Clase 15: Métodos y Propiedades Básicos

Índice

Métodos de Arrays (Básicos):

- Push
- Pop
- Shift
- Unshift
- Join
- indexOf
- lasIndexOf
- includes

Propiedades en Strings:

- .length
- indexOf

Métodos de String:

- Slice
- Trim
- Replace
- split

Métodos de Arrays (Basicos)

Métodos

En Arrays

Para JavaScript, los arrays son un **tipo especial de objetos**. Por esta razón disponemos de muchos **métodos** muy útiles a la hora de trabajar con la información que hay adentro.

Métodos

En Arrays

Ya vimos antes que una función es un bloque de código que nos permite agrupar funcionalidad para usarla muchas veces.

Cuando una función pertenece a un objeto, en este caso nuestro array, la llamamos **método**.

Métodos

.push()

Agrega uno o varios elementos al final del array.

- Recibe uno o más elementos como parámetros.
- Retorna la nueva longitud del array.


```
let colores = ['Rojo','Naranja','Azul'];
colores.push('Violeta'); // retorna 4
console.log(colores);
// ['Rojo','Naranja','Azul','Violeta']

colores.push('Gris','Oro');
console.log(colores);
// ['Rojo','Naranja','Azul','Violeta','Gris','Oro']
```

Métodos

.pop()

Elimina el último elemento de un array.

- No recibe parámetros.
- Devuelve el elemento eliminado.


```
let series = ['Mad Men','Breaking Bad','The Sopranos'];

// creamos una variable para guardar lo que devuelve .pop()
let ultimaSerie = series.pop();

console.log(series); // ['Mad men', 'Breaking Bad']
console.log(ultimaSerie); // ['The Sopranos']
```

Métodos

.shift()

Elimina el primer elemento de un array.

- No recibe parámetros.
- Devuelve el elemento eliminado.


```
let nombres = ['Frida','Diego','Sofía'];

// creamos una variable para guardar lo que devuelve .shift()
let primerNombre = nombres.shift();

console.log(nombres); // ['Diego', 'Sofía']
console.log(primerNombre); // ['Frida']
```

Métodos

.unshift()

Agrega uno o varios elementos al principio de un array.

- Recibe uno o más elementos como parámetros.
- Retorna la nueva longitud del array.


```js
let marcas = ['Audi'];

marcas.unshift('Ford');
console.log(marcas); // ['Ford', 'Audi']

marcas.unshift('Ferrari','BMW');
console.log(marcas); // ['Ferrari','BMW','Ford', 'Audi']
```

Métodos

.join()

Une los elementos de un array utilizando el separador que le especifiquemos. Si no lo especificamos, utiliza comas.

- Recibe un separador (string), es opcional.
- Retorna un string con los elementos unidos.


```js
let dias = ['Lunes','Martes','Jueves'];

let separadosPorComa = dias.join();
console.log(separadosPorComa); // 'Lunes,Martes,Jueves'

let separadosPorGuion = dias.join(' - ');
console.log(separadosPorGuion); // 'Lunes - Martes - Jueves'
```

Métodos

.indexOf()

Busca en el array el elemento que recibe como parámetro.

- Recibe un elemento a buscar en el array.
- Retorna el primer índice donde encontró lo que buscábamos. Si no lo encuentra, retorna un -1.


```
let frutas = ['Manzana','Pera','Frutilla'];
frutas.indexOf('Frutilla');
// Encontró lo que buscaba. Devuelve 2, el índice del elemento


frutas.indexOf('Banana');
// No encontró lo que buscaba. Devuelve -1
```

Métodos

.lastIndexOf()

Similar a .indexOf(), con la salvedad de que empieza buscando el elemento por el final del array (de atrás hacia adelante).
En caso de haber elementos repetidos, devuelve la posición del primero que encuentre (o sea el último si miramos desde el principio).


```
let clubes = ['Racing','Boca','Lanús','Boca'];

clubes.lastIndexOf('Boca');
// Encontró lo que buscaba. Devuelve 3

clubes.lastIndexOf('River');
// No encontró lo que buscaba. Devuelve -1
```

Métodos

.includes()

También similar a .indexOf(), con la salvedad que retorna un booleano.

