Theory: ForEach method

21 minutes 0 / 5 problems solved

You already know how to initialize an array and reference its elements one by one. To become more confident with arrays, you also need to know how to conveniently perform multiple identical operations with the elements. That's what the `forEach` method does: it allows you to perform the same action for the array elements without having to access each of them separately.

§1. What is forEach

Let's say you have an array of names: `arrayNames = ["Mike", "Alex", "Asya"]`. You want to show each element through the `console.log`. As you remember, you can do this using the index:

```
1 console.log(arrayNames[0]);
2 console.log(arrayNames[1]);
```

Now let's imagine that you have an array of a hundred elements, or you don't even know how many elements there are. How do you print all the elements in this case?

The best way to do this is the **forEach** method. It reads each element of the array sequentially and performs the same action with it. We can use it to display each element of the array via the `console.log`:

```
1 arrayNames.forEach(console.log);
```

As a result, `console.log` first shows the name "Mike", then "Alex", and finally, "Asya".

The main feature of `forEach` is the sequential processing of each array element. It reads the elements one by one and can't go any further without performing the necessary actions with the previous one.

The method skips the elements that were deleted or not initialized. For example, if we change `arrayNames` to `arrayNames = ["Mike",, "Asya"],` we will only see two strings, "Mike" and "Asya".

§2. Syntax

In the previous example, we passed the `console.log` function to the `forEach` method, but you can also replace it with a custom function. The `forEach` method has two arguments:

1. *callback* — a function to be executed for each element (for example, `console.log` like in the previous case). Callback has three parameters:

- *currentValue:* the element we are currently processing;
- *index*: the index of the *currentValue* element;
- *array*: the array for which we called `forEach`.

2. *thisArg* — a value that we can use as *this* in the callback function.

Here is an example:

```
1 const arrayFruit = ["pineapples", "oranges", "apples"];
2 arrayFruit.forEach(function(value) {
3   console.log("Today I ate" + value);
4 });
```

The result will be:

```
"Today I ate pineapples"
"Today I ate oranges"
"Today I ate apples"
```

We used the first argument (*currentValue*) inside the *callback* to display the value of the current element. As a result, we have three different phrases, the last words of which correspond to the elements of the array Fruit.

§3. Examples

You can rewrite the previous code using a named callback function:

```
function showText(value) {
  console.log("Today I eat " + value);
}
arrayFruit.forEach(showText);
```

The result will be the same.

We can also use other callback arguments:

```
function showItem(item, index, array) {

  console.log("My value is " + item + ". I’m the " + index + " element of array "
+ array);
}
arrayFruit.forEach(showItem);
```

The result will be:

```

"My value is pineapples. I'm the 0 element of array pineapples,oranges,apples"

"My value is oranges. I'm the 1 element of array pineapples,oranges,apples"
"My value is apples. I'm the 2 element of array pineapples,oranges,apples"
```

This way, you can use not only the current element inside the callback function, but also its index and the array itself.

If you want to specify *custom this* in the *callback* function, you must use the second `forEach` argument:

```
const customThis = {
  value: 10
}

function showThisValue() {
  console.log(this.value);
}

arrayFruit.forEach(showThisValue, customThis);
```

The result will be 10.

§4. Conclusion

In this topic, we have considered a way to iterate over an array using the `forEach` method. It has a callback function that sequentially processes each array value and allows you to work with the value, the index, and the array itself inside the callback function. We also considered how to set a custom `this` inside the callback function using the second `forEach` argument. Now it's time for practice!

Report a typo

17 users liked this theory. **3** didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion