Arrow functions were introduced in `ES6`.

Arrow functions allow us to write shorter function syntax:

```
let myFunction = (a, b) => a * b;
```

Before Arrow

```
hello = function () {
  return "Hello World!";
};
```

With Arrow Function

```
hello = () => {
  return "Hello World!";
};
```

Arrow Functions Return Value by Default

If the function has only one statement, and the statement returns a value, you can remove the `brackets` and the `return` keyword.

```
hello = () => "Hello World!";
```

Note: This works only if the function has only one statement.

Arrow Function With Parameters

If you have parameters, you pass them inside the parentheses

```
hello = (val) => "Hello " + val;
```

Arrow Function Without Parentheses

In fact, if you have only one parameter, you can skip the parentheses as well:

```
hello = (val) => "Hello " + val;
```

Arrow Function Without Parentheses

```
hello = (val) => "Hello " + val;
```

What About `this`?

The handling of `this` is also different in arrow functions compared to regular functions.

In short, with arrow functions there are no binding of `this`.

In regular functions the `this` keyword represented the object that called the function, which could be the window, the document, a button or whatever.

With arrow functions the `this` keyword always represents the object that defined the arrow function.

Let us take a look at two examples to understand the difference.

Both examples call a method twice, first when the page loads, and once again when the user clicks a button.

The first example uses a regular function, and the second example uses an arrow function.

The result shows that the first example returns two different objects (window and button), and the second example returns the window object twice, because the window object is the "owner" of the function.