- Recibe un elemento a buscar en el array.
- Retorna true si encontró lo que buscábamos, false en caso contrario.


```
let frutas = ['Manzana','Pera','Frutilla'];

frutas.includes('Frutilla');
// Encontró lo que buscaba. Devuelve true

frutas.includes('Banana');
// No encontró lo que buscaba. Devuelve false
```

Métodos en Arrays

Arrays

- miArray.**length**:
  - Devuelve la cantidad de elementos del array.
- miArray.**push**(elemento1, elemento2, …., elementox):
  - Modifica miArray agregando elemento1, elemento2, …, elementox al final de miArray y devuelve la nueva cantidad de elementos de miArray.
- miArray.**pop**():
  - Modifica miArray eliminando el último elemento y devuelve el elemento eliminado.
- miArray.**shift**():
  - Modifica miArray eliminando el primer elemento y devuelve el elemento eliminado.
- miArray.**unshift**(elemento1, elemento2, …, elementox):
  - Modifica miArray agregando elemento1, elemento2, …, elementox al principio de miArray y devuelve la nueva cantidad de elementos de miArray.
- miArray.**join**():
  - Junta los elementos de miArray y devuelve todo en un string unidos por ",".
- miArray.**join**(divisorDeElementos):
  - Junta los elementos de miArray y devuelve todo en un string unidos por el texto divisorDeElementos. Por ejemplo: ["A", "B"].join("/") devolverá "A/B".
- miArray.**indexOf**(elemento):
  - Busca al elemento en miArray. Si existe, devuelve la primera posición donde lo encuentra. Si no, devuelve -1.
- miArray.**lastIndexOf**(elemento):
  - Busca al elemento en miArray empezando desde atrás (desde el último elemento). Si existe devuelve la posición de la primera vez que lo encuentra contando desde atrás. Si no, devuelve -1. Por ejemplo: ["a", "b", "c", "b"].lastIndexOf("b") devuelve 3 porque empezó a buscar desde atrás y la primera vez que lo encontró es en la última posición.
- miArray.**includes**(elemento):
  - Busca al elemento en miArray. Si existe, devuelve true. Si no, false.

Memori Rapido:

Pop para extraer

Push para insertar elementos al final

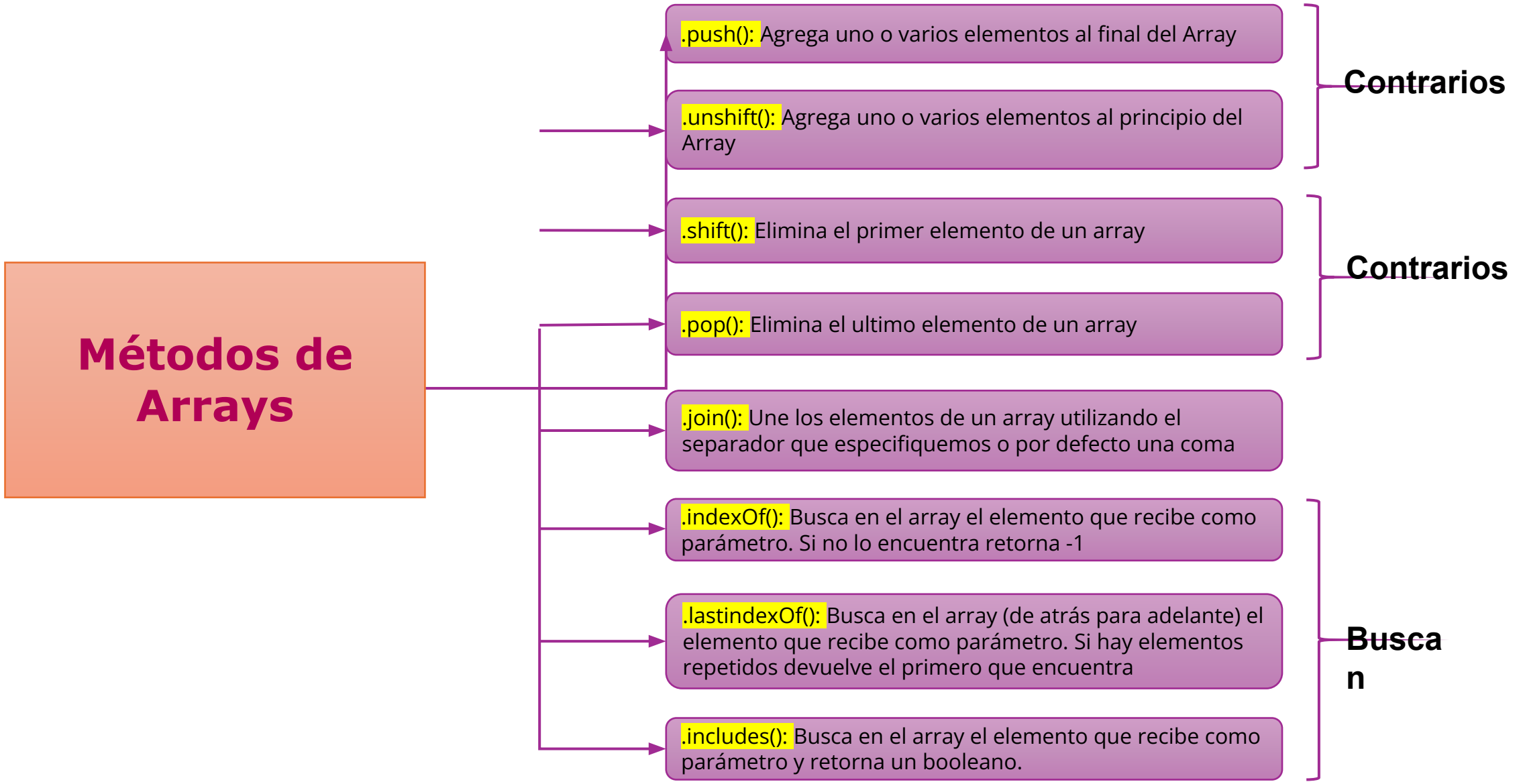
Shift para extraer

Unshift para insertar elementos al principio de un array

IndexOf / lastIndexOf para preguntar el índice de una ocurrencia

Join para unificar todos los elementos presentes en un array

Veamos mas en VSC!


Métodos de Arrays
.push(): Agrega uno o varios elementos al final del Array
.unshift(): Agrega uno o varios elementos al principio del Array
Contrarios
.shift(): Elimina el primer elemento de un array
.pop(): Elimina el ultimo elemento de un array
Contrarios
.join(): Une los elementos de un array utilizando el separador que especifiquemos o por defecto una coma
.indexOf(): Busca en el array el elemento que recibe como parámetro. Si no lo encuentra retorna -1
.lastindexOf(): Busca en el array (de atrás para adelante) el elemento que recibe como parámetro. Si hay elementos repetidos devuelve el primero que encuentra
.includes(): Busca en el array el elemento que recibe como parámetro y retorna un booleano.
Busca n

Propiedades y Métodos de Strings

Propiedades y Métodos

Strings

Para JavaScript los strings son como una **colección** de caracteres. Por esta razón disponemos de **propiedades** y **métodos** muy útiles a la hora de trabajar con la información que hay adentro.

Los strings en JavaScript

En muchos sentidos, para JavaScript, un string no es más que un array de caracteres. Al igual que en los arrays, la primera posición siempre será 0.

Para acceder a un carácter puntual de un string, nombramos al string y, dentro de los corchetes, escribimos el índice al cual queremos acceder.

Propiedades en Strings

.length

Esta propiedad retorna la cantidad total de caracteres del string, incluidos los espacios.
Al ser una propiedad (veremos más sobre ellas en clases siguientes), solo debemos llamarla, sin necesidad de los paréntesis.


```
let miSerie = 'Mad Men';
miSerie.length; // devuelve 7

let arrayNombres = ['Bart', 'Lisa', 'Moe'];
arrayNombres.length; // devuelve 3

arrayNombres[0].length; // Corresponde a 'Bart', devuelve 4
```

Propiedades en Strings

.indexOf()

Busca, en el string, el string que recibe como parámetro.

