`tt()`

Schedule

1. Workshop OOP
   (Object Oriented Programming)
2. Terminology in development
3. Continue working on concept + questions

Schedule

1. **Workshop OOP (Object Oriented Programming)**
2. Terminology in development
3. Continue working on concept + questions

OOP workshop

Schedule

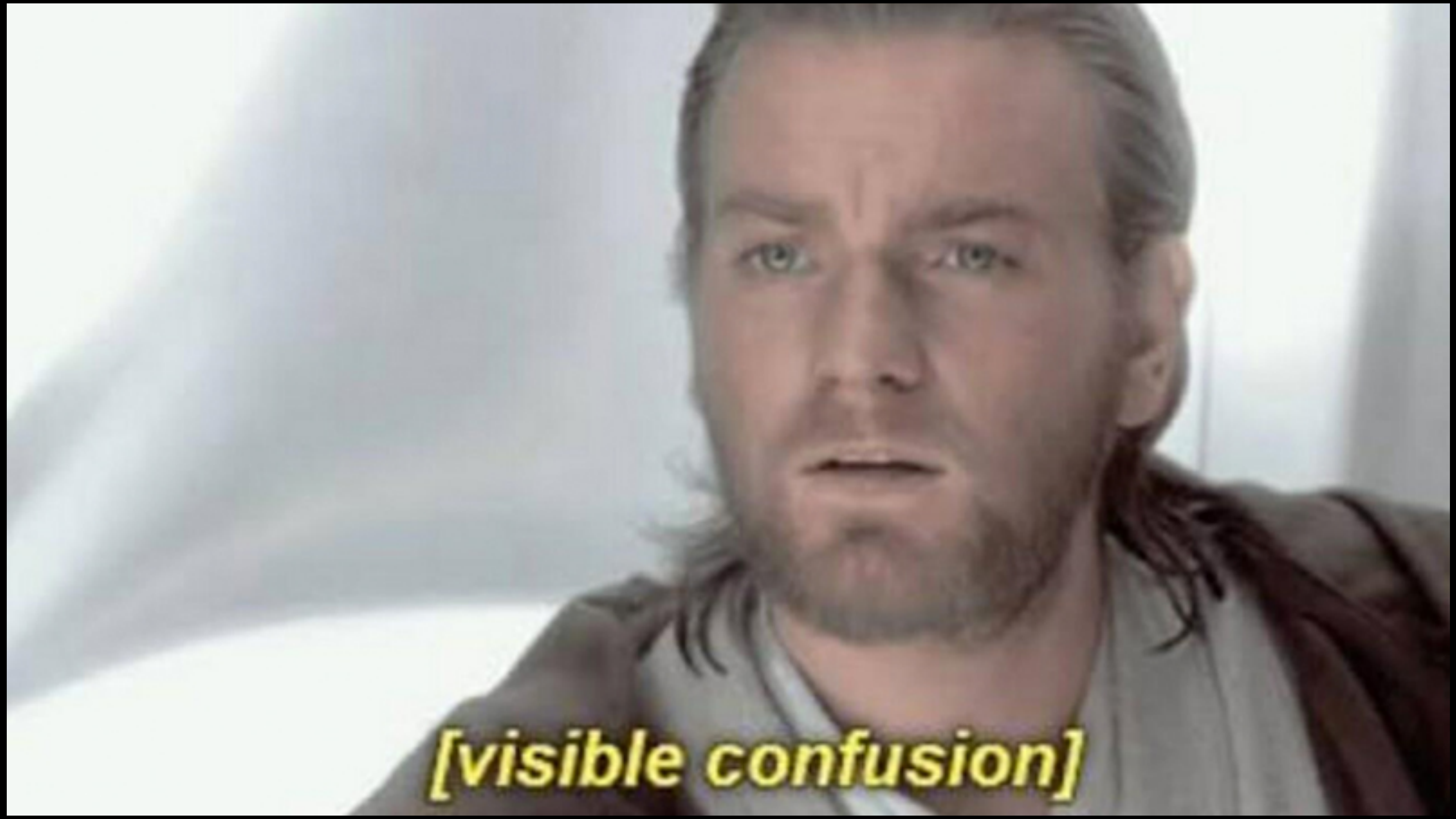
1. Workshop OOP
   (Object Oriented Programming)

2. **Terminology in development**

3. Continue working on concept + questions

Terminology

Terminology



Modules

Functions

Frameworks

Libraries

Classes??

