Traditional Web Apps

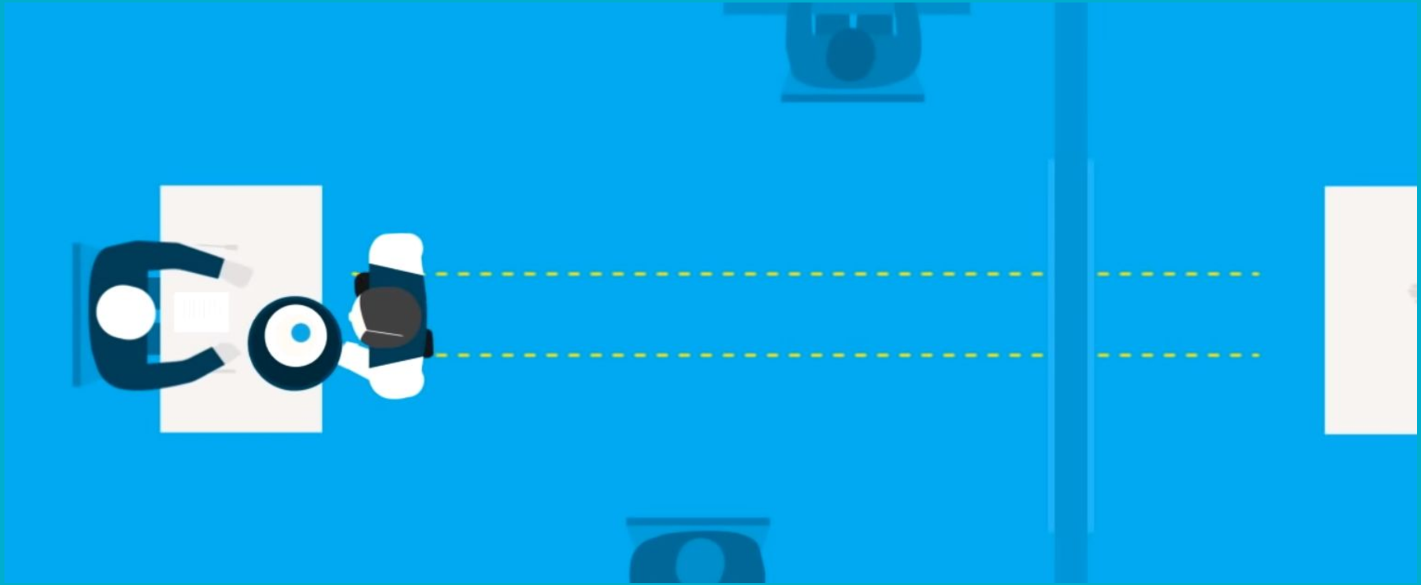
(API, HTTP, REST, Ajax, SPA, Promise, Fetch)

API

- Tehnički, API je skraćenica za Application Programming Interface.
- WWW - mreža povezanih servera.
- Svaka web aplikacija se nalazi na nekom serveru.
- Pristup web-u (Browser, Mobile App) - CLIENT.
- Pristup serveru - deo servera koji prima zahteve(request) i šalje odgovore (response) naziva se API.

API

- Mnogi internet servisi daju pristup svojim API-ima.
- Format zahteva i odgovora (HTML, XML, JSON)



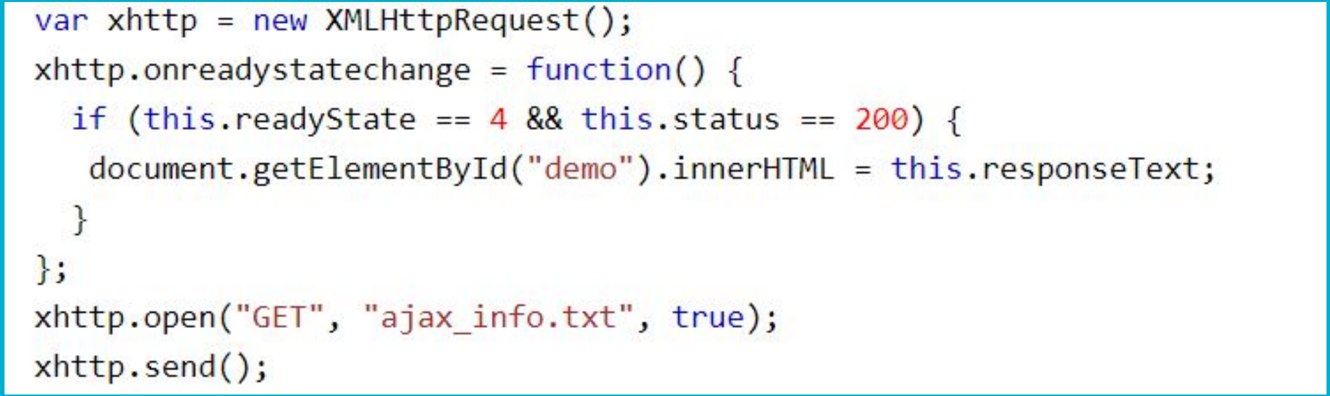
HTTP(Hypertext Transfer Protocol)