- Recibe un elemento a buscar en el array.
- Retorna el primer índice donde encontró lo que buscábamos. Si no lo encuentra, retorna un -1.


```
let saludo = '¡Hola! Estamos programando';

saludo.indexOf('Estamos'); // devuelve 7
saludo.indexOf('vamos'); // devuelve -1, no lo encontró
saludo.indexOf('o'); // encuentra la letra 'o' que está en la posición 2, devuelve 2 y corta la ejecución
```

Métodos en Strings

A diferencia de las propiedades, llamamos métodos a las funciones que se encuentran dentro de objetos (los veremos en detalle en las próximas clases).

A estos métodos debemos invocarlos como lo haríamos al llamar una función, con sus paréntesis y parámetros (si fuese necesario).

Métodos en Strings

.slice()

Corta el string y devuelve una parte del string donde se aplica.

- Recibe 2 números como parámetros (pueden ser negativos):
  - El índice desde donde inicia el corte.
  - El índice hasta donde hacer el corte (es opcional).
- Retorna la parte correspondiente al corte.


```
let frase = 'Breaking Bad Rules!';

frase.slice(9,12); // devuelve 'Bad'
frase.slice(13); // devuelve 'Rules!'
frase.slice(-10); // ¿Qué devuelve? ¡A investigar!
```

Métodos en Strings

.trim()

Elimina los espacios que estén al principio y al final de un string.

- No recibe parámetros.
- No quita los espacios del medio.


```
let nombreCompleto = ' Homero Simpson ';
nombreCompleto.trim(); // devuelve 'Homero Simpson'

let nombreCompleto = ' Homero  J.  Simpson ';
nombreCompleto.trim();
// devuelve 'Homero  J.  Simpson'
```

Métodos en Strings

.replace()

Reemplaza una parte del string por otra.

- Recibe dos strings como parámetros:
  - El string que queremos buscar.
  - El string que usaremos de reemplazo.
- Retorna un nuevo string con el reemplazo.


```
let frase = 'Aguante Python!';
frase.replace('Python', 'JS');
// devuelve 'Aguante JS!'
frase.replace('Py', 'JS');
// devuelve 'Aguante JSthon!'
```

Métodos en Strings

.split()

Divide un string en partes.

- Recibe un string que usará como separador de las partes.
- Devuelve un array con las partes del string.


```
let cancion = 'And bingo was his name, oh!';

cancion.split(' ');
// devuelve ['And', 'bingo', 'was', 'his', 'name,'
, 'oh!']

cancion.split(', ');
// devuelve ['And bingo was his name', 'oh!']
```

Métodos en Strings

Si bien cada método realiza una acción muy simple, cuando los combinamos podemos lograr resultados mucho más complejos y útiles.

Propiedades y Métodos en Strings

Ada

Strings

- miTexto.**length**:
  - Devuelve la longitud del texto miTexto.
- miTexto.**indexOf**(textoABuscar):
  - Devuelve el índice donde está el texto a buscar, si no existe devuelve -1.
- miTexto.**slice**(inicio, fin): // inicio y fin son posiciones
  - Devuelve el subtexto de miTexto que empieza en inicio y termina en la posición fin.
- miTexto.**slice**(ini):
  - Devuelve el subtexto de miTexto que empieza en ini hasta el final.
- miTexto.**slice**(-n):
  - Devuelve el subtexto de miTexto que empieza en la posición n contando desde atrás hasta el final.
- miTexto.**trim**():
  - Devuelve miTexto eliminando los espacios que existan al principio y al final.
- miTexto.**split**(","): // cualquier string
  - Devuelve un array con strings a partir de separar miTexto donde existe una ",".
- miTexto.**split**(""): // carácter vacío
  - Devuelve un array con todos los caracteres del texto separados.
- miTexto.**split**(): // sin parámetros
  - Devuelve un array con un único elemento que es miTexto.
- miTexto.**replace**(textoABuscar, nuevoTextoParaReemplazar):
  - Devuelve a miTexto, pero reemplazando textoABuscar por nuevoTextoParaReemplazar (primera vez).
- miTexto.**toUpperCase**():
  - Devuelve miTexto, pero convirtiendo todo a mayúsculas.

Memori Rapido:

length para saber la longitud

indexOf() para saber si existe algo

slice() para tomar una porción de texto

trim() para eliminar los espacios en blanco

split() para generar un array

replace() para cambiar una porción de texto

Veamos mas Ejemplos en VSC!

Momento de poner a prueba lo aprendido!