JS Modules

ES6 Modules

Evo na šta ćemo da obratimo pažnju:

1. Zašto su potrebni ES6 module-i
2. Povratak u dane kada su se skripte ručno učitavale
3. Kako rade ES6 module-i(**import** vs **export**)
4. Vežba

Zašto su potrebni ES6 module-i?

- Scenario 1 - Nemojte ponovo izumeti točak
- Scenario 2 - Prenos znanja
- Scenario 3 - Neočekivano ponašanje

Povratak u dane kada su se skripte ručno učitavale

```
1  <!DOCTYPE html>
2  <head>
3  </head>
4  <body>
5
6    <!--HTML content goes here-->
7
8    <script src="js/script1.js"></script>
9    <script src="js/script2.js"></script>
10   <script src="js/script3.js"></script>
11   <script src="js/script4.js"></script>
12 </body>
13 </html>
```

Ne želimo da file index.html za ove skripte preuzme odgovornost za čitanje - tražimo da imamo određenu strukturu i logiku raščlanjivanja.

Povratak u dane kada su se skripte ručno učitavale

Rešenje:

- Upotreba ES6 module-a, elegantan i održiv pristup koji nam omogućava da razdvojimo stvari i da stvari budu dostupne samo kada su nam potrebne.
- Ključna reč export se koristi kada želimo da negde učinimo nešto dostupnim, a import se koristi za pristup onome što je export učinio dostupnim.

Za šta zapravo možemo da koristimo export ?

- Promenljivu
- Objekat
- Klasu
- Funkciju

Povratak u dane kada su se skripte ručno učitavale

app.js

C: > Users > RS-marjank-01 > Desktop > app.js

```
1  import { script1 } from './js/script1.js';
2  import { script2 } from './js/script2.js';
3  import { script3 } from './js/script3.js';
4  import { script4 } from './js/script4.js';
5
```

index.html

C: > Users > RS-marjank-01 > Desktop > es6modules > index.html > body

```
1  <!DOCTYPE html>
2  <head>
3  </head>
4  <body>
5    <!--HTML content goes here-->
6    <script type="module" src="js/app.js"></script>
7  </body>
8  </html>
```

Kao što možete videti, index.html sada je odgovoran za jednu skriptu, što olakšava održavanje i skaliranje.

Kako rade ES6 module-i

Šta je modul?

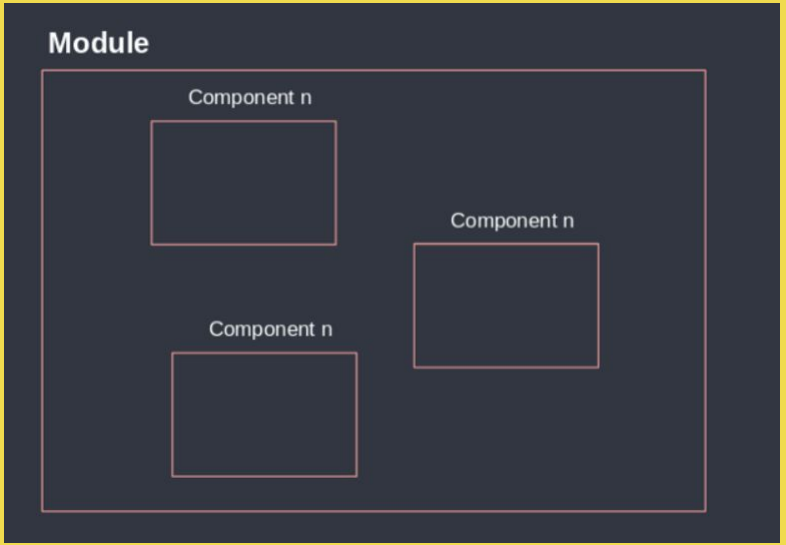
Modul je zbirka malih nezavisnih jedinica (komponenti) koje možemo ponovo koristiti u našoj aplikaciji.

Koja je svrha?

- Lako je raditi
- Lako za održavanje
- Lako za skaliranje

Pa šta je komponenta zaista?

Komponenta može biti promenljiva, funkcija, klasa itd. Drugim rečima, sve što se može izvesti je komponenta (ili je možete nazvati blokom, jedinicom itd.).