- Skup pravila za prenos datoteka kao što su text, grafičke slike, zvuk , video i druge multimedijske datoteke na Internetu.
- Client upućuje HTTP zahtev serveru i taj server odgovara resursom.
- Svaki zahtev mora imati URL adresu i metodu.
- Glavne metode (GET, POST, PUT, DELETE)
- HTTP Headers
- HTTP status : (1xx, 2xx, 3xx, 4xx, 5xx)

REST

- REST je arhitektonski stil ili obrazac dizajna za API.
- Aplikacija izlaže informacije o sebi u obliku podataka o svojim resursima.
- Omogućava klijentu da preduzme akcije na tim resursima.

AJAX(Asynchronous JavaScript And XML)



```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

SPA(Single page app)

Past:

- Browser - manje sposobnosti
- Javascript - slabe perfomanse
- Svaka stranica je dolazila sa servera

Today:

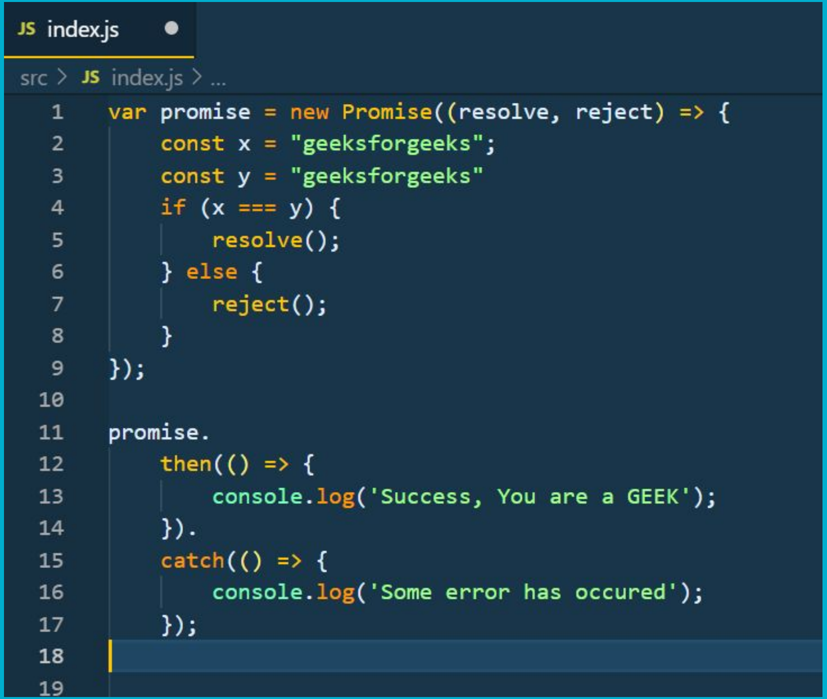
- Aplikacija od jedne stranice
- Kod se učitava samo jednom (HTML, CSS, JavaScript)
- JSON razmena podataka sa serverom pomocu HTTP-a

SPA(Single page app)

- Angular
- ReactJS
- VueJS

Promise

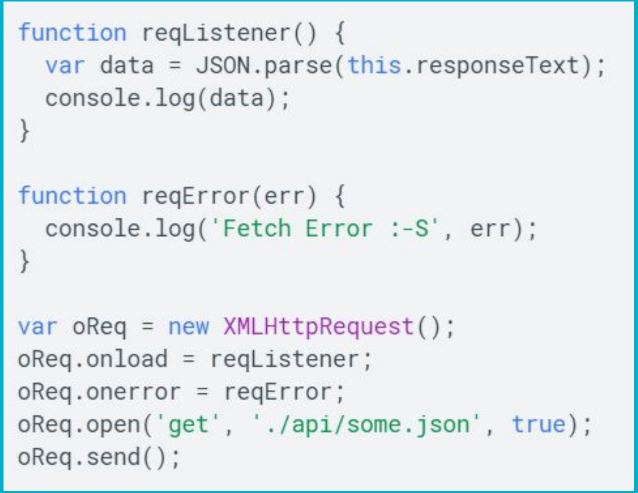
- Rukovanje asinhronim operacijama u JS-u.
- Lako se nose sa više asinhronih operacija
- Bolje rukovanje sa greškama
- Poboljšava čitljivost koda



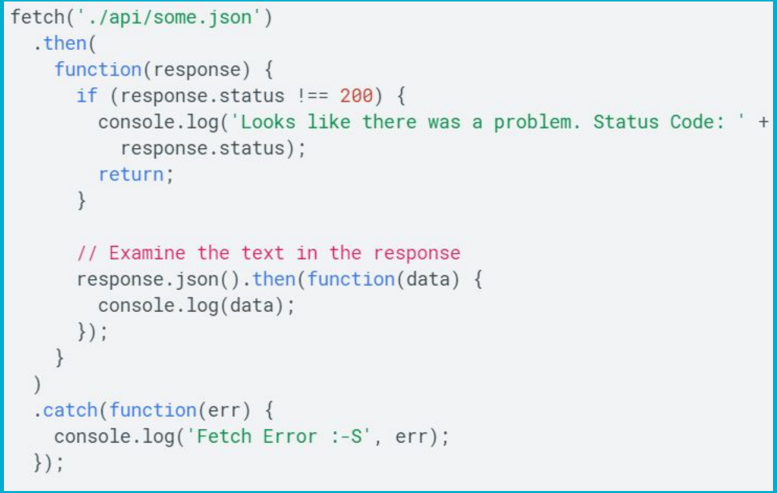
```javascript
var promise = new Promise((resolve, reject) => {
    const x = "geeksforgeeks";
    const y = "geeksforgeeks"
    if (x === y) {
        resolve();
    } else {
        reject();
    }
});

