howest
hogeschool

JavaScript: Arrow functions and the ‘this’ keyword

Web, Mobile and Security

Frédéric Vlummens

Agenda

- Arrow functions
- The ‘this’ keyword in Java vs JavaScript

() => { }

howest hogeschool

Different ways of defining a function

- Different ways:

```
// function declaration
function sum1(a, b) {
    return a + b;
}

// function expression
let sum2 = function(a, b) {
    return a + b;
}

// arrow function
let sum3 = (a, b) ⇒ a + b;
```

- As we can see, an arrow function allows for shorter syntax
- Let’s take a look at them in more detail…

howest hogeschool

Basic syntax of an arrow function

- **Basic syntax:**
  $(param_1, param_2, \ldots, param_N) \Rightarrow \{ \text{statements} \}$
  $(param_1, param_2, \ldots, param_N) \Rightarrow \text{expression}$
  $(param_1, param_2, \ldots, param_N) \Rightarrow \{ \text{return expression;} \}$

- **Parentheses are optional if only one parameter:**
  param => expression
  param => { statements }

- **If no parameters, use ():**
  () => expression
  () => { statements }

Arrow functions: some more examples

```
// regular function definition
function avg1(a, b) {
    return (a + b) / 2;
}

// arrow function with expression
let avg2 = (a, b) ⇒ (a + b) / 2;

// arrow function with return statement
let avg3 = (a, b) ⇒ { return (a + b) / 2; };
```


howest hogeschool

Arrow functions: some more examples

```
// regular function definition
function hi1() {
    return "hello world world";
}

// arrow function with expression, no params
let hi2 = () ⇒ "hello world";

// arrow function with return statement, no params
let hi3 = () ⇒ { return "hello world"; }
```


howest hogeschool

Arrow functions: why use them?

- Shorter syntax
- No own binding of ‘this’ (see later)
- Very useful in combination with higher order functions, timers and intervals (see later)
- Arrow functions allow for elegant and shorter code, if used wisely.

The ‘this’ keyword

- In **Java**:

```
Person p = new Person("Frédéric", 39);
```

```
public class Person {

    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }
```

howest hogeschool

The ‘this’ keyword

- In **Java**:

```
Person p = new Person("Frédéric", 39);
```

```
public class Person {

    private String name;
    private int age;

    public String getName() {
        return this.name;
    }
```

Within an instance method or a constructor, this is a reference to the current object — the object whose method or constructor is being called. You can refer to any member of the current object from within an instance method or a constructor by using this.

howest hogeschool

The ‘this’ keyword

- And in **JavaScript**?
- It depends…

howest hogeschool

The ‘this’ keyword in JavaScript

- In global context (i.e. outside of a function)
- References the window object

```
"use strict";

console.log(this);  // Window

this.a = "hello";

console.log(this.a);   // "hello"
console.log(window.a); // "hello"
```


howest hogeschool

The ‘this’ keyword in JavaScript

- In function context (i.e. inside a function)
- Different behavior if strict mode enabled

```
"use strict";

function func1() {
    console.log(this);
}

func1();    // undefined
```

```
// "use strict";

function func1() {
    console.log(this);
}

func1();    // Window
```

- In strict mode, **this** by default has the value undefined (in non-strict: global scope)

howest hogeschool

The ‘this’ keyword in JavaScript

- As always, make sure to enable strict mode!
- Avoids you from unwantedly meddling with global scope.

howest hogeschool

The ‘this’ keyword in JavaScript

- In object context
- Comparable to Java
- Within an object method, references the object itself

```
"use strict";

let person = {
    name: "John",
    age: 39,
    sleep: function() {
        console.log(this);
        console.log("zzzZZZ");
    }
}

person.sleep(); // Object
                // zzzZZZ
```


howest hogeschool

The ‘this’ keyword in JavaScript

- As an event handler
- **this** references the element that fires (=causes) the event)

```
<button>Click me</button>
```

```
"use strict";

document.addEventListener("DOMContentLoaded", init);

function init() {
    document.querySelector("button").addEventListener("click", buttonClick);
}

function buttonClick() {
    console.log(this);  // button
}
```


howest hogeschool

The ‘this’ keyword in JavaScript

- Arrow functions do not have their own **this**
- Base rule: they take over the **this** from the enclosing scope
- Arrow function defined in global scope, has **this** referring to global object, **even in strict mode.**

```
let this1 = () ⇒ { return this; };

function wrapper() {
    let this2 = () ⇒ { return this; };
    return this2();
}

console.log(this1());    // Window
console.log(wrapper()); // undefined
```


howest hogeschool

Questions?

howest hogeschool