Web Design
& Development

Refactor 🛠️

Feb-28

Refactor 🛠️

Planning

1. Handle routes to manage state
2. Modules in JS

Handle routes to manage state

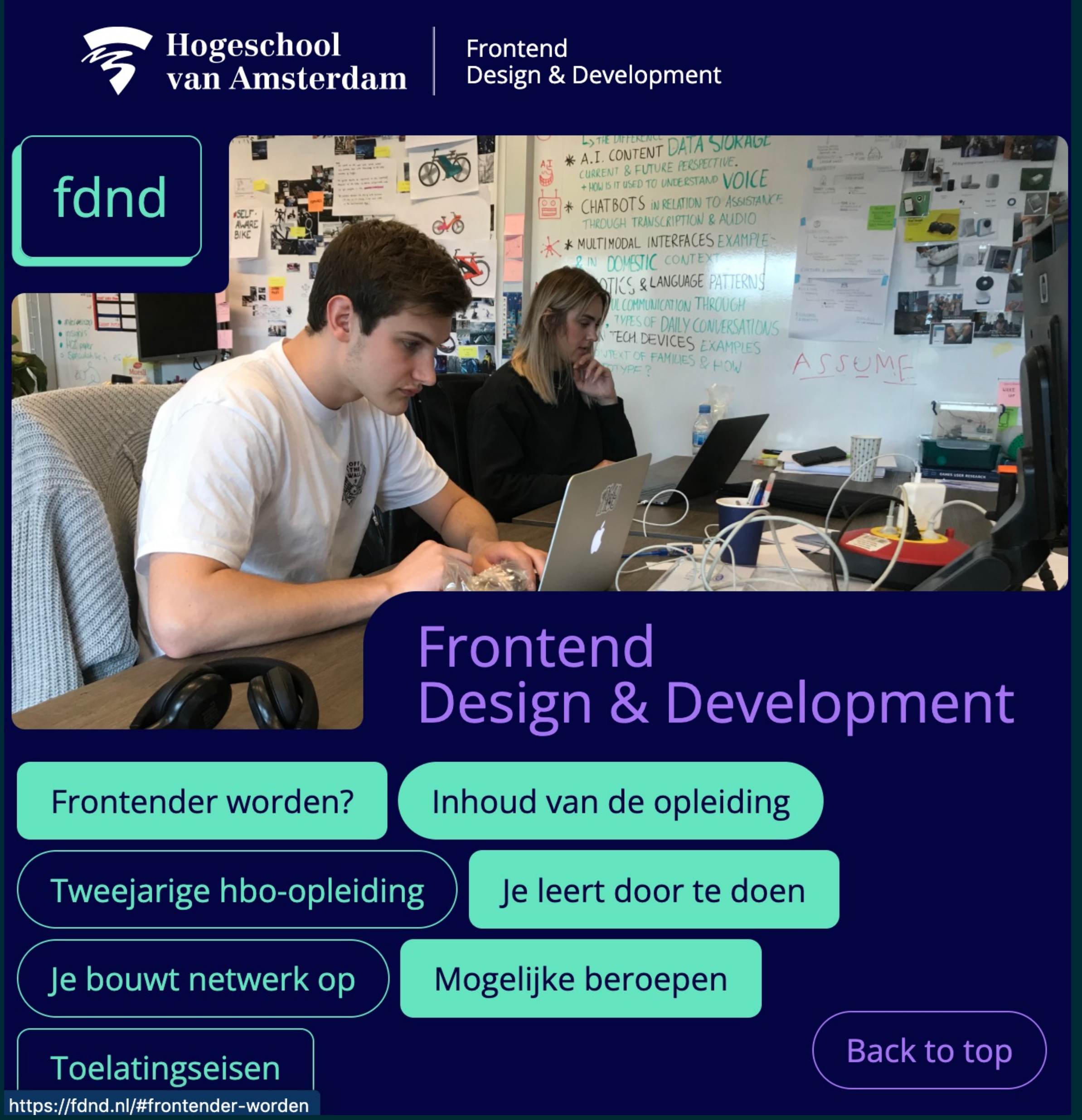
Dit is net server side programmeren.

Sophie van der Burg

Routing

#hash router

The hash-part of an url points to a specific resource in the web page. An element with a corresponding value for the id attribute.