promise.
    then(() => {
        console.log('Success, You are a GEEK');
    }).
    catch(() => {
        console.log('Some error has occured');
    });
```

Fetch

- Omogućava da napravite HTTP zahteve slične XMLHttpRequest.
- Glavna razlika je sto Fetch koristi Promise-e.



```
function reqListener() {
  var data = JSON.parse(this.responseText);
  console.log(data);
}

function reqError(err) {
  console.log('Fetch Error :-S', err);
}

var oReq = new XMLHttpRequest();
oReq.onload = reqListener;
oReq.onerror = reqError;
oReq.open('get', './api/some.json', true);
oReq.send();
```

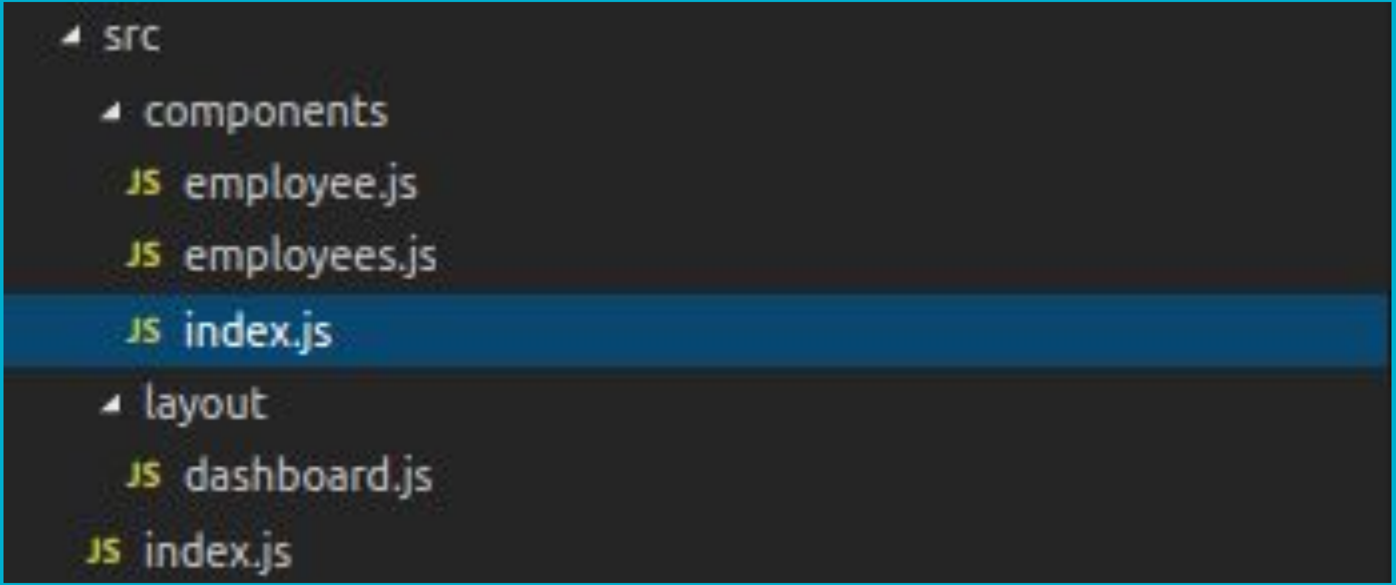


