21st October 2019

# Arrow functions revisited

Let's revisit arrow functions.

Arrow functions are not just a "shorthand" for writing small stuff. They have some very specific and useful features.

JavaScript is full of situations where we need to write a small function that's executed somewhere else.

For instance:

- `arr.forEach(func)` – `func` is executed by `forEach` for every array item.
- `setTimeout(func)` – `func` is executed by the built-in scheduler.
- …there are more.

It's in the very spirit of JavaScript to create a function and pass it somewhere.

And in such functions we usually don't want to leave the current context. That's where arrow functions come in handy.

## Arrow functions have no "this"

As we remember from the chapter Object methods, "this", arrow functions do not have `this`. If `this` is accessed, it is taken from the outside.

For instance, we can use it to iterate inside an object method:

```
let group = {
  title: "Our Group",
  students: ["John", "Pete", "Alice"],

  showList() {
    this.students.forEach(
      student => alert(this.title + ': ' + student)
    );
  }
};

group.showList();
```

Here in `forEach`, the arrow function is used, so `this.title` in it is exactly the same as in the outer method `showList`. That is: `group.title`.

If we used a "regular" function, there would be an error:

```
let group = {
  title: "Our Group",
  students: ["John", "Pete", "Alice"],

  showList() {
    this.students.forEach(function(student) {
      // Error: Cannot read property 'title' of undefined
      alert(this.title + ': ' + student)
    });
  }
};

group.showList();
```

The error occurs because `forEach` runs functions with `this=undefined` by default, so the attempt to access `undefined.title` is made.

That doesn't affect arrow functions, because they just don't have `this`.

**Arrow functions can't run with `new`**

Not having `this` naturally means another limitation: arrow functions can't be used as constructors. They can't be called with `new`.

**Arrow functions VS bind**

There's a subtle difference between an arrow function `=>` and a regular function called with `.bind(this)`:

- `.bind(this)` creates a "bound version" of the function.
- The arrow `=>` doesn't create any binding. The function simply doesn't have `this`. The lookup of `this` is made exactly the same way as a regular variable search: in the outer lexical environment.

# Arrows have no "arguments"

Arrow functions also have no `arguments` variable.

That's great for decorators, when we need to forward a call with the current `this` and `arguments`.

For instance, `defer(f, ms)` gets a function and returns a wrapper around it that delays the call by `ms` milliseconds:

```
function defer(f, ms) {
  return function() {
    setTimeout(() => f.apply(this, arguments), ms)
  };
}

function sayHi(who) {
  alert('Hello, ' + who);
}

```

```
let sayHiDeferred = defer(sayHi, 2000);
sayHiDeferred("John"); // Hello, John after 2 seconds
```

The same without an arrow function would look like:

```
function defer(f, ms) {
  return function(...args) {
    let ctx = this;
    setTimeout(function() {
      return f.apply(ctx, args);
    }, ms);
  };
}
```

Here we had to create additional variables `args` and `ctx` so that the function inside `setTimeout` could take them.

## Summary

Arrow functions:

- Do not have `this`
- Do not have `arguments`
- Can't be called with `new`
- They also don't have `super`, but we didn't study it yet. We will on the chapter Class inheritance

That's because they are meant for short pieces of code that do not have their own "context", but rather work in the current one. And they really shine in that use case.

Previous lesson Next lesson

Share Tutorial map

## Comments

- If you have suggestions what to improve - please submit a GitHub issue or a pull request instead of commenting.
- If you can't understand something in the article – please elaborate.
- To insert a few words of code, use the `<code>` tag, for several lines – use `<pre>`, for more than 10 lines – use a sandbox (plnkr, JSBin, codepen…)