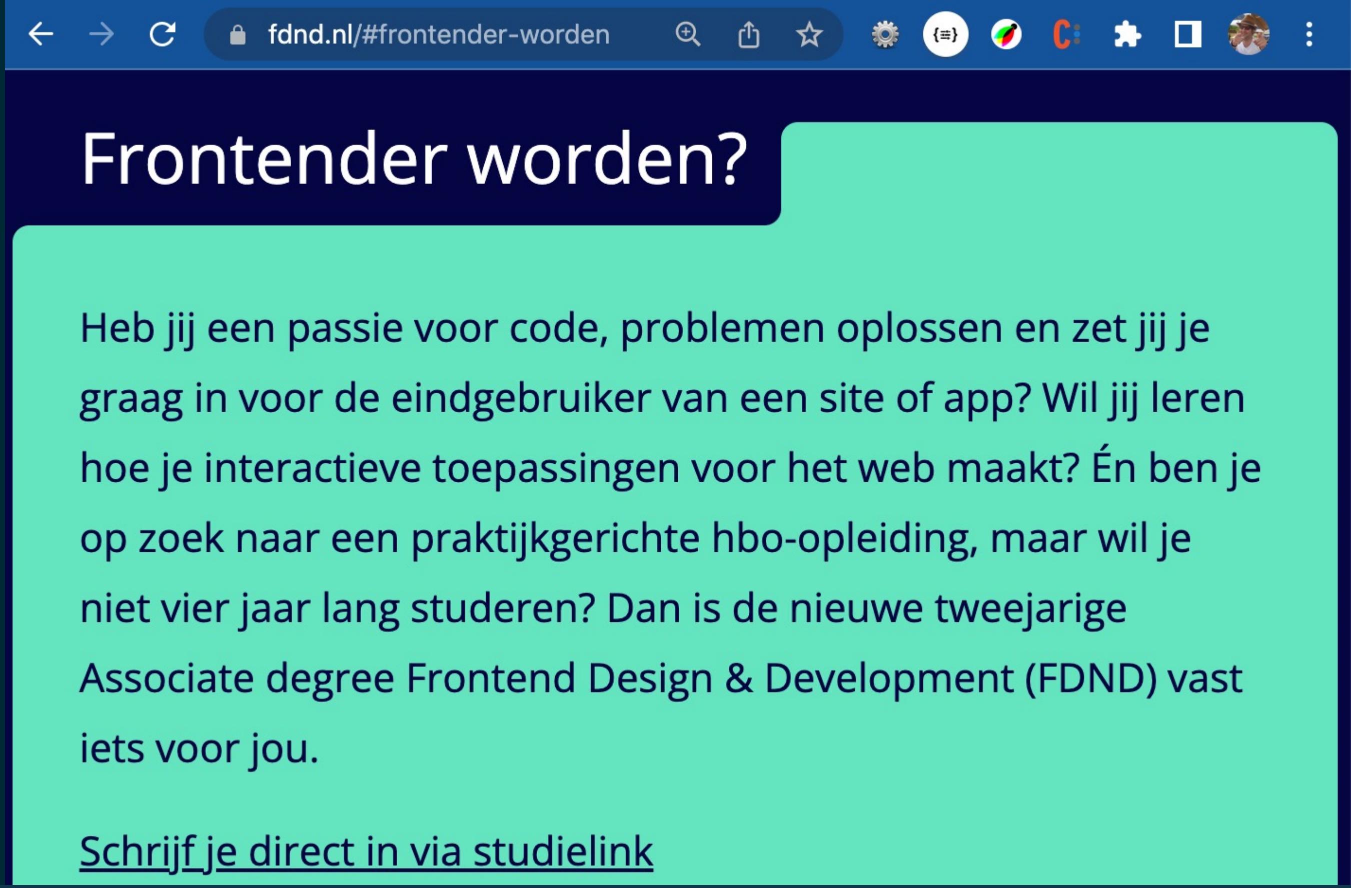


```
<a href="#frontender-worden">Frontender
```