```
fetch('./api/some.json')
  .then(
    function(response) {
      if (response.status !== 200) {
        console.log('Looks like there was a problem. Status Code: ' +
          response.status);
        return;
      }

      // Examine the text in the response
      response.json().then(function(data) {
        console.log(data);
      });
    }
  )
  .catch(function(err) {
    console.log('Fetch Error :-S', err);
  });
```

Fetch

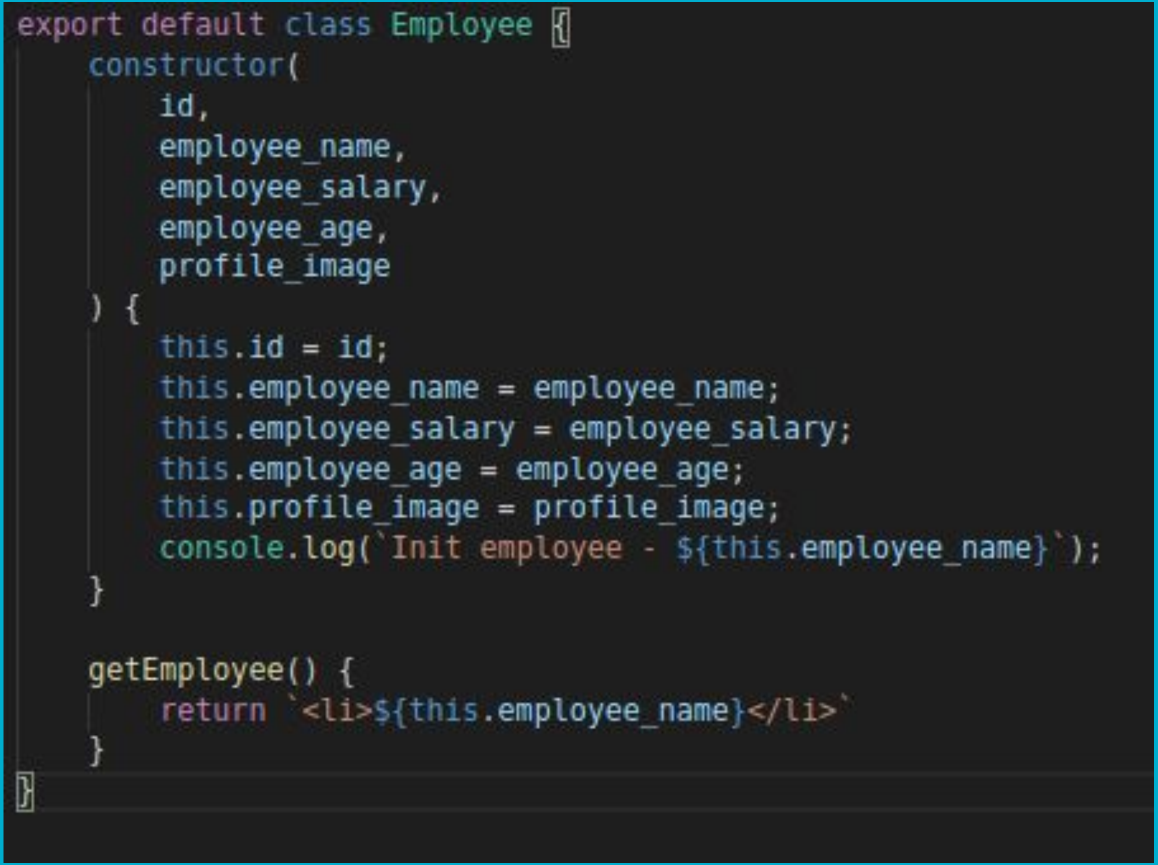
```
fetch(url, {
    method: 'post',
