JavaScript Arrays

Cos'è un Array?

Un array è un tipo di struttura dati che contiene una lista ordinata di valori. Questi valori possono essere di qualsiasi tipo (numeri, stringhe, oggetti, ecc.). La sintassi per creare un array è:

```
let arrayName = [element0, element1, ...];
```

Esempi:

```
const characters = ['Stella', 'Daffodil', 'Gwen'];
let raceWinners = [33, 72, 64];
let myFavorites = ['Bruce', 47032, 'Elena'];
```

La proprietà `.length` restituisce la dimensione dell'array:

```
console.log(characters.length); // Output: 3
```

Accesso agli Elementi

Gli elementi di un array si accedono usando la *bracket notation* (`[]`). Gli indici partono da 0.

Esempio:

```
const rainbowColors = ['Red', 'Orange', 'Yellow'];
let firstColor = rainbowColors[0]; // 'Red'
let lastColor = rainbowColors[2]; // 'Yellow'
```

Modifica degli Array

Puoi modificare gli elementi di un array o aggiungerne di nuovi:

1. Modificare un elemento:

```
let myFavoriteThings = ['Broccoli', 60481];
```

```
myFavoriteThings[0] = 'Celery Root'; // Cambia 'Broccoli' in 'Celery Root'
```

2. Aggiungere un elemento:

```
myFavoriteThings[2] = 'Playgrounds'; // Aggiunge un nuovo elemento
myFavoriteThings.push('Dancing');   // Usa il metodo push per aggiungere alla fine
```

Creazione di Array

Esistono due modi principali per creare un array:

- **Modo consigliato**:

```
let points = [];
```

- **Modo sconsigliato**:

```
let points = new Array();
```

Motivi per evitare `new Array()`:

1. È più lento.
2. Può causare comportamenti incoerenti:

```
let points = new Array(10); // Crea un array con 10 elementi vuoti
let points = [10];          // Crea un array con un solo elemento (10)
```

Cicli con gli Array

Usare i cicli `for` è utile per processare ogni elemento di un array:

Esempio con array:

```
const rainbowColors = ['Red', 'Orange', 'Yellow'];
for (let i = 0; i < rainbowColors.length; i++) {
  console.log(rainbowColors[i]);
}
```

Esempio con stringhe:

```
const rainbowColorsLetters = 'ROYGBIV';
for (let i = 0; i < rainbowColorsLetters.length; i++) {
  console.log(rainbowColorsLetters.charAt(i)); // Preferibile rispetto a []
}
```

Metodi più usati sugli Array

Aggiungere o rimuovere elementi

- `push()` : Aggiunge elementi alla fine.
- `pop()` : Rimuove l'ultimo elemento.
- `shift()` : Rimuove il primo elemento.
- `unshift()` : Aggiunge elementi all'inizio.

Manipolare o estrarre dati

- `splice()` : Modifica l'array rimuovendo, sostituendo o aggiungendo elementi.
- `slice()` : Estrae una porzione dell'array senza modificarlo.
- `join()` : Unisce gli elementi in una stringa.
- `concat()` : Combina più array in uno nuovo.

Ricerca e ordinamento

- `sort()` : Ordina gli elementi.
- `find()` : Trova il primo elemento che soddisfa una condizione.
- `indexOf()` : Restituisce l'indice del primo elemento trovato (o -1 se non esiste).
- `includes()` : Verifica se un valore è presente nell'array.

Metodi per Aggiungere o Rimuovere Elementi

push()

Aggiunge uno o più elementi alla fine dell'array e restituisce la nuova lunghezza dell'array.

```
let frutta = ['Mela', 'Banana'];
frutta.push('Ciliegia', 'Arancia');
console.log(frutta); // Output: ['Mela', 'Banana', 'Ciliegia', 'Arancia']
```

pop()

Rimuove l'ultimo elemento dell'array e restituisce il valore rimosso.

```
let frutta = ['Mela', 'Banana', 'Ciliegia'];
let removed = frutta.pop();
console.log(frutta); // Output: ['Mela', 'Banana']
console.log(removed); // Output: 'Ciliegia'
```

shift()

Rimuove il primo elemento dell'array e restituisce il valore rimosso.

```
let frutta = ['Mela', 'Banana', 'Ciliegia'];
let removed = frutta.shift();
console.log(frutta); // Output: ['Banana', 'Ciliegia']
console.log(removed); // Output: 'Mela'
```

unshift()

Aggiunge uno o più elementi all'inizio dell'array e restituisce la nuova lunghezza dell'array.

```
let frutta = ['Banana', 'Ciliegia'];
frutta.unshift('Mela', 'Pera');
console.log(frutta); // Output: ['Mela', 'Pera', 'Banana', 'Ciliegia']
```

splice()

Modifica l'array rimuovendo, sostituendo o aggiungendo elementi. E' uno dei metodi più importanti!