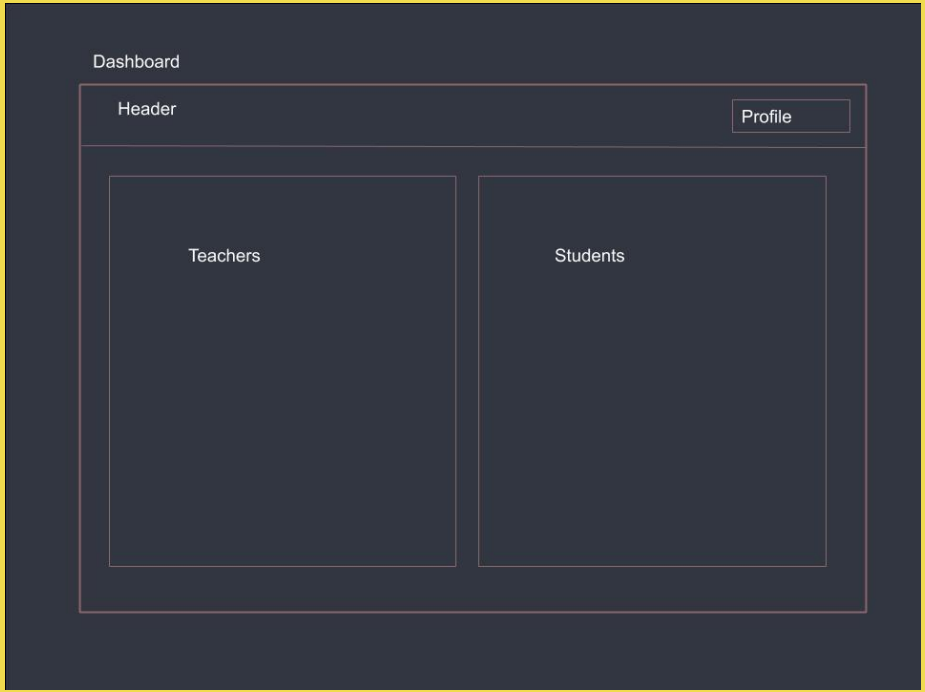


Pa šta je tačno modul?

Kao što je pomenuto, modul je kolekcija komponenti. Ako imamo više komponenti koje komuniciraju ili ih jednostavno moramo prikazati zajedno kako bi formirali integrisanu celinu, onda vam je najverovatnije potreban modul.

Izgradimo Dashboard pomocu module-a

Dizajnirate šta vam treba

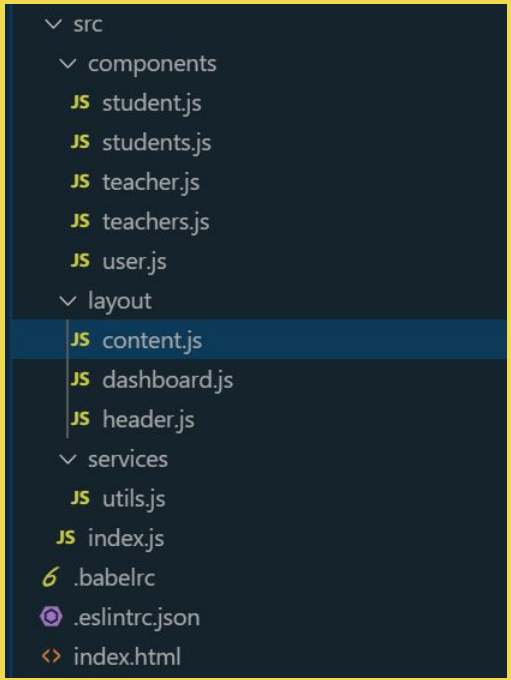


- Komponente (student.js, students.js, teacher.js, teachers.js, user.js)
- Service (utils.js)

- Izgled(dashboard.js, header.js, content.js)

Izgradimo Dashboard pomocu module-a

Podešavanje strukture



student.js

```
export default class Student{
    constructor(
        firstName,
        lastName,
        age,
        gender,
        yearsOfStudy
    ){
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
        this.gender = gender;
        this.yearsOfStudy = yearsOfStudy;
        console.log(`Init student - ${this.firstName}`);
    }
}
```

students.js

```
import Student from './student';
import { countItems } from '../services/utils';

const students = [
    {
        firstName: 'Prvi',
        lastName:'Student',
        age:20,
        gender:'Muski',
        yearsOfStudy:1
    },
    {
        firstName: 'Drugi',
        lastName:'Student',
        age:21,
        gender:'Zenski',
        yearsOfStudy:2
    }
];

export default class Students{
    constructor(){
        students.forEach(student => {
            new Student(student.firstName, student.lastName, student.age,student.gender, student.yearsOfStudy);
        });
        console.log('Init students', countItems(students));
    }
}
```