    headers: {
      "Content-type": "application/x-www-form-urlencoded; charset=UTF-8"
    },
    body: 'foo=bar&lorem=ipsum'
  })
  .then(json)
  .then(function (data) {
    console.log('Request succeeded with JSON response', data);
  })
  .catch(function (error) {
    console.log('Request failed', error);
  });
```

Dashboard



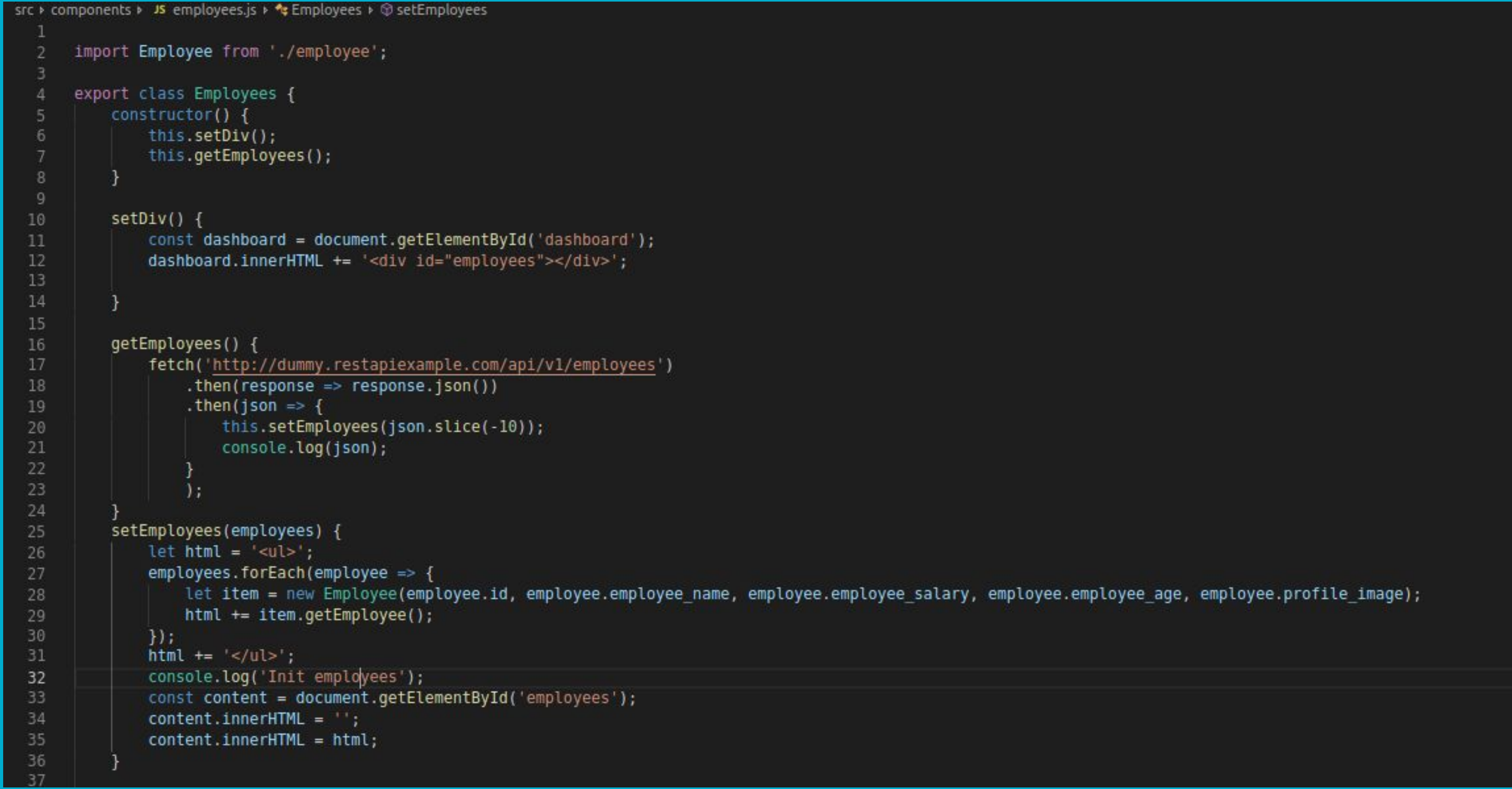
```
◢ src
  ◢ components
    JS employee.js
    JS employees.js
    JS index.js
  ◢ layout
    JS dashboard.js
  JS index.js
```

Dashboard