Tools

?????

Modules

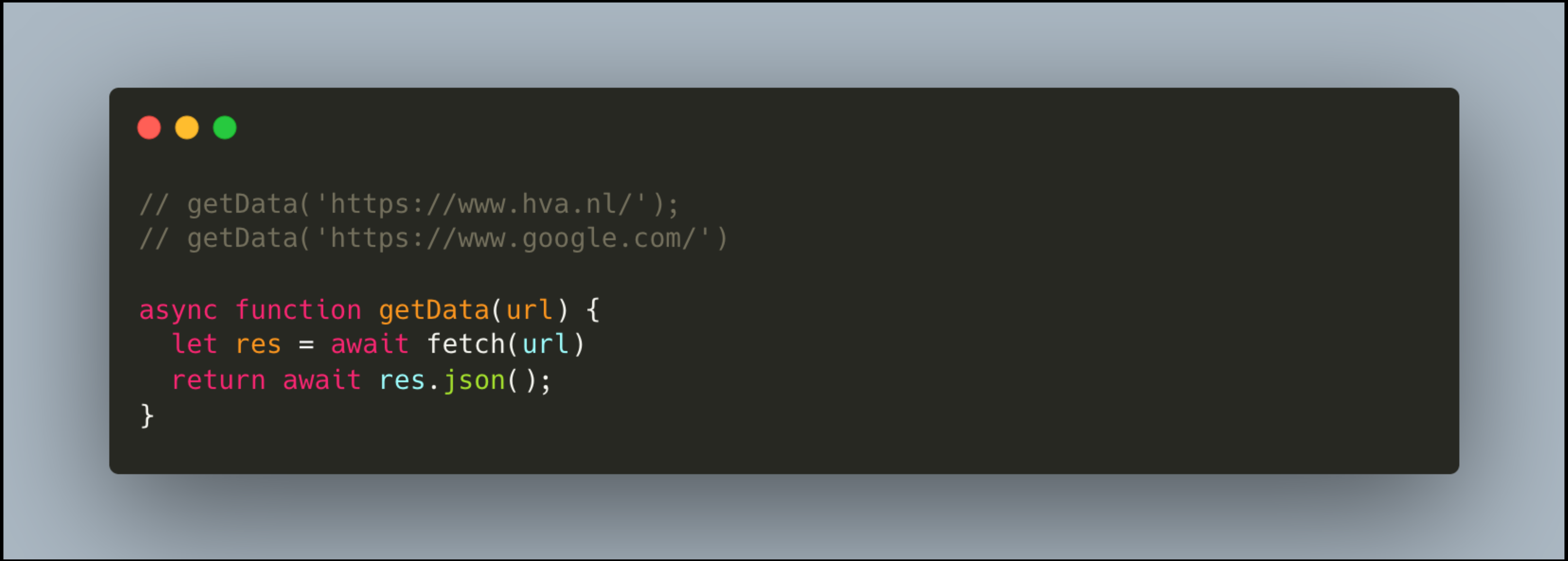
Modules are a way to divide our codebase into seperate files. They're a build-in functionality of JavaScript. We use them as a means to **modularise our code.** This enables us to invoke the **separation of concerns** principles.

You may remember this from earlier courses, and is the same reason we separate HTML, CSS and JavaScript

Functions

Functions enable functionality (no way). We use functions inside modules to handle individual tasks. We seek to use functions only for a single task in order to make them easily reusable. For instance, the function getData() can grab multiple sources of data, depending on the provided URL.

Functions



```javascript
// getData('https://www.hva.nl/');
// getData('https://www.google.com/')

async function getData(url) {
  let res = await fetch(url)
  return await res.json();
}
```

Frameworks

Frameworks change how we deal with our code. They call us, we don’t call them. This principle is known as **Inversion of Control.** In frameworks, all the data is being guided through the framework, we merely ask it to use specific functions in order to get what we want.

Frameworks

Frameworks

All of these (at least in their current form) will be obsolete in the next 3-5 years.

Frameworks

It’s better to learn vanilla code, as it enables you to switch between framework whenever the need requires it.

