JavaScript Anonymous Functions

Summary: in this tutorial, you will learn about JavaScript anonymous functions which are functions without names.

Introduction to JavaScript anonymous functions

An anonymous function is a function without a name. The following shows how to define an anonymous function:

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

Note that if you don't place the anonymous function inside the parentheses `()`, you'll get a syntax error. The parentheses `()` make the anonymous function an expression that returns a function object.

An anonymous function is not accessible after its initial creation. Therefore, you often need to assign it to a variable.

For example, the following shows an anonymous function that displays a message:

```
let show = function() {
    console.log('Anonymous function');
};

show();
```

In this example, the anonymous function has no name between the `function` keyword and parentheses `()`.

Because we need to call the anonymous function later, we assign the anonymous function to the `show` variable.

Since the whole assignment of the anonymous function to the show variable makes a valid expression, you don't need to wrap the anonymous function inside the parentheses `()`.

Using anonymous functions as arguments

In practice, you often pass anonymous functions as arguments to other functions. For example:

```
setTimeout(function() {
    console.log('Execute later after 1 second')
}, 1000);
```

In this example, we pass an anonymous function into the `setTimeout()` function. The `setTimeout()` function executes this anonymous function one second later.

> Note that functions are first-class citizens in JavaScript. Therefore, you can pass a function to another function as an argument.

Immediately invoked function execution

If you want to create a function and execute it immediately after the declaration, you can declare an anonymous function like this:

```
(function() {
    console.log('IIFE');
})();
```

Output:

```
IIFE
```

How it works.

First, define a function expression:

```
(function () {
    console.log('Immediately invoked function execution');
})
```

This expression returns a function.

Second, call the function by adding the trailing parentheses `()` :

```
(function () {
    console.log('Immediately invoked function execution');
})();
```

Sometimes, you may want to pass arguments into an anonymous function, like this:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

(function () {
    console.log(person.firstName + ' ' + person.lastName);
})(person);
```

Output:

```
John Doe
```

Arrow functions

ES6 introduced arrow function expressions that provide a shorthand for declaring anonymous functions.

For example, this function:

```
let show = function () {
    console.log('Anonymous function');
};
```

... can be shortened using the following arrow function:

```
let show = () => console.log('Anonymous function');
```

Similarly, the following anonymous function:

```
let add = function (a, b) {
    return a + b;
};
```

... is functionally equivalent to the following arrow function:

```
let add = (a, b) => a + b;
```

Summary

- Anonymous functions are functions without names.

- Use an anonymous functions can be arguments of other functions or as an immediately invoked function execution.

Quiz