```
export default class Employee {
    constructor(
        id,
        employee_name,
        employee_salary,
        employee_age,
        profile_image
    ) {
        this.id = id;
        this.employee_name = employee_name;
        this.employee_salary = employee_salary;
        this.employee_age = employee_age;
        this.profile_image = profile_image;
        console.log(`Init employee - ${this.employee_name}`);
    }

    getEmployee() {
        return `<li>${this.employee_name}</li>`
    }
}
```

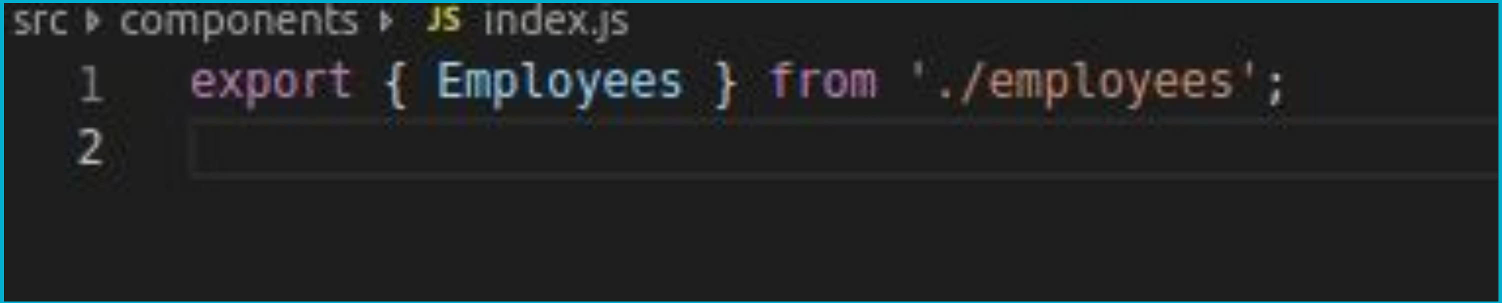
Dashboard



src › components › JS employees.js › Employees › setEmployees

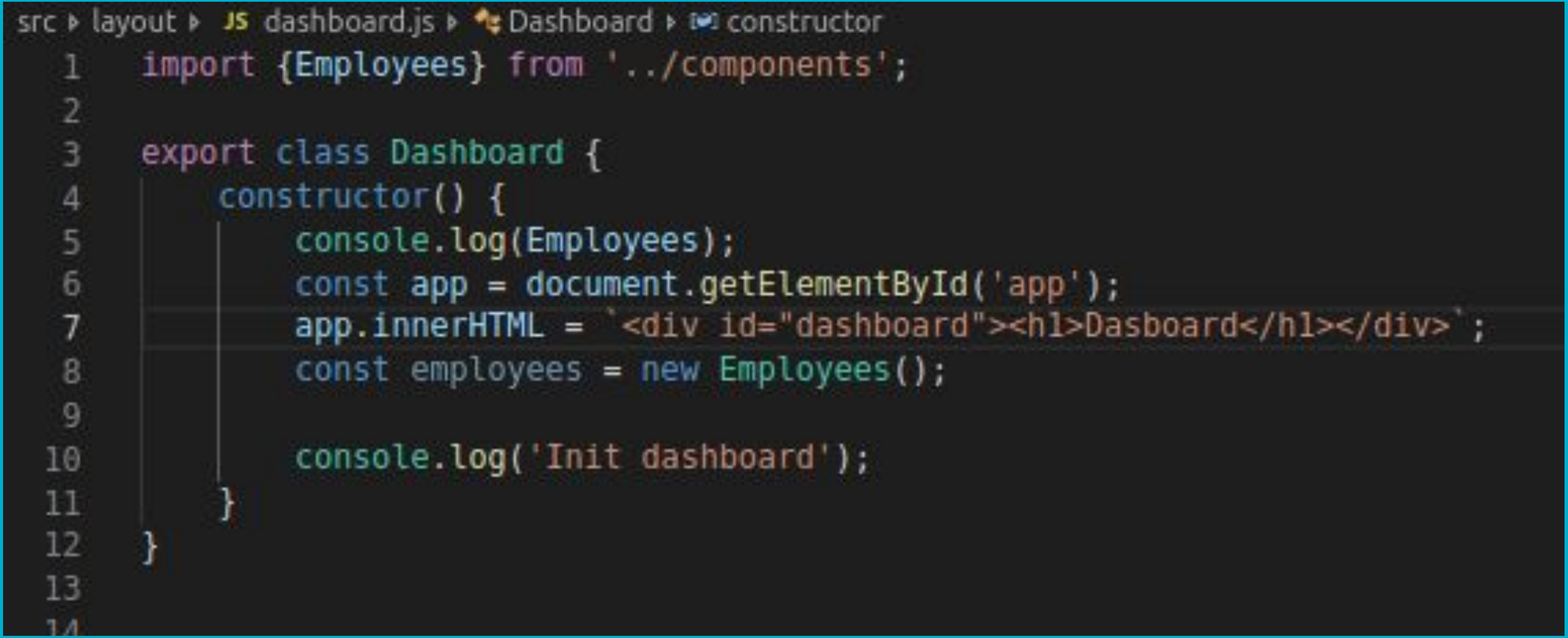
```javascript

import Employee from './employee';

export class Employees {
    constructor() {
        this.setDiv();
        this.getEmployees();
    }

    setDiv() {
        const dashboard = document.getElementById('dashboard');
        dashboard.innerHTML += '<div id="employees"></div>';

