Frontendᵝ → JavaScript → Functions → Anonymous function

Theory: Anonymous function

19 minutes 0 / 5 problems solved Skip this topic Start practicing

110 users solved this topic. Latest completion was 2 days ago.

In the previous topic, you have learned how to work with functions. In our brief review, we omitted an important detail: some functions can be anonymous. Using such functions allows you to write shorter code and makes your program more readable. Let's learn more about anonymous functions and their advantages.

§1. What is an anonymous function?

Let's remember a common way to define a function:

```
function myFunction() {
  // ...
}

myFunction();
```

However, it's also possible to rewrite the code above in a different way:

```
const myFunction = function() {
  //...
}

myFunction();
```

As you've noticed, we saved the function in the variable `myFunction` and didn't use any function name between the `function` keyword and parentheses. It means that we used an **anonymous function**, that is, a function declared without specifying its name.

There are some nuances to anonymous functions. When you define a named function, you can call it in any part of your code because the browser creates a reference to this function before calling it. So it doesn't matter if you place the function definition before or after calling it: the program will still work fine.

```
// this code works

setPrice = function setPrice() {
  // ...
}

setPrice();

// this code works too

setColor();

const setColor = function setColor() {
 // ...
}
```

However, this is not the case with anonymous functions. These functions are created when they are called. Thus, you can only access an anonymous function after it has been defined.

Current topic:

Anonymous function

Topic depends on:

- Arrow functions
- setTimeout and setInterval
- Comparison operators
- "then", "catch" and "finally" methods
- The "addEventListener" method

Table of contents:

```
// this code works

const setPrice = function() {
  // ...
}

setPrice();

// this code doesn't work


setColor();


const setColor = function() {

  // ...

}
```

§2. Anonymous functions as parameters

Let's take a closer look at situations where you might need anonymous functions. You do not need to fully understand how the examples will work. We will study it in more detail in the next topics, but for now your goal is to remember what anonymous functions can look like. One of the most common cases is passing anonymous functions as arguments to other functions. Here is an example:

```
setTimeout(function () {
  console.log("Timer has finished!")
}, 5000);
```

As a special case, anonymous functions are very popular for DOM iterations:

```
document.getElementById("form").addEventListener("click", function() {
  // ...
})
```

Anonymous functions are also useful in promises when you need to pass functions for a success and failure argument:

```
promise.then(
    function (response) {
        console.log(response);
    },
    function (error) {
        console.log(error);
    }
);
```

Programmers like to apply anonymous functions as arguments because they're short.

§3. Immediately invoked anonymous functions

There is another common way to use anonymous functions. Sometimes you need to define a function and call it immediately. It makes sense to use a function without a name for cases when you do not need to call it again.

```
(function() {
 console.log("I've been launched!");
}());
```

We wrapped the anonymous function in parentheses and initiated the function call by adding `()` at the end. This function will be executed as soon as the browser reads this code. The main idea is to use local variables inside an anonymous function without affecting variables with the same name located outside of this function. This approach works well for plugins and other additional developer tools.

You can also use arguments immediately when calling an anonymous function:

```
const text = "I've been launched!";

(function(text) {
 console.log(text);
}(text));
```

As before, you will see the text `"I've been launched!"` in the console.

§4. Conclusion

Anonymous functions are functions without a name. They are very convenient mostly due to their conciseness. The most common application of the anonymous functions is passing them as an argument and immediately calling them. Developers regularly use anonymous functions to create modern web applications.

Report a typo

15 users liked this theory. 1 didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

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