplemente una lista formada por otros valores de JS.
- Los valores individuales de un array se denominan elementos.

Creación de arrays

- Para crear un array se escriben entre corchetes sus valores separados por comas.

```
let x = [100, 200, 300];
```

- A diferencia de otros lenguajes de programación, los elementos de un array de JS no tienen que ser todos del mismo tipo.

```
let x = [1, true, “casa”];
```

Acceso a los elementos de un array

- Para acceder a un elemento de un array se emplea el índice de dicho elemento entre corchetes.
- Un índice es el número que identifica la posición del elemento dentro del array, comenzando por la posición cero.

```
let x = [1, true, “casa”];
let y = x[1]; // devuelve true
```

Acceso a los elementos de un array

- JavaScript es flexible en cuanto al acceso y modificación de los elementos de un array.
- Si se da valor a una posición de array que no existe, en lugar de generar un error, se crea un nuevo elemento en dicha posición.

```
let x = [1];
x[2] = “hola”;
// Resultaría x = [1 undefined “hola”]
```

Propiedades `length`

- La propiedad `length` de un array permite conocer el número de elementos del array.
- De ese modo, el último índice de un array coincide con el valor de `length` menos 1.

```
for (let i=0; i < unArray.length; i++)
{
 console.log(unArray[i]);
}
```

Añadir elementos a un array

- Para añadir un elemento al final de un array se puede emplear el método `.push()`.
- Cuando se invoca `.push()` se llevan a cabo dos acciones:
  1. El nuevo elemento se añade al array.
  2. La nueva longitud del array se devuelve.
- Si se necesita añadir elementos al comienzo del array se emplea `.unshift()` en lugar de `.push()`.

Eliminar elementos de un array

- Para eliminar el último elemento de un array se emplea el método `.pop()`.
- Si se necesita eliminar elementos al comienzo del array se emplea `.shift()` en lugar de `.pop()`.

Unir arrays

- Para unir arrays se emplea el método `.concat()`.

  `unArray.concat(otroArray)`
- El método anterior combinará ambos arrays añadiendo los valores de `otroArray` a continuación de los valores de `unArray`.
- Se puede usar `concat` para unir más de dos arrays pasando todos los arrays separados por comas.

Unir arrays

- ATENCIÓN: concat no modifica ninguno de los arrays unidos, sino que devuelve como resultado el nuevo array.

Índice de un elemento del array

- Se puede obtener el índice de un elemento dentro de un array usando `unArray.indexOf(elemento)`.
- Puede emplearse dicho método para comprobar si un elemento está dentro de un array o no.

Convertir un array en una cadena

- El método `.join()` permite generar una cadena con todos los elementos de un array.
- Si se llama sin argumentos, el método genera una cadena con los elementos separados por comas.
- Se puede emplear `.join(separador)` para pasar un separador distinto de la coma.

Referencias

- JavaScript en w3schools
- JavaScript en Mozilla Developer Network

FORMACIÓN PROFESIONAL

MONTECASTELO