Routing

#hash router

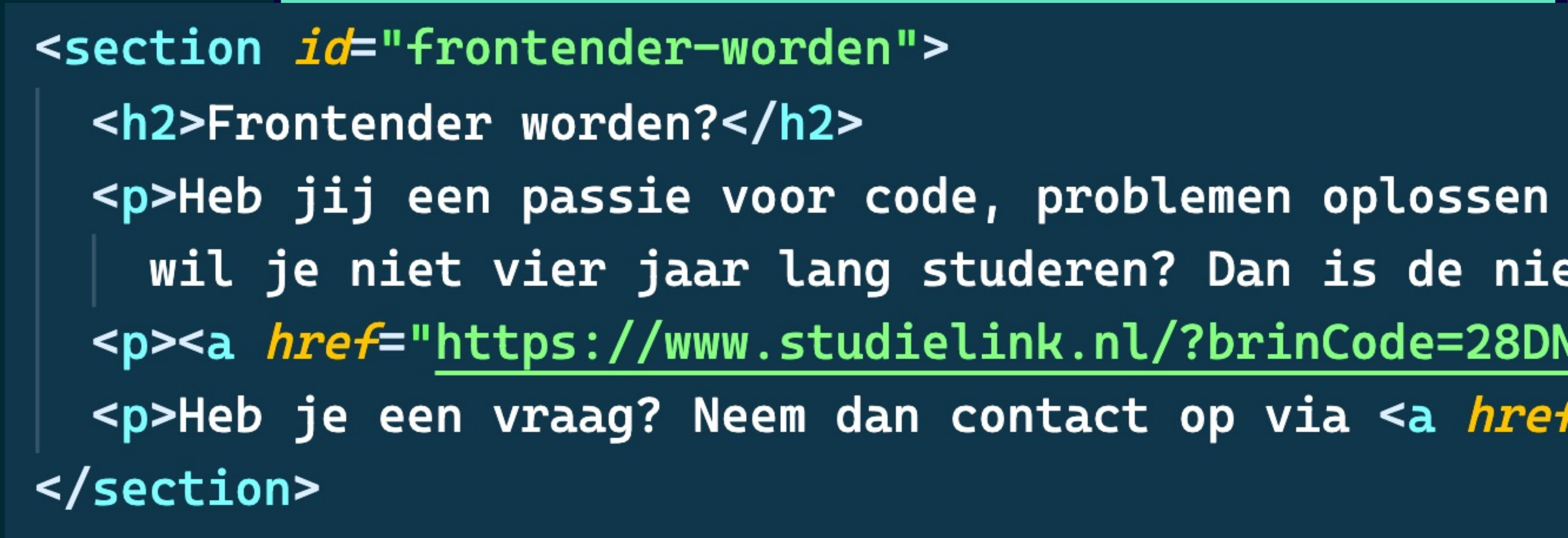
The hash-part of an url points to a specific resource in the web page. An element with a corresponding value for the id attribute.


fdnd.nl/#frontender-worden

Frontender worden?

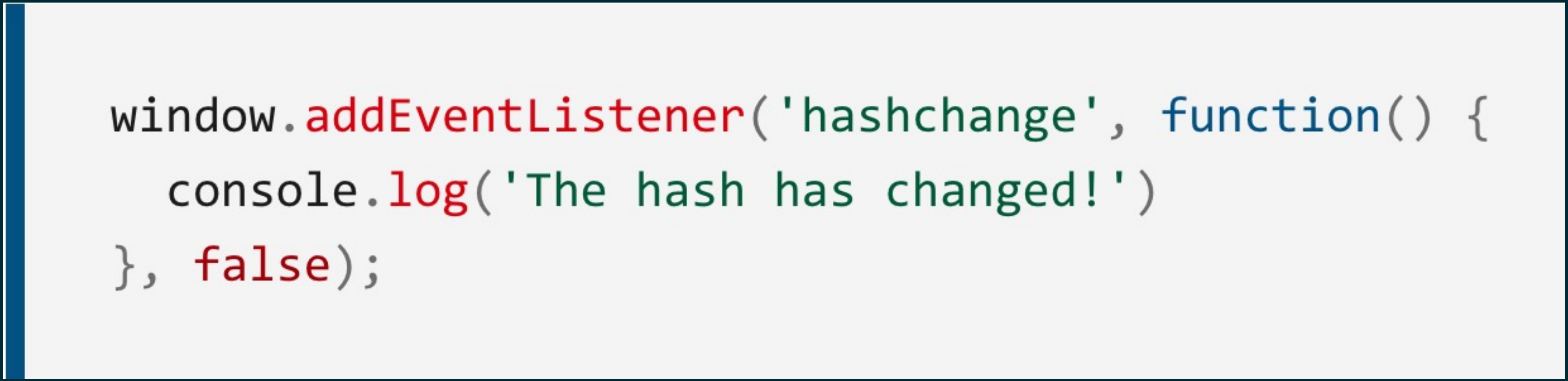
Heb jij een passie voor code, problemen oplossen en zet jij je graag in voor de eindgebruiker van een site of app? Wil jij leren hoe je interactieve toepassingen voor het web maakt? Én ben je op zoek naar een praktijkgerichte hbo-opleiding, maar wil je niet vier jaar lang studeren? Dan is de nieuwe tweejarige Associate degree Frontend Design & Development (FDND) vast iets voor jou.