utils.js

```
1  const countItems = (items) => {
2      console.log('Number of items', items.length);
3      return items.length;
4  }
5
6  export {
7      countItems
8  }
```

content.js

```
import Students from '../components/students'

class Content{
    constructor(){
        const students = new Students();
        console.log('Init content');
    }
}

export default Content;
```

dashboard.js

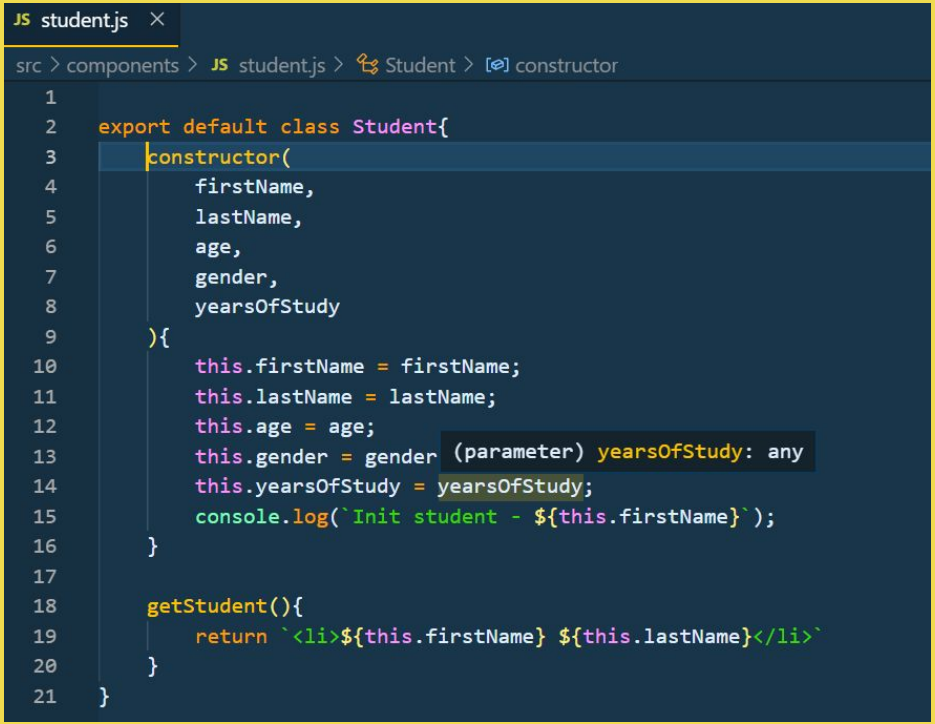
```
import Header from './header.js'
import Content from './content'

export class Dashboard{
    constructor(){
        const header = new Header();
        const content = new Content();
    }
}
```

index.js

```
import '../assets/css/style.css';
import { Dashboard } from './layout/dashboard';
const app = document.getElementById('app');
const dashboard = new Dashboard();
```

Ubaciti HTML u komponente



Ubaciti HTML u komponente

students.js

src > components > students.js > studentLists

```
import Student from './student';
import { countItems } from '../services/utils';

const studentLists = [
    {
        firstName: 'Prvi',
        lastName: 'Student',
        age: 20,
        gender: 0,
        yearsOfStudy: 1
    },
    {
        firstName: 'Drugi',
        lastName: 'Student',
        age: 21,
        gender:1,
        yearsOfStudy: 2
    }
];

export default class Students {
    constructor() {}
    getStudents() {
        let html = '<ul>';
        studentLists.forEach(student => {
            let item = new Student(student.firstName, student.lastName, student.age, student.gender, student.yearsOfStudy);
            html += item.getStudent();
        });
        html += '</ul>';
        console.log('Init students', countItems(studentLists));
        return html;
    }
}
```