    }

    getEmployees() {
        fetch('http://dummy.restapiexample.com/api/v1/employees')
            .then(response => response.json())
            .then(json => {
                this.setEmployees(json.slice(-10));
                console.log(json);
            }
            );
    }
    setEmployees(employees) {
        let html = '<ul>';
        employees.forEach(employee => {
            let item = new Employee(employee.id, employee.employee_name, employee.employee_salary, employee.employee_age, employee.profile_image);
            html += item.getEmployee();
        });
        html += '</ul>';
        console.log('Init employees');
        const content = document.getElementById('employees');
        content.innerHTML = '';
        content.innerHTML = html;
    }

```

Dashboard



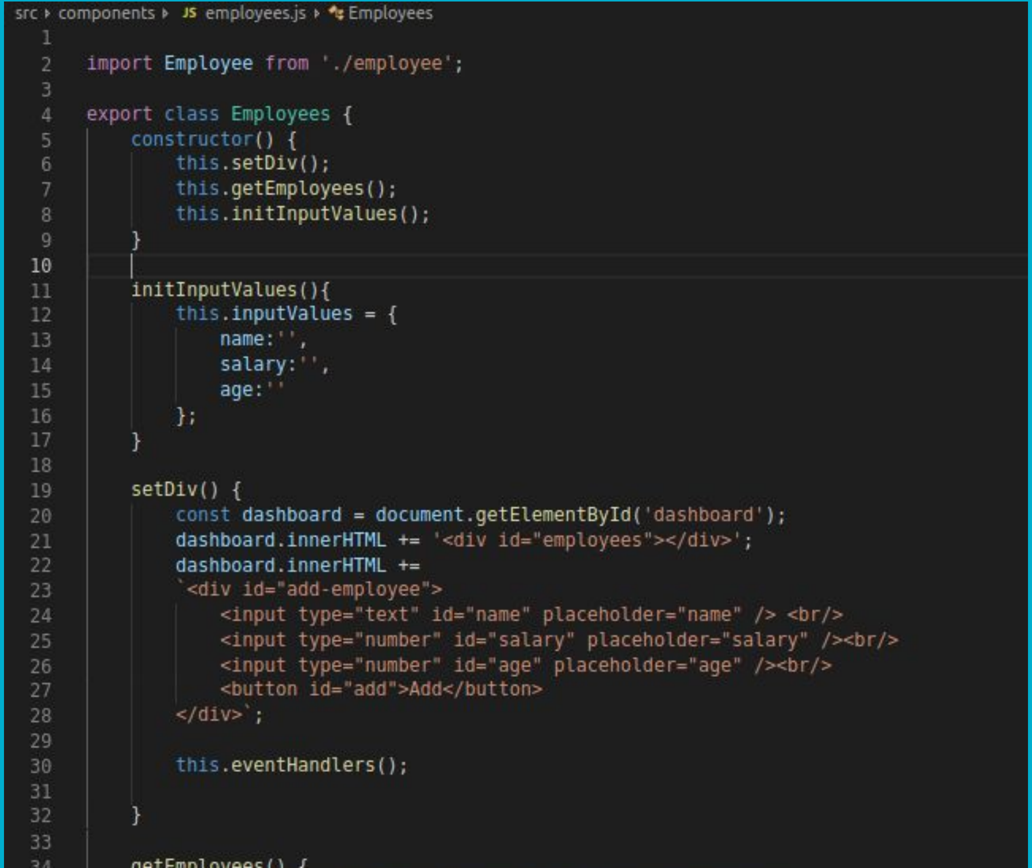
```
export { Employees } from './employees';

```

Dashboard



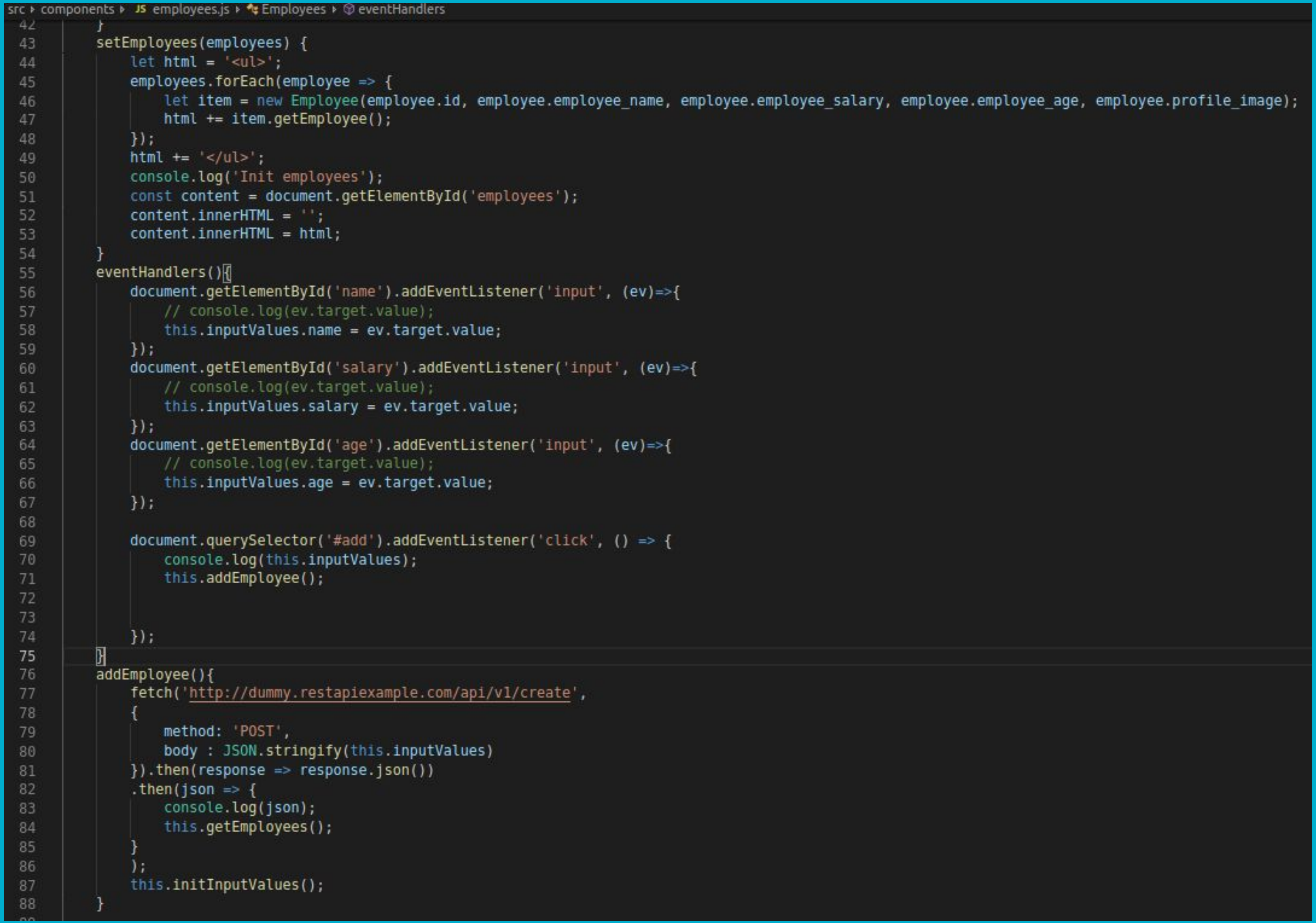
```
import {Employees} from '../components';