This is also why we hammer on documentation skills, both writing and reading it.

Libraries

Libraries are just a collection of related functionality.

This is why D3 is a **library**, as it contains functions that enable us to convert data into SVGs.

React/Vue/Svelte/Whatever differs in this as it takes over our entire application, and we write code to complement it.

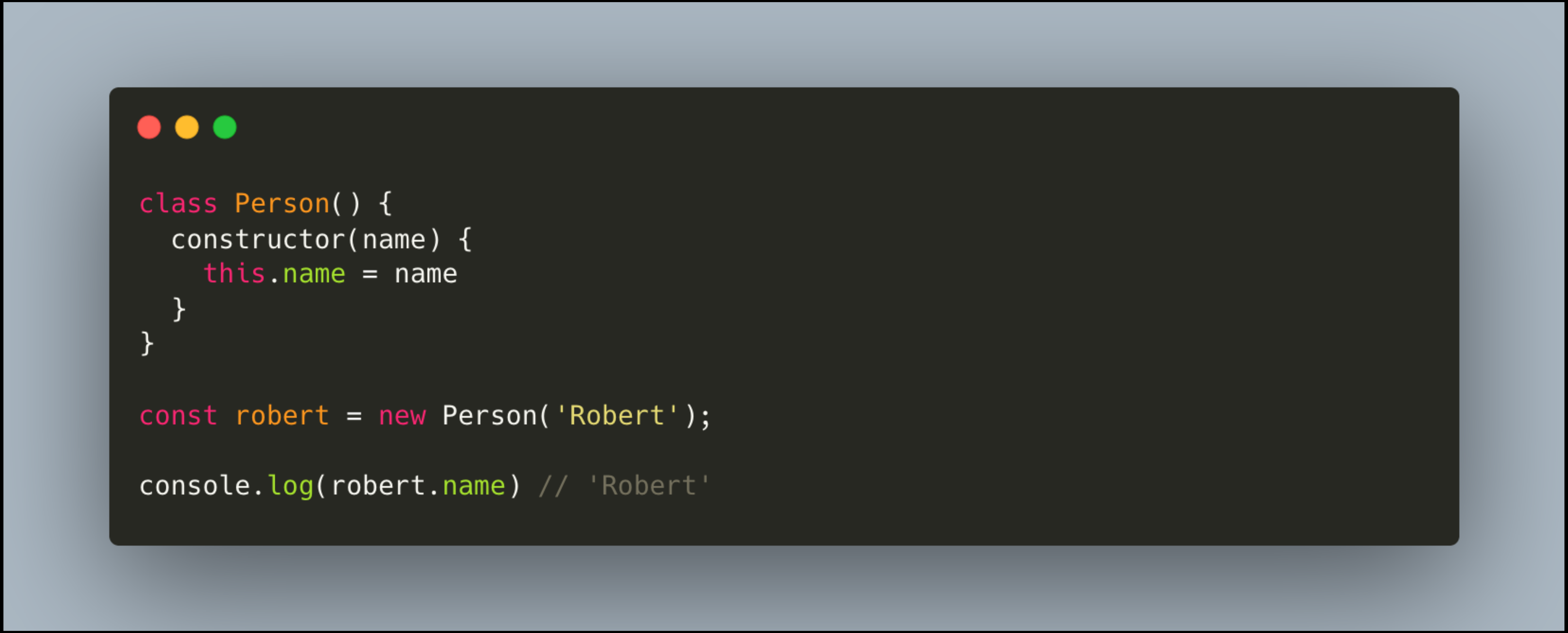
(This is why combining them is so hard, as both React and D3 wish to play with the data we provide!)

Classes

Classes are a special kind of Objects. They’re usually made by people that create libraries or modules.

