"this" - Summary

The this keyword can lead to some headaches in JavaScript - this summary hopefully acts as a remedy.

this refers to different things, depending on where it's used and how (if used in a function) a function is called.

Generally, this refers to the "thing" which called a function (if used inside of a function). That can be the global context, an object or some bound data/ object (e.g. when the browser binds this to the button that triggered a click event).

1) this in Global Context (i.e. outside of any function)

```
function something() { ... }

console.log(this); // logs global object (window in browser) - ALWAYS (also in strict mode)!
```

2) this in a Function (non-Arrow) - Called in the global context

```
function something() {
    console.log(this);
}

something(); // logs global object (window in browser) in non-strict mode, undefined in strict mode
```

3) this in an Arrow-Function - Called in the global context

```
const something = () => {
    console.log(this);
}

something(); // logs global object (window in browser) - ALWAYS (also in strict mode)!
```

4) this in a Method (non-Arrow) - Called on an object

```
const person = {
    name: 'Max',
    greet: function() { // or use method shorthand: greet() { ... }
        console.log(this.name);
    }
};

person.greet(); // logs 'Max', "this" refers to the person object
```

5) this in a Method (Arrow Function) - Called on an object

```
const person = {
    name: 'Max',
    greet: () => {
```

```
        console.log(this.name);
    }
};

person.greet(); // logs nothing (or some global name on window object), "this" refers to global (window) object, even in strict mode
```

this can refer to unexpected things if you call it on some other object, e.g.:

```
const person = {
    name: 'Max',
    greet() {
        console.log(this.name);
    }
};

const anotherPerson = { name: 'Manuel' }; // does NOT have a built-in greet method!

anotherPerson.sayHi = person.greet; // greet is NOT called here, it's just assigned to a new property/ method on the "anotherPerson" object

anotherPerson.sayHi(); // logs 'Manuel' because method is called on "anotherPerson" object => "this" refers to the "thing" which called it
```

If in doubt, a `console.log(this);` can always help you find out what `this` is referring to at the moment!