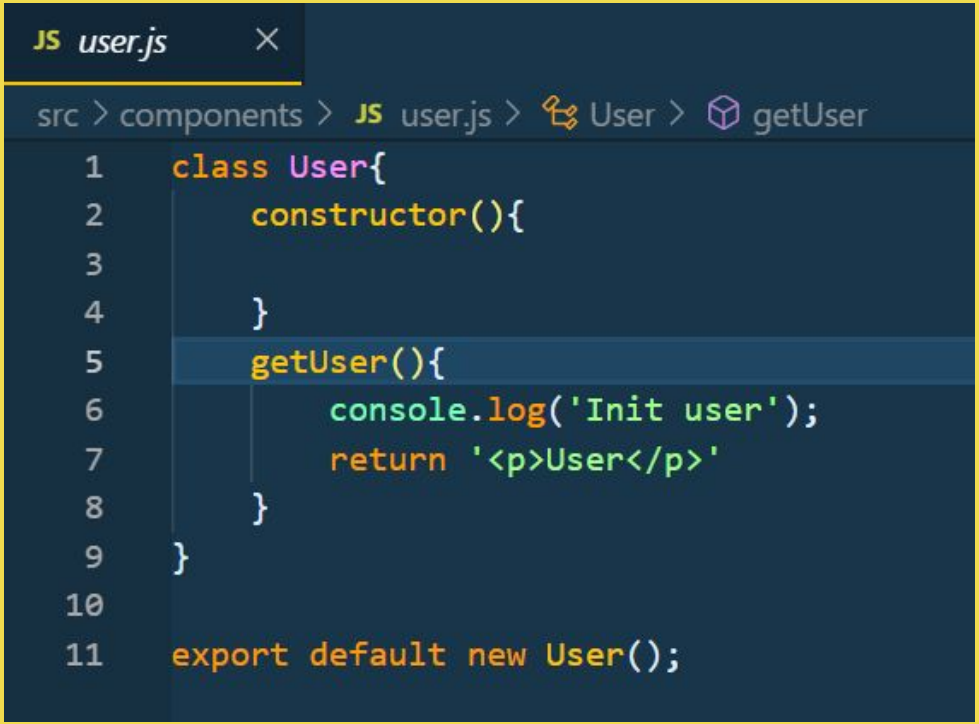
Ubaciti HTML u komponente

content.js

src > layout > content.js > Content > getHtml > students

```
1  import Students from '../components/students'
2
3  class Content{
4      constructor(){
5          console.log('Init content');
6      }
7      getHtml(){
8          const students = new Students();
9          let html = '<div>';
10         html +=  students.getStudents();
11         html += '</div>';
12         console.log('Init content');
13         return html;
14
15     }
16 }
17
18 export default Content;
```

Ubaciti HTML u komponente



Ubaciti HTML u komponente

header.js

src > layout > header.js > Header > getHtml

```
1   import User from '../components/user'
2
3   class Header{
4       constructor(){}
5       getHtml(){
6           console.log('Init header');
7           return `<header>Header${ User.getUser()}</header>`;
8
9       }
10  }
11
12  export default Header;
```

Ubaciti HTML u komponente

dashboard.js

src > layout > dashboard.js > Dashboard > constructor

```
1   import Header from './header.js'
2   import Content from './content'
3
4   export class Dashboard {
5       constructor() {
6           const dashboard = document.getElementById('dashboard');
7           const header = new Header();
8           const content = new Content();
9           dashboard.innerHTML += '<h1>Dasboard</h1>';
10          dashboard.innerHTML += header.getHtml();
11          dashboard.innerHTML += content.getHtml();
12          console.log('Init dashboard');
13      }
14  }
15
```

Ubaciti HTML u komponente

```
import '../assets/css/style.css';
import { Dashboard } from './layout/dashboard';
const app = document.getElementById('app');
app.innerHTML = `<div id="dashboard"></div>`;
const dashboard = new Dashboard();
```

Zaključak

- Pomoću ES6 modula lako možemo ponovo koristiti, održavati, odvajati i enkapsulirati komponente koje ne menjaju spoljno ponašanje
- Modul je zbirka komponenti
- Komponenta je jedan blok
- Ne pokušavajte ponovo da koristite sve jer je potrebno vreme i resursi, a većinu vremena ne koristimo ponovo
- Nacrtajte arhitektonski dijagram pre ronjenja u kod
- Da bi komponente bile dostupne u drugim datotekama, prvo ih moramo izvoziti, a zatim uvesti
- Korišćenjem index.js fajla možemo stvoriti dinamičke interfejs za brz pristup stvarima koje su nam potrebne s manje koda i manje hijerarhijskih staza.
- Možete koristiti izvoznu instancu tokom izvođenja korišćenjem export new Class()