Classes are **initialised** by importing or creating them, using the *new* or *import* keywords, an example:

Classes

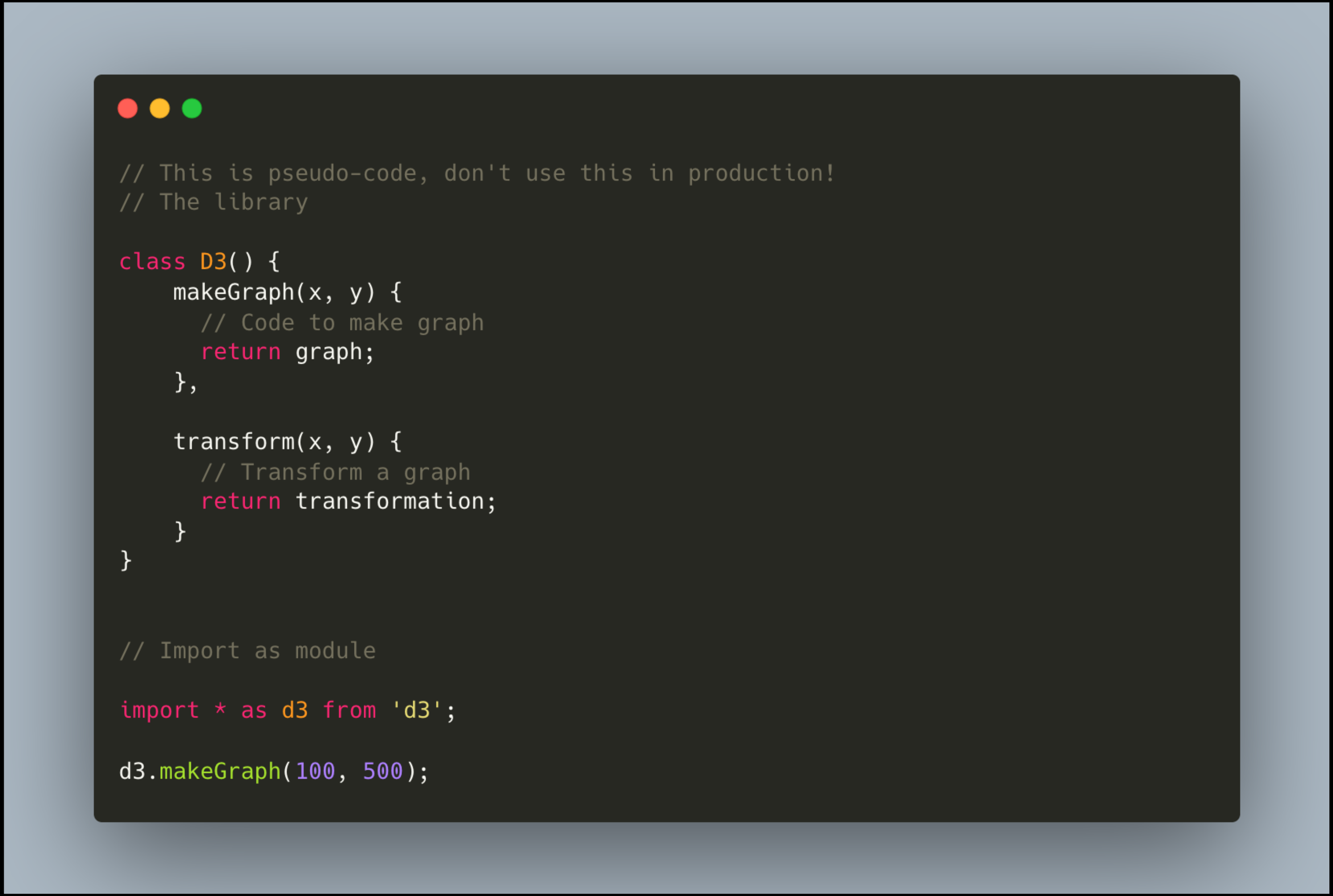


```
class Person() {
  constructor(name) {
    this.name = name
  }
}

const robert = new Person('Robert');

console.log(robert.name) // 'Robert'
```

Classes



```
// This is pseudo-code, don't use this in production!
// The library

class D3() {
    makeGraph(x, y) {
      // Code to make graph
      return graph;
    },

    transform(x, y) {
      // Transform a graph
      return transformation;
    }
}


// Import as module

import * as d3 from 'd3';

d3.makeGraph(100, 500);
```

And tools?

Tools have no direct visible result.

We use tools to optimise our codebase and workflow

ESlint is a tool to validate our JavaScript

Vite is a tool to bundle our code before production

Sass is a tool to process our .scss into .css

Conclusion

Nice, in case this GIF doesn’t work.

Schedule

1. Workshop OOP
   (Object Oriented Programming)

2. Terminology in development

3. **Continue working on concept + questions**