Schrijf je direct in via studielink


```
<section id="frontender-worden">
  <h2>Frontender worden?</h2>
  <p>Heb jij een passie voor code, problemen oplossen
    wil je niet vier jaar lang studeren? Dan is de nie
  <p><a href="https://www.studielink.nl/?brinCode=28DN
  <p>Heb je een vraag? Neem dan contact op via <a href
</section>
```

#hash router

The hash-part of an url isn’t send to the server. It is interpreted by the browser and accessible through JavaScript...

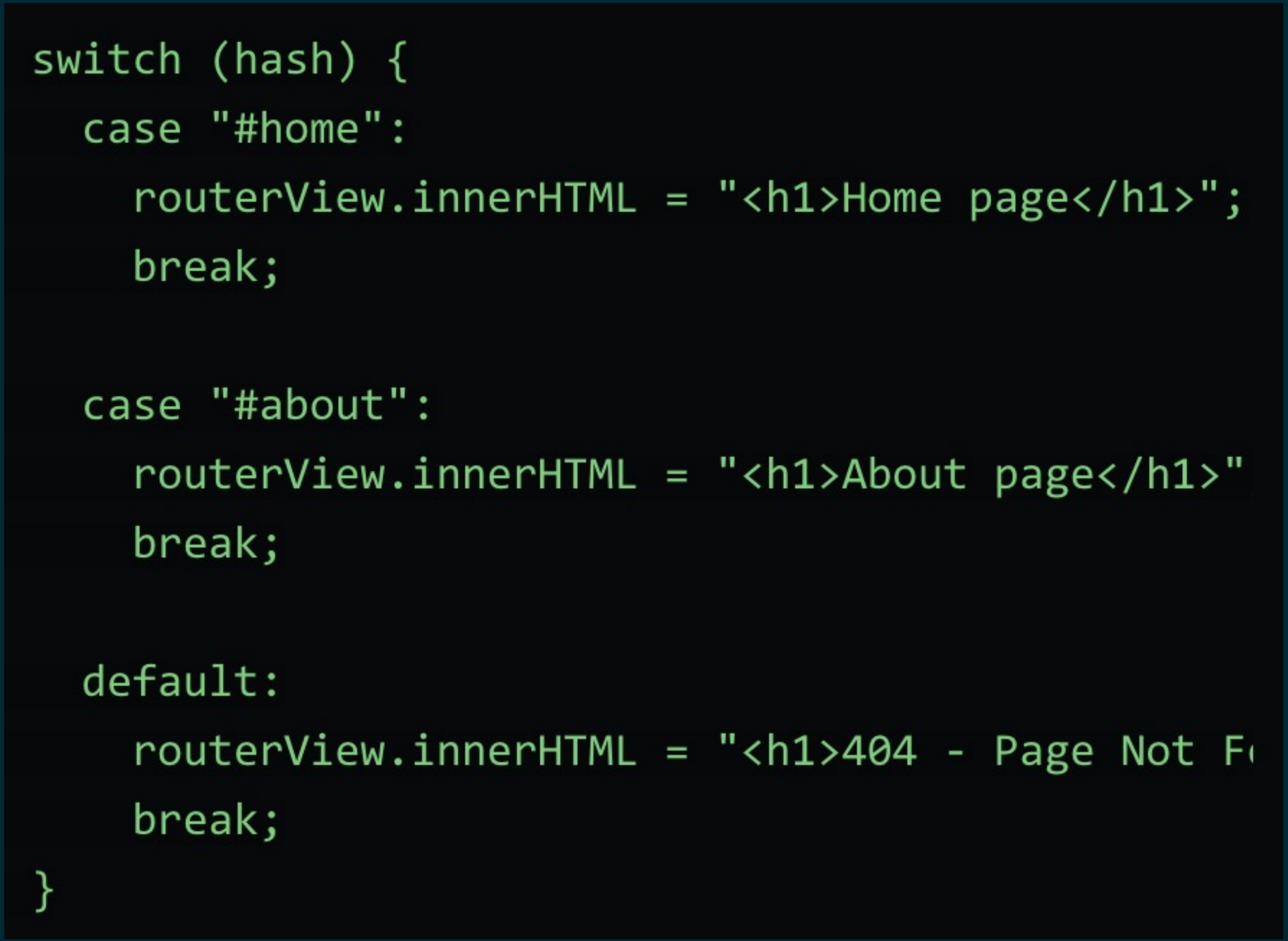


```
window.addEventListener('hashchange', function() {
  console.log('The hash has changed!')
}, false);
```

Routing

#hash router

… and can be used to navigate the different routes (aka states) of your SPA.



```
switch (hash) {
  case "#home":
    routerView.innerHTML = "<h1>Home page</h1>";
    break;

  case "#about":
    routerView.innerHTML = "<h1>About page</h1>"
    break;

  default:
    routerView.innerHTML = "<h1>404 - Page Not F
    break;
}
```

Routing

#hash router

... this can be a bit complex, so you might want to use a micro library

```
routie(
  {
  'gifs': () => {
    loader('active')
    getData().then(data => {
      render(data)
      updateUI('gifs')
    });
  },
  'gifs/:id': id => {
    loader('active')
    getData(id).then(data => {
      render(data, id)
      updateUI('giphy')
    });
  },
  'about': () => {
    updateUI('about')
  }
})
```

History API

The History API exposes useful methods and properties that let you navigate back and forth through the user's history, and manipulate the contents of the history stack (MDN)

Refactor 🛠

Planning

1. ~~Handle routes to manage state~~
2. Modules in JS

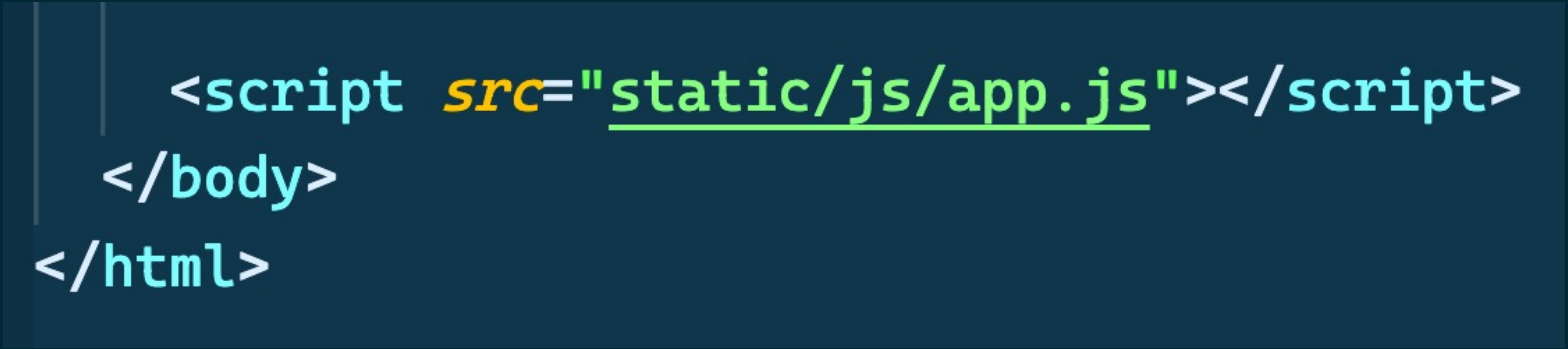
Modules in JS

Het lijkt wel of we zelf een framework maken.

Lotte Koblens

~~ES Modules~~

HTML



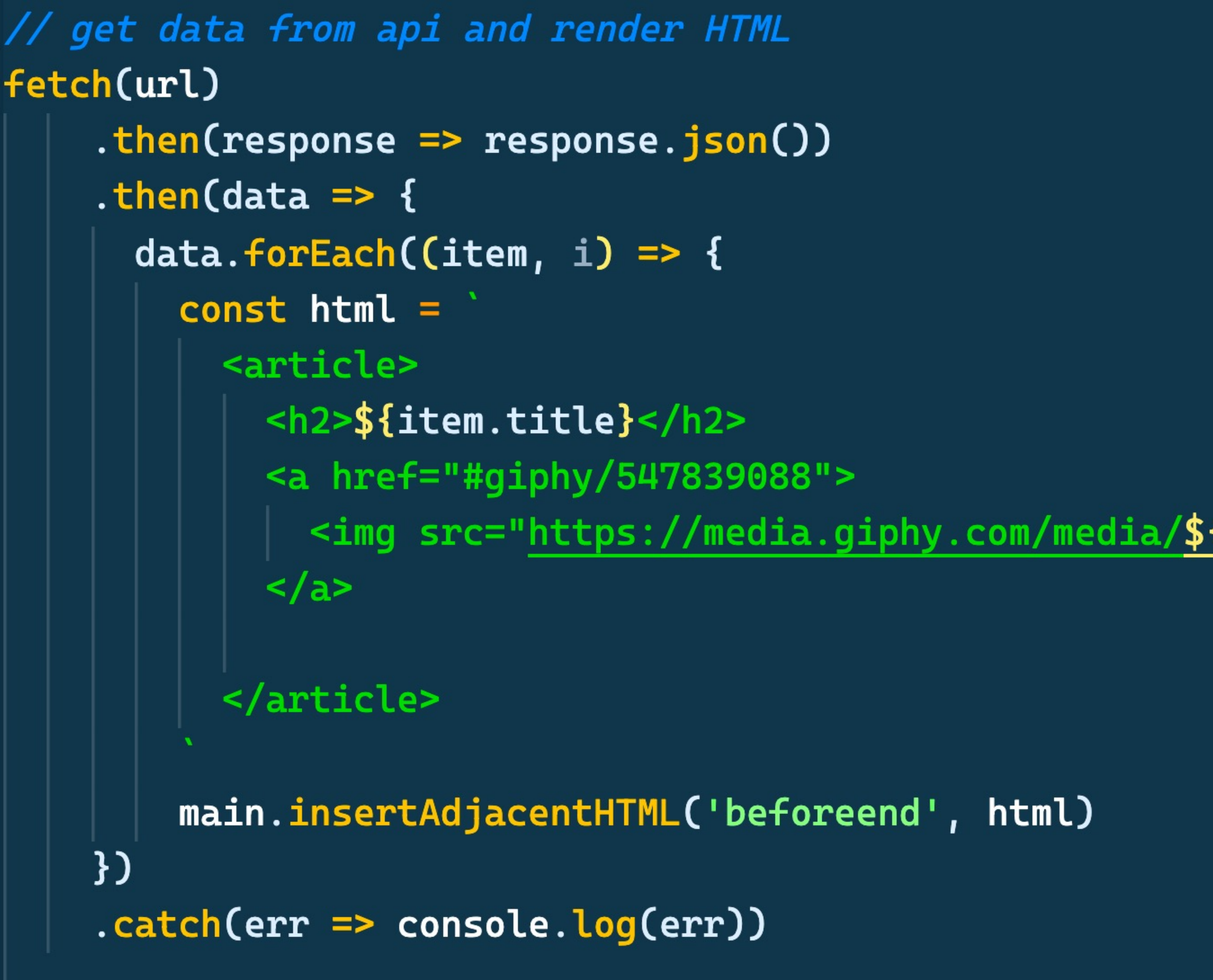
```html
    <script src="static/js/app.js"></script>
  </body>
</html>
```

~~ES Modules~~

JavaScript

No structure,
code in
one function

All code in
one file



```
// get data from api and render HTML
fetch(url)
    .then(response => response.json())
    .then(data => {
      data.forEach((item, i) => {
        const html = `
          <article>
            <h2>${item.title}</h2>
            <a href="#giphy/547839088">
              <img src="https://media.giphy.com/media/$
            </a>

          </article>
        `

        main.insertAdjacentHTML('beforeend', html)
    })
    .catch(err => console.log(err))
```

ES Modules

Split code into separate files and import them when needed

only main script in HTML
export / import bindings
execute in strict mode
defer by default
scoped by default

~~ES Modules~~

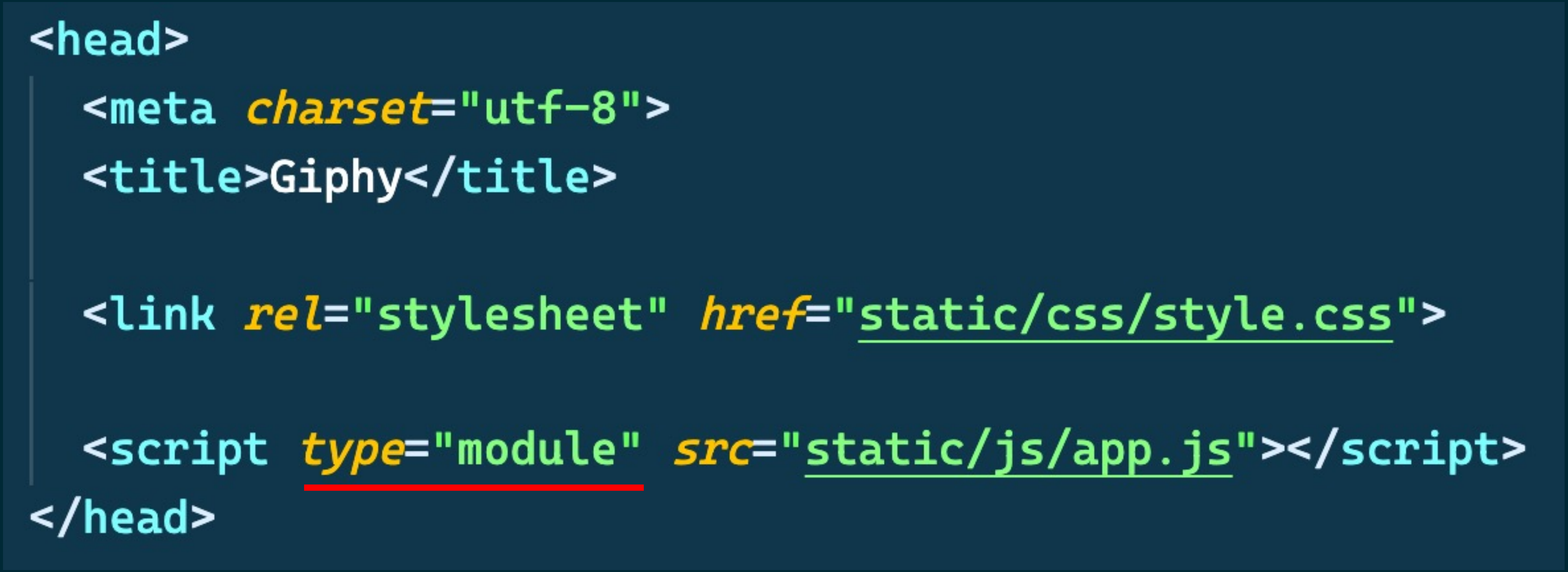
JavaScript

Some structure, each function does one thing and has a logical name

all code in one file

```
// get data from api
function getData(url) {
  return fetch(url)
    .then(response => response.json())
    .then(data => {
      render(data.data)
    })
    .catch(err => console.log(err))
}

// render HTML with retrieved data
function renderHTML(data) {
  data.forEach((item, i) => {
    const html = `
      <article>
        <h2>${item.title}</h2>
        <a href="#giphy/547839088">
          <img src="https://media.giphy.com/media/${item.id}/giphy.gif">
        </a>

      </article>
    `

    main.insertAdjacentHTML('beforeend', html)
  })
}
```

HTML

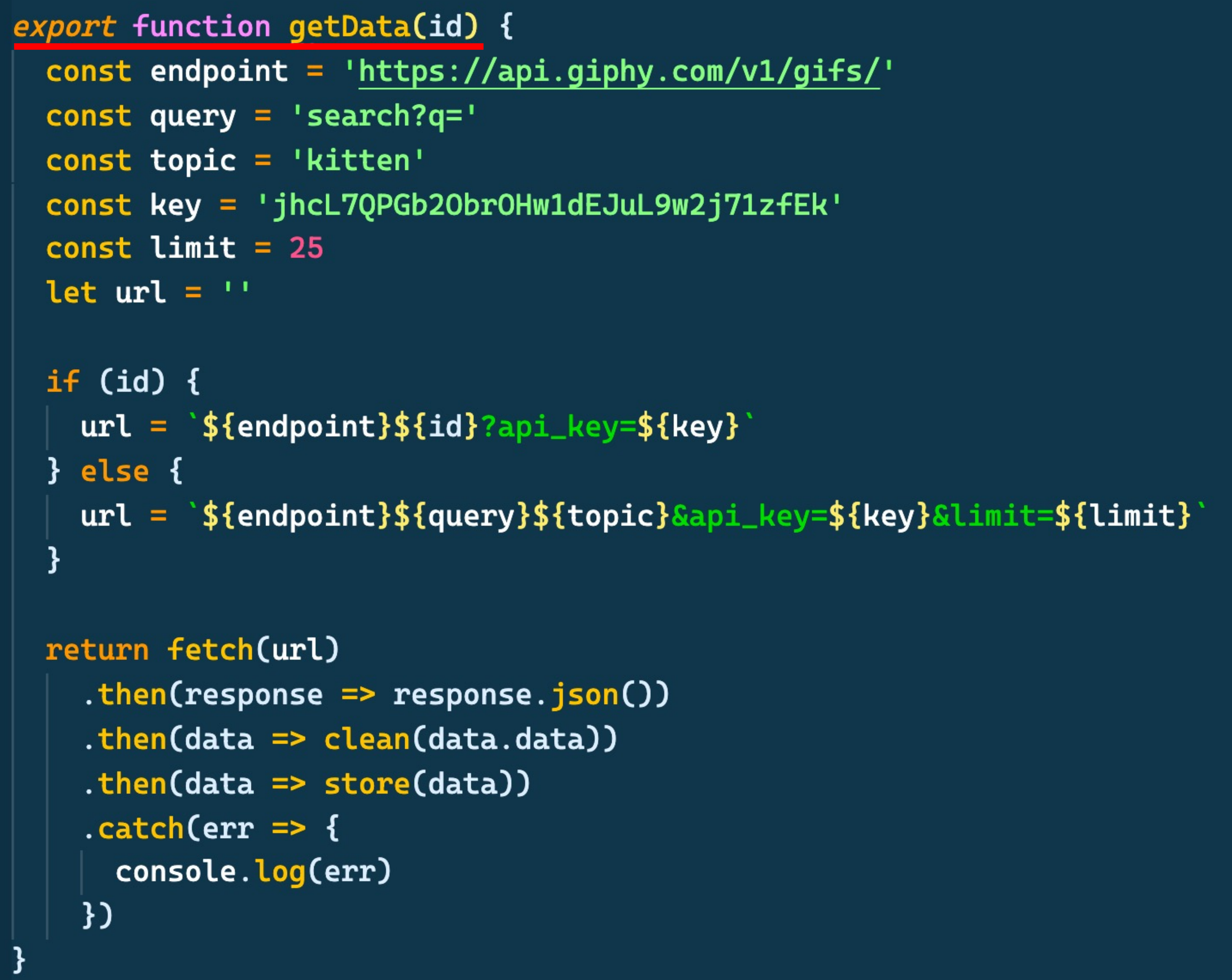


```html
<head>
  <meta charset="utf-8">
  <title>Giphy</title>

  <link rel="stylesheet" href="static/css/style.css">

  <script type="module" src="static/js/app.js"></script>
</head>
```

ES Modules

JavaScript

Code is placed in separate files, and stuff can be exported...

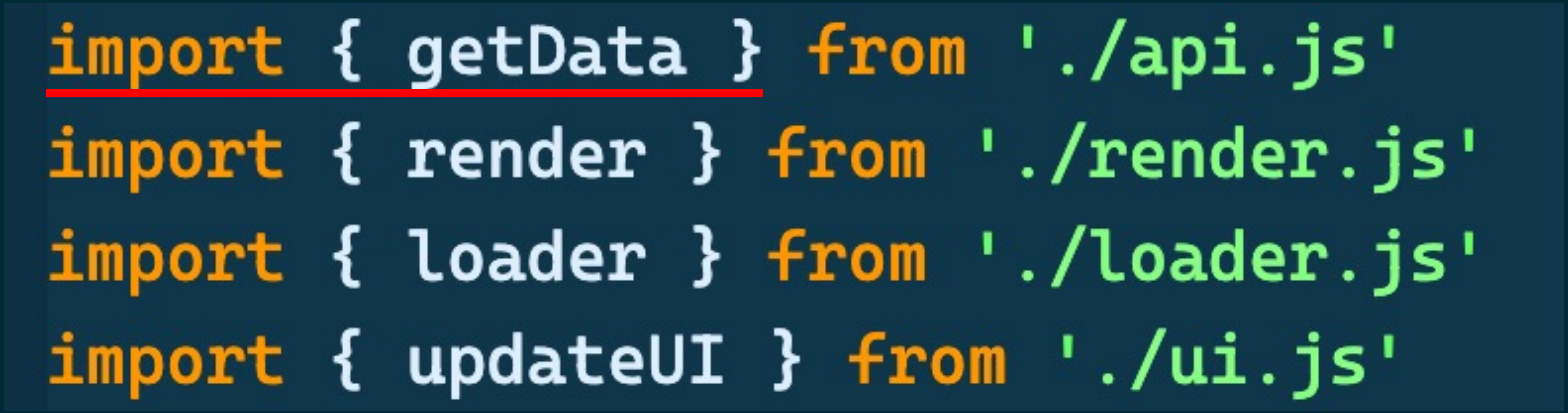


```
export function getData(id) {
  const endpoint = 'https://api.giphy.com/v1/gifs/'
  const query = 'search?q='
  const topic = 'kitten'
  const key = 'jhcL7QPGb2ObrOHw1dEJuL9w2j71zfEk'
  const limit = 25
  let url = ''

  if (id) {
    url = `${endpoint}${id}?api_key=${key}`
  } else {
    url = `${endpoint}${query}${topic}&api_key=${key}&limit=${limit}`
  }

  return fetch(url)
    .then(response => response.json())
    .then(data => clean(data.data))
    .then(data => store(data))
    .catch(err => {
      console.log(err)
    })
}
```

ES Modules

JavaScript

... and imported
where needed



```
import { getData } from './api.js'
import { render } from './render.js'
import { loader } from './loader.js'
import { updateUI } from './ui.js'
```

Refactor 🛠️

Planning

1. ~~Handle routes to manage state~~
2. ~~Modules in JS~~

Klik klik klik