Sintassi di splice()

La sintassi generale è:

```
array.splice(indice, numeroElementiDaRimuovere, elemento1, elemento2, ...);
```

- **indice**: L'indice a partire dal quale si vuole iniziare a modificare l'array.
- **numeroElementiDaRimuovere**: Il numero di elementi da rimuovere a partire dall'indice specificato. Se impostato a 0, non vengono rimossi elementi.
- **elemento1, elemento2, ...**: Gli elementi da aggiungere all'array.

Esempio di Inserimento

Supponiamo di avere un array `[1, 3, 4, 5]` e di voler inserire il numero `2` alla seconda posizione (indice `1`).

```
let array = [1, 3, 4, 5];
array.splice(1, 0, 2); // Inserisce 2 alla seconda posizione senza rimuovere elementi
console.log(array); // Output: [1, 2, 3, 4, 5]
```

In questo esempio:

- **indice**: `1` (seconda posizione).
- **numeroElementiDaRimuovere**: `0`, quindi non vengono rimossi elementi.
- **elemento**: `2`, che viene inserito alla seconda posizione.

Inserimento con Rimozione

Se vuoi sostituire un elemento esistente, puoi specificare un numero maggiore di `0` per `numeroElementiDaRimuovere`. Ad esempio, per sostituire il secondo elemento con `2`:

```
let array = [1, 3, 4, 5];
array.splice(1, 1, 2); // Rimuove il secondo elemento e lo sostituisce con 2
console.log(array); // Output: [1, 2, 4, 5]
```

In questo caso:

- **indice**: `1`.
- **numeroElementiDaRimuovere**: `1`, quindi viene rimosso un elemento.
- **elemento**: `2`, che sostituisce l'elemento rimosso.

```
let frutta = ['Mela', 'Banana', 'Ciliegia'];
frutta.splice(1, 1, 'Pera', 'Uva');
console.log(frutta); // Output: ['Mela', 'Pera', 'Uva', 'Ciliegia']
```

Metodi per Iterare e Trasformare

forEach()

Esegue una funzione per ogni elemento dell'array.

```
let frutta = ['Mela', 'Banana', 'Ciliegia'];
frutta.forEach(fruit => console.log(fruit));
```

map()

Crea un nuovo array applicando una funzione a ogni elemento dell'array originale.

```
let numbers = [1, 2, 3];
let doubleNumbers = numbers.map(num => num * 2);
console.log(doubleNumbers); // Output: [2, 4, 6]
```

filter()

Crea un nuovo array con tutti gli elementi che soddisfano una condizione.

```
let numbers = [1, 2, 3, 4, 5];
let evenNumbers = numbers.filter(num => num % 2 === 0);
console.log(evenNumbers); // Output: [2, 4]
```

reduce()

Calcola un singolo valore basato sul contenuto dell'array.

```
let numbers = [1, 2, 3, 4, 5];
let sum = numbers.reduce((acc, num) => acc + num, 0);
console.log(sum); // Output: 15
```

Altri Metodi Utili

sort()

Ordina gli elementi dell'array.

```
let numbers = [5, 2, 8, 1];
numbers.sort((a, b) => a - b);
console.log(numbers); // Output: [1, 2, 5, 8]
```

indexOf()

Cerca l'indice del primo elemento che corrisponde al valore specificato.

```
let frutta = ['Mela', 'Banana', 'Ciliegia'];
let index = frutta.indexOf('Banana');
console.log(index); // Output: 1
```

includes()

Verifica se un valore è presente nell'array.

```
let frutta = ['Mela', 'Banana', 'Ciliegia'];
let hasBanana = frutta.includes('Banana');
console.log(hasBanana); // Output: true
```