1. Ներածություն React modules

React ծրագրի առաջին ստեղծում (5 - 10 րոպե)՝

React-ով աշխատելու համար հարկավոր է համակարգչում ունենալ Node.js: Այն անհրաժեշտ է npm-ի (Node Package Manager) հետ աշխատելու համար, որն օգտագործվում է կախվածությունների (արտաքին գրադարաններ) կառավարման համար:
Այն պետք է ներբեռնեք և տեղադրեք պաշտոնական կայքից:

```
npx create-react-app myapp
```

Այս հրամանը կստեղծի նոր նախագիծ myapp անունով, որտեղ կներառվեն React-ի բոլոր անհրաժեշտ մոդուլները:

React -ի հիմնական կիրառվող ֆայլերը (5 - 10 րոպե)՝

Առաջին դասի համար մենք կուսումնասիրենք այս երեք ֆայլերը, որոնք ստեղծվում են npx հրամանով.

- **node_modules** թղթապանակում պահվում են բոլոր գրադարաններն ու մոդուլները, որոնք մեր նախագիծն օգտագործում է:
- **package-lock.json** ֆայլը ստեղծվում է ավտոմատ և ծառայում է բոլոր օգտագործվող փաթեթների և կախվածությունների ստույգ տարբերակները (վերսիաները):
- **package.json** ֆայլը պարունակում է պրոեկտի կոնֆիգուրացիաները, կախվածությունների ցուցակը, տարբերակների (վերսիաների) միջակայքը, սկրիպտները և այլ կարգավորումներ:
- **public** թղթապանակում սովորաբար պահպանում են ստատիկ ֆայլերը: Բոլոր ֆայլերը, որոնք առկա են այս թղթապանակում ուղղակիորեն հասանելի են վերջնական պրոեկտում:
- **Src** թղթապանակը պարունակում է React կոմպոնենտի ողջ լոգիկան: Բոլոր ֆայլերը src թղթապանակում ներառվում են (import) ES6 սինտաքսով:

React -ի առավելությունը JavaScript -ի նկատմամբ (5 - 10 րոպե)՝

`React` գրադարանը բավականաչափ հեշտացնում է `HTML` տարրերի ստեղծումը: Եկեք համեմատենք, թե ինչպես կգրենք `HTML` կոդը՝ օգտագործելով `JavaScript` և `React`:
Սկսենք մաքուր `JavaScript`-ից.

```
const div = document.createElement('div');
const h1 = document.createElement('h1');
h1.classList.add('d1');
h1.innerHTML = 'Hello world';
const button = document.createElement('button');
button.setAttribute('id', 'btn');
button.innerHTML = 'Click me';
```

```
button.addEventListener("click", () => {
  alert('U clicked me')
})
div.append(h1, button);
document.getElementById('root').append(div);
```

Այսինքն, մենք ստեղծեցինք `div` տարր, ավելացրեցինք `h1` և `button`, և դրանք տեղադրեցինք `root` տարրի մեջ:
Հիմա նույնը կներկայացնենք React-ով.

```
const h1 = React.createElement('h1', { className: 'd1' }, 'Hello world');
const button = React.createElement('button', { id: 'btn',  onClick:() => alert('Button clicked!') }, 'Click Me');
const div = React.createElement('div', {}, h1, button);

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(div);
```

Կարևոր առավելություններից է այն փաստը, որ React-ը թույլ է տալիս մեզ գրել կոդ, որը հեշտ ընթեռնելի ինչպես նաև վերահսկելի է, հատկապես մեծ նախագծերում:

React կլասսային կոմպոնենտներ (10 - 15 րոպե)`

Արդեն գիտենք, որ React-ում ամեն ինչ հիմնված է բաղադրիչների (կոմպոնենտների) վրա: Եկեք ստեղծենք մեր առաջին **Home** կոմպոնենտը: Առաջին դասերին մենք կօգտագործենք կլասսային կոմպոնենտները, սակայն հետագայում կանցնենք ֆունկցիոնալ կոմպոնենտներին, քանի որ սկսած React 16.8 ից, երբ հայտնվեցին React հուկերը ֆունկցիոնալ կոմտոնենտները ստացան ավելի մեծ տարածում:
React-ում class կոմպոնենտները մեզ թույլ են տալիս օգտագործել state՝ տվյալների պահպանման համար: Երբ մենք ստեղծում ենք կլասսային կոմպոնենտ, օգտագործում ենք **կոնստրուկտոր**, որը հատուկ ֆունկցիա է և կանչվում է, երբ կոմպոնենտը ստեղծվում է:
Այժմ ուշադրություն դարձրեք, որ մենք օգտագործում ենք `super()` ֆունկցիան:
Երբ մենք ստեղծում ենք class կոմպոնենտ, այն ժառանգում է React.Component-ը: Իսկ `super()`-ը անհրաժեշտ է, որպեսզի կանչի ծնողական class-ի (այս դեպքում՝ React.Component) կոնստրուկտորը: Առանց դրա, մենք չենք կարողանա օգտվել `this`-ից կոմպոնենտի ներսում, և դա կհանգեցնի սխալի:

```
import React from "react";

class Home extends React.Component {
  constructor() {
    super();
    this.state = {
      name: "Ani",
```

```
      users: [
        { name: "Ani1", surname: "Sargsyan1", age: 25 },
        { name: "Ani2", surname: "Sargsyan2", age: 26 },
        { name: "Ani3", surname: "Sargsyan3", age: 27 },
        { name: "Ani4", surname: "Sargsyan4", age: 28 },
        { name: "Ani5", surname: "Sargsyan5", age: 29 },
      ],
    };
  }
  render() {
    return (
      <React.Fragment>
        <h1 className="d1">Hello {this.state.name}</h1>
        <button id="btn" onClick={() => {
            alert('Button clicked!')
        }}>Click me</button>
        <hr />
        {this.state.users.map((elm, i) => {
          return (
            <h3 key={i}>
              {elm.name} {elm.surname} {elm.age}
            </h3>
          );
        })}
      </React.Fragment>
    );
  }
}

export default Home;
```

Այս կոնստրուկտորը տալիս է մեզ հնարավորություն՝ ստեղծելու մեր **state**-ը: **State**-ը այն տեղն է, որտեղ կոմպոնենտը պահպանում է իր տվյալները: Օրինակ, այստեղ մենք պահում ենք `name` և `users` տվյալները:

Կոմպոնենտը վերադարձնում է `JSX` և ավտոմատ կերպով մեր էջը թարմացվում է, երբ մեր state-ը փոփոխվում է:

Հիմա, երբ մենք այս կոդը գրել ենք, պարզապես պետք է ներմուծենք այն մեր հիմնական ֆայլում `(index.js)` և ցուցադրենք.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Home from './Home';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Home />);
```

`<React.StrictMode>` իրականացնում է թեստավորում և ցույց է տալիս կոնսոլում երկու անգամ