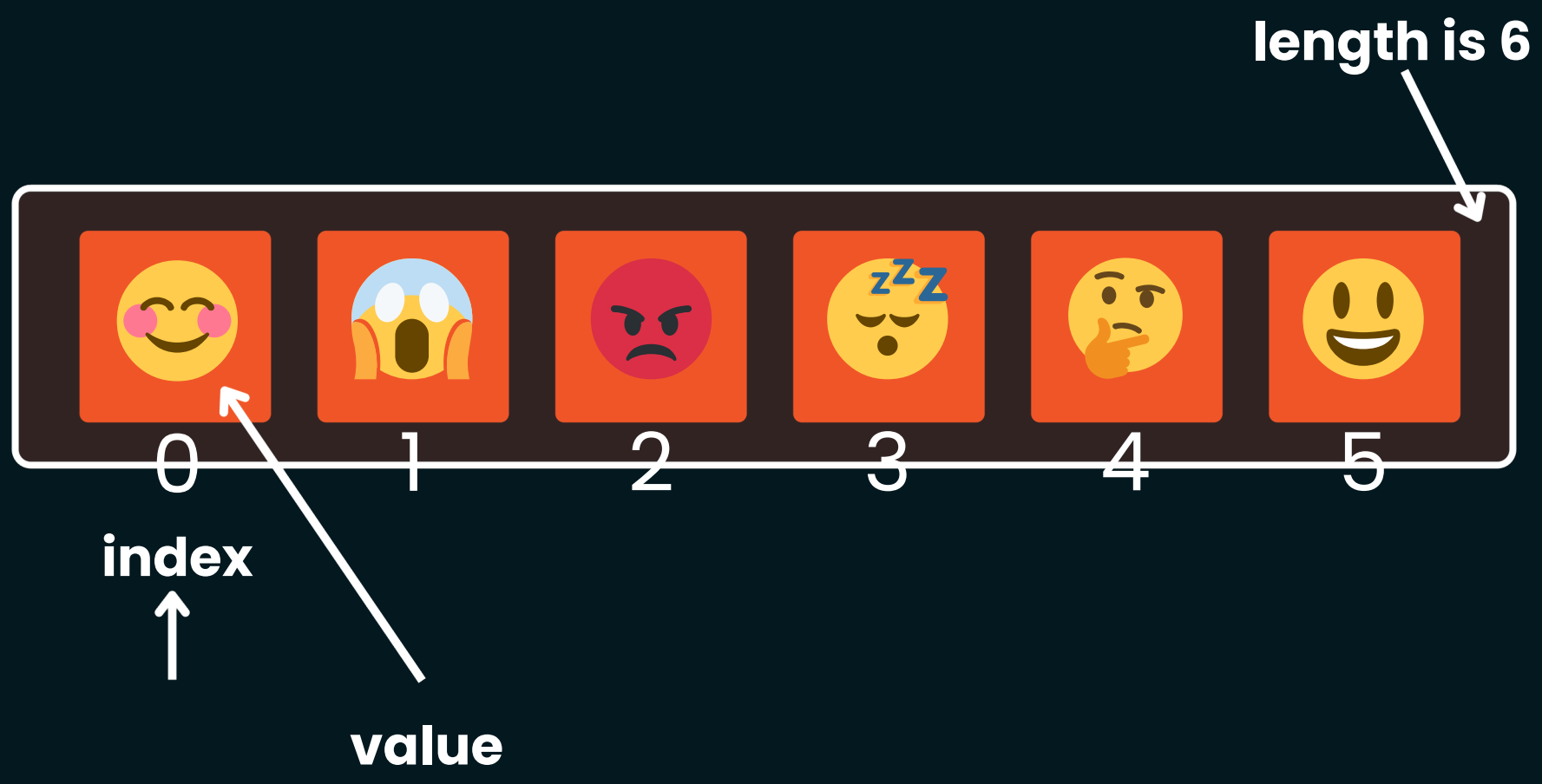
Top 14 JavaScript Array Methods

Every Beginner Must Know



.push()

- **What it does**: Adds elements to the end of an array.
- **Use case**: When you need to expand your array.

index.js

```
let emotions = ["😊", "😱"];
emotions.push("😀");  // Adds "😀" to the end
console.log(emotions);  // ["😊", "😱", "😀"]
```

.pop()

- **What it does**: Removes the last element and returns it.
- **Use case**: When you want to remove or process the last item.

index.js

```
let emotions = ["😊", "😱", "😃"];
let lastEmotion = emotions.pop();  // Removes "😃"
console.log(emotions);  // ["😊", "😱"]
console.log(lastEmotion);  // "😃"
```

.shift()

- **What it does**: Removes the first element and returns it.
- **Use case**: When you need to dequeue or process the first item.

index.js

```
let emotions = ["😊", "😱", "😃"];
let firstEmotion = emotions.shift();  // Removes "😊"
console.log(emotions);  // ["😱", "😃"]
console.log(firstEmotion);  // "😊"
```

.unshift()

- **What it does**: Adds elements to the beginning of an array.
- **Use case**: When you want to prepend items.

index.js

```
let emotions = ["😱", "😃"];
emotions.unshift("😊");  // Adds "😊" to the start
console.log(emotions);  // ["😊", "😱", "😃"]
```

.slice()

- **What it does**: Returns a shallow copy of a portion of an array without modifying the original.
- **Use case**: When you need a subset of an array.

index.js

```
let emotions = ["😊", "😱", "😡", "😃"];
let sliced = emotions.slice(1, 3);  // Extracts elements at index 1 and 2
console.log(sliced);  // ["😱", "😡"]
console.log(emotions);  // ["😊", "😱", "😡", "😃"]
```

.splice()

- **What it does**: Adds, removes, or replaces elements in an array.
- **Use case**: When you need to modify the array directly.

index.js

```
let emotions = ["😊", "😱", "😡"];
emotions.splice(1, 1, "😃");  // Replaces "😱" with "😃"
console.log(emotions);  // ["😊", "😃", "😡"]
```

Note:
In this case, the first 1 is the index where the change starts (the second element), and the second 1 specifies how many elements to remove (one element, "😱"). It then replaces the removed element with "😃", resulting in the array ["😊", "😃"].😡

.map()

- **What it does**: Transforms every element in an array and returns a new array.
- **Use case**: When you want to apply the same function to each item.

index.js

```
let emotions = ["😊", "😱", "😡"];
let excitement = emotions.map(emotion => emotion + "🎉");
console.log(excitement);  // ["😊🎉", "😱🎉", "😡🎉"]
```

.filter()

- **What it does**: Filters the array based on a condition and returns a new array.
- **Use case**: When you want only specific items.

index.js

```
let emotions = ["😊", "😱", "😡", "😃"];
let happyEmotions = emotions.filter(emotion =>
  emotion === "😊" || emotion === "😃"
);
console.log(happyEmotions);  // ["😊", "😃"]
```

.reduce()

- **What it does**: Reduces the array to a single value by applying a callback function.
- **Use case**: When you need a summary value (e.g., concatenation).

index.js

```
let emotions = ["😊", "😱", "😃"];
let allEmotions = emotions.reduce(
  (acc, emotion) => acc + emotion, ""
);
console.log(allEmotions);  // "😊😱😃"
```

.forEach()

- **What it does**: Executes a provided function once for each array element.
- **Use case**: When you want to iterate through an array but don't need a new array.

index.js

```
let emotions = ["😊", "😱", "😡"];
emotions.forEach(emotion => console.log(emotion));
// Logs: "😊", "😱", "😡"
```

.find()

- **What it does**: Returns the first element that satisfies a condition.
- **Use case**: When you’re looking for one specific item.

index.js

```
let emotions = ["😊", "😱", "😃"];
let firstHappy = emotions.find(emotion => emotion === "😊");
console.log(firstHappy);  // "😊"
```

.findIndex()

- **What it does**: Returns the index of the first element that satisfies a condition.
- **Use case**: When you need the position of an item.

index.js

```
let emotions = ["😊", "😱", "😃"];
let index = emotions.findIndex(emotion => emotion === "😃");
console.log(index);  // 2
```

.includes()

- **What it does**: Checks if an array includes a certain value.
- **Use case**: When you want to verify the existence of an item.

index.js

```
let emotions = ["😊", "😱", "😡"];
console.log(emotions.includes("😊"));  // true
console.log(emotions.includes("😀"));  // false
```

.sort()

- **What it does**: Sorts the elements of an array in place.
- **Use case**: When you need a sorted array.

index.js

```
let numbers = [3, 1, 4, 2];
numbers.sort();
console.log(numbers);  // [1, 2, 3, 4]

let characters = ['b', 'd', 'a', 'c'];
characters.sort();
console.log(characters);  // ['a', 'b', 'c', 'd']
```

Hopefully You Found It Usefull!

Be sure to save this post so you can come back to it later

like Comment Share