tt()

Schedule

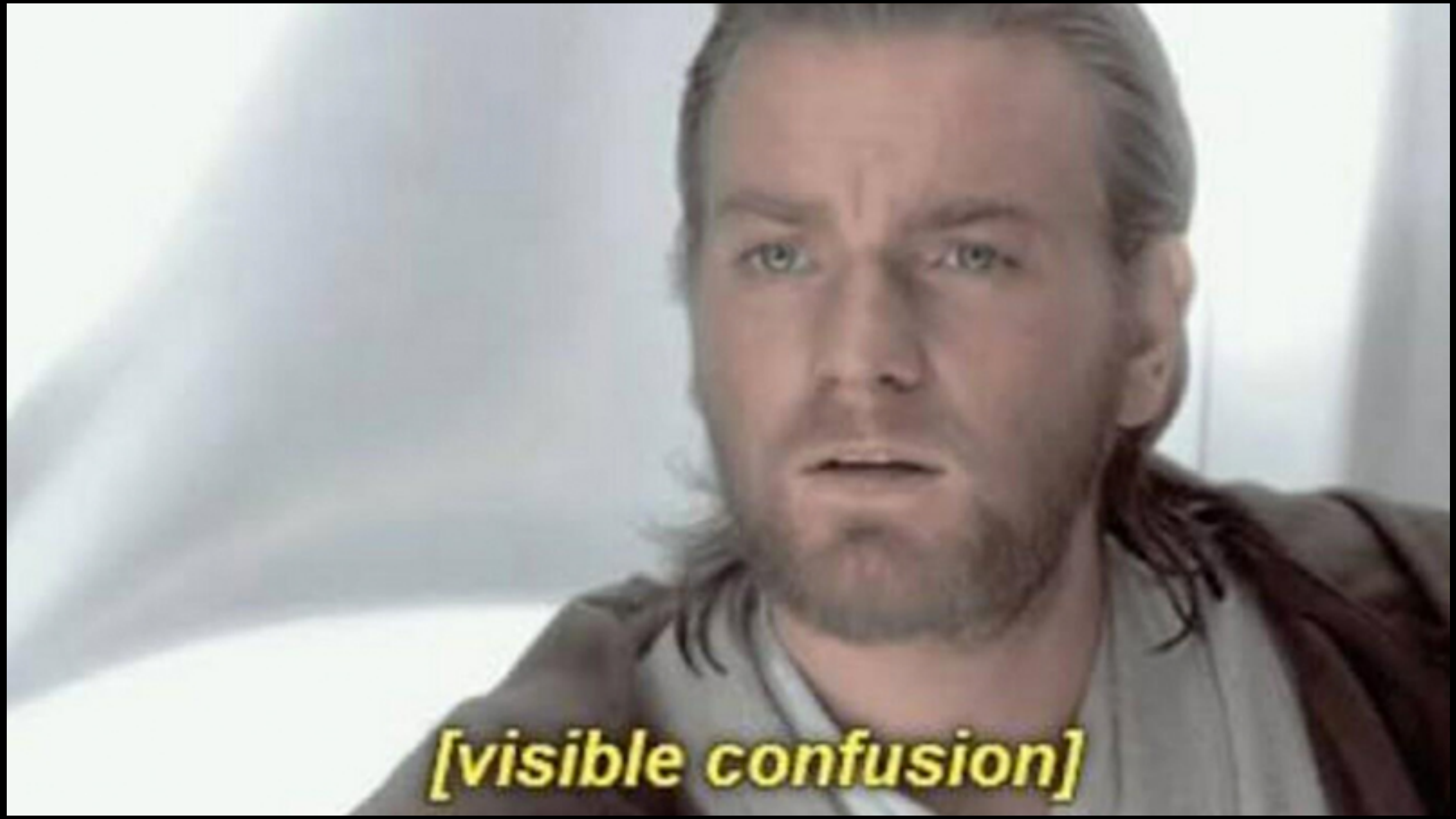
1. Terminology in development
2. Formats for data transfer
3. Continue working on concept + questions

Schedule

1. **Terminology in development**
2. Formats for data transfer
3. Continue working on concept + questions

Terminology

Terminology



Tools

Frameworks

Functions

Scoping

Classes

Libraries

Components

Modules

(and Pokémons)

Tools

Tools have no direct visible result.

Developers use tools to optimise their workflow

ESlint is a tool to validate JavaScript

Vite is a tool to bundle code before production

Sass is a tool to process.scss into .css

Frameworks

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

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

Frameworks

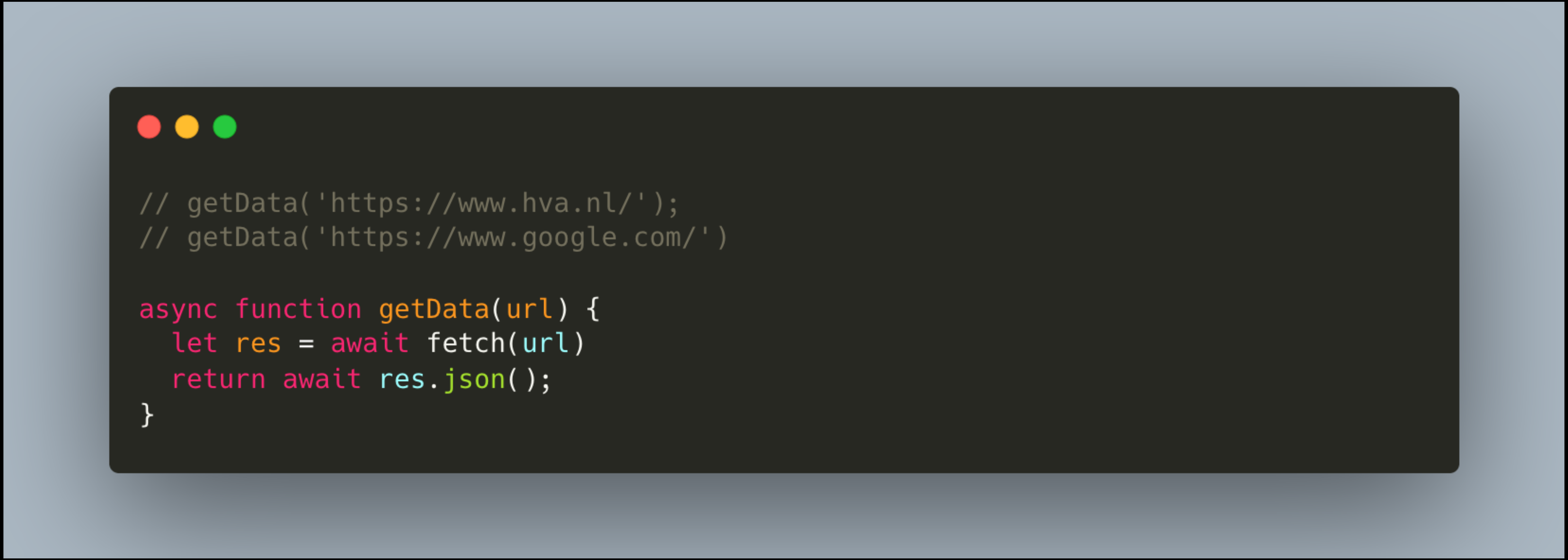
It’s better to learn vanilla code, as it enables you to switch between framework whenever the need requires it.

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

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



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

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

Scoping / scope / scoped

The scope of a variable describes where the variable is available, where it “lives”.

Svelte is scoped: in Svelte components, variables only “live” inside the component.

Variables and constants in ES6 are block scoped

Scope

- The **scope** of a variable refers to where the variable is accessible

```
function groet(naam) {

   let wens = “Goedemorgen, ”;

   console.log(wens + naam);

}

wens=?  (A) wens == “Goedemorgen, “  (B) wens bestaat niet meer
```