export class Dashboard {
    constructor() {
        console.log(Employees);
        const app = document.getElementById('app');
        app.innerHTML = `<div id="dashboard"><h1>Dasboard</h1></div>`;
        const employees = new Employees();

        console.log('Init dashboard');
    }
}
```

Dashboard



```

import Employee from './employee';

export class Employees {
    constructor() {
        this.setDiv();
        this.getEmployees();
        this.initInputValues();
    }

    initInputValues(){
        this.inputValues = {
            name:'',
            salary:'',
            age:''
        };
    }

    setDiv() {
        const dashboard = document.getElementById('dashboard');
        dashboard.innerHTML += '<div id="employees"></div>';
        dashboard.innerHTML +=
        `<div id="add-employee">
            <input type="text" id="name" placeholder="name" /> <br/>
            <input type="number" id="salary" placeholder="salary" /><br/>
            <input type="number" id="age" placeholder="age" /><br/>
            <button id="add">Add</button>
        </div>`;

        this.eventHandlers();

    }

    getEmployees() {
```

Dashboard


src › components › JS employees.js › Employees › eventHandlers

```
    }
    setEmployees(employees) {
        let html = '<ul>';
        employees.forEach(employee => {
            let item = new Employee(employee.id, employee.employee_name, employee.employee_salary, employee.employee_age, employee.profile_image);
            html += item.getEmployee();
        });
        html += '</ul>';
        console.log('Init employees');
        const content = document.getElementById('employees');
        content.innerHTML = '';
        content.innerHTML = html;
    }
    eventHandlers(){
        document.getElementById('name').addEventListener('input', (ev)=>{
            // console.log(ev.target.value);
            this.inputValues.name = ev.target.value;
        });
        document.getElementById('salary').addEventListener('input', (ev)=>{
            // console.log(ev.target.value);
            this.inputValues.salary = ev.target.value;
        });
        document.getElementById('age').addEventListener('input', (ev)=>{
            // console.log(ev.target.value);
            this.inputValues.age = ev.target.value;
        });

        document.querySelector('#add').addEventListener('click', () => {
            console.log(this.inputValues);
            this.addEmployee();


        });
    }
    addEmployee(){
        fetch('http://dummy.restapiexample.com/api/v1/create',
        {
            method: 'POST',
            body : JSON.stringify(this.inputValues)
        }).then(response => response.json())
        .then(json => {
            console.log(json);
            this.getEmployees();
        }
        );
        this.initInputValues();
    }
```