

## REACT

### #1 - Indecision APP

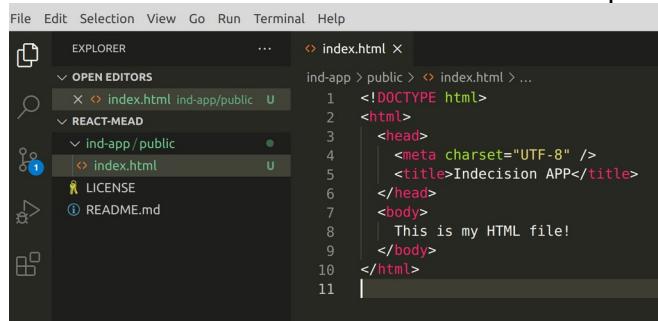
## Table of Contents

3 Start with REACT.....	3
3.1 Basic setup (before REACT) - 007.....	3
3.2 Create a React APP (scripts) - 008.....	4
3.3 Using BABEL to compile JSX to ES5 - 009.....	5
3.4 Exploring JSX - 010.....	6
3.5 JS expressions inside JSX - 011.....	6
3.5 Conditional Rendering in JSX - 012.....	6
3.6 ES6 aside: var vs const and let - 013.....	7
3.6.1 Declaring a variable.....	7
3.6.2 Block and function scoping.....	7
3.7 ES6 aside: arrow functions (part 1) - 014.....	8
3.8 ES6 aside: arrow functions (THIS and MAP()) (part 2) - 015.....	8
3.9 Events and Attributes - 016.....	9
3.10 Manual data binding - 017.....	9
3.11 Forms and Inputs - 018.....	10
3.12 Arrays in JSX (Iteration with map()) - 019.....	10
3.13 A random number. Conditional rendering a button - 020.....	10
3.14 Built a final app - 021.....	11

### 3 Start with REACT

#### 3.1 Basic setup (before REACT) - 007

Create a new repository on GitHub. Clone it to a local folder. Create a simple HTML file.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with a folder named 'REACT-MEAD' containing a 'public' folder which in turn contains an 'index.html' file. The 'index.html' file is open in the main editor area. The code in the editor is:

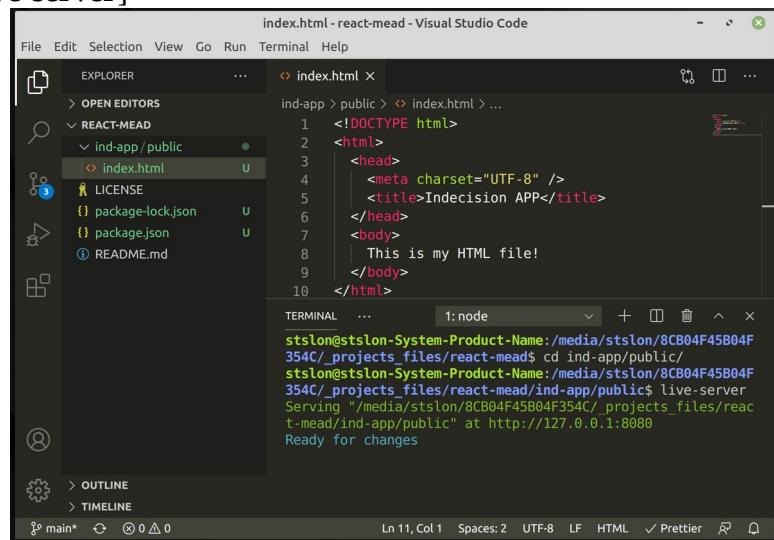
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Indecision APP</title>
</head>
<body>
    This is my HTML file!
</body>
</html>
```

Npm init [CLI=> **npm init -y**]

Install live server [CLI=> **sudo npm install -g live-server**]