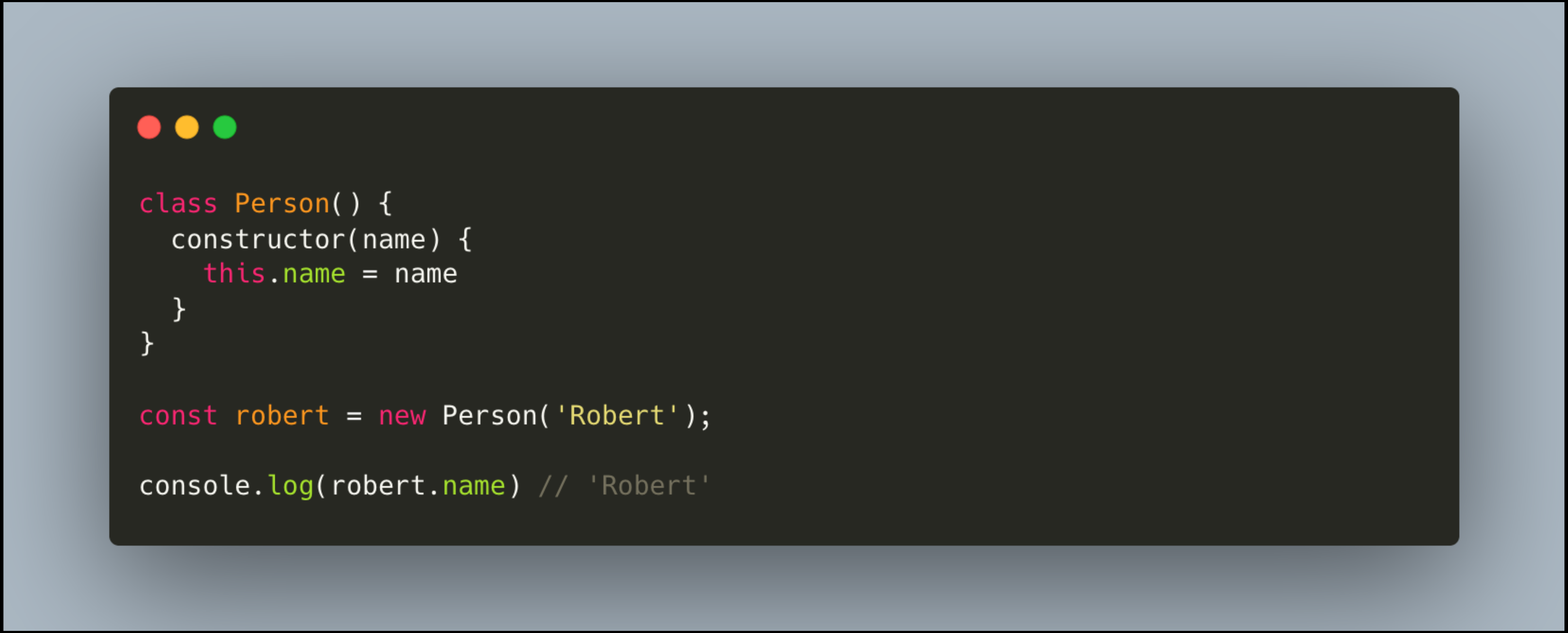
Classes

Classes define Objects. Objects are micro systems with data and functionality.
Classes are usually written by people that create libraries or modules.

Objects are **initialised** by importing or creating them, using
the *new* keyword, an example:

```
let myDobbelsteen = new Dobbelsteen()
```

Classes



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

const robert = new Person('Robert');

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

Modularization

Modules are a way to divide codebase into separate blocks.

JavaScript supports basic modularization of code: functionality can be **exported** and **imported** elsewhere

Developers use it as a means to **modularise their code.** This enables them 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

Libraries

Libraries are collections of related functionality. We can think of Libraries as collections of modularized functions we can access separately.

D3 is a **Library**, as it contains functions that enable us to convert data into SVGs and to show those SVG’s in our HTML pages.

Components

Components also are used for modularization: a way to divide a codebase into separate blocks.

In Front End Development, we use User Interface components to define (preferably) reusable UI elements.

Svelte is component-based. Svelte components combine HTML, CSS and JavaScript

Modules

In Software Engineering, **Modules** are containers, often made of (reusable) components.

Modules are logical parts of big software products as in *module Finance*, *module HR*, *module Manufacturing* that can be added to an ERP system.

Components - with SvelteKit

SvelteKit supports component based development

But it also supports working in one, huge component

Choose your strategy!

Show and Tell

??

Schedule

1. Terminology in development
2. **Formats for data transfer**
3. Continue working on concept + questions

Formats for data transfer

CSV: Comma Separated Values

XML: eXtensible Markup Language

JSON: Java Script Object Notation

CSV

CSV: Comma Separated Values

Oldest format for data transfer

```
"Timestamp","Huisdier","OogKleur","Windrichting","Naam","Dag","Wat was je lievelingsdatum ook
alweer?","Kleuren","Unicorns","Zuivelproduct","Later","Wat wilde je later worden als je groot bent, maar nu
toen je zelf 8 was?","ZinIn","Tosti"

"2023/10/15 3:11:13 PM GMT+3","Schildpad","grijs","West","Laura","2023-10-15","15/10/
23","Rood;Blauw","Ze kunnen nukkig zijn maar daar kan ik wel mee
omgaan”,"Yoghurt","Pensionado","Juf","10","Verticaal"
```

+ lightweight, fast

- suitable for tabular data (not for complex data), conversion recommended for JavaScript

XML

Similar to HTML, but eXtensible

```
    <publicatie>

        <artikel>Benvenuti, Laura, Erik Barendsen, Gerrit C. van der Veer, and Johan Versendaal.
"Understanding Computing in a Hybrid World: On the Undergraduate Curriculum Front-End Development." In
Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 580-585. ACM,
2018.</artikel>

            <link>docs/UnderstandingComputing.pdf</link>

     </publicatie>
```

+ suitable for complex data; supports validation rules (reliable data)

`pattern value="[0-9]{4}[]+[a-zA-Z]{2}"`

- can be slow, not compatible with JavaScript as it is - conversion required

JSON

Java Script Object Notation

```
{  artikel: "Benvenuti L.,G.C.van der Veer: 'Multimedia design kan je leren, kunnen we het ook doceren?', In: F.J. Verbeek, D. Lenior and M. Steen (Eds.) Change! - Proceedings CHI-NL 2009. CHI-NL, Leiden, Netherlands, 39-41",

   link: “../docs/MultimediaDesignDoceren.pdf”,

   hoofdstuk: "chapter 3"

   }
```

+ Lightweight, suitable for complex data, compatible with JavaScript

- Does not support validation rules, [LB] it is an Object Notation (does not describe objects, only for attributes/properties)

So what?

We use JSON

Data, provided in other formats, should be converted

Converters available online!

Schedule

1. Terminology in development
2. Formats for data transfer
3. **Continue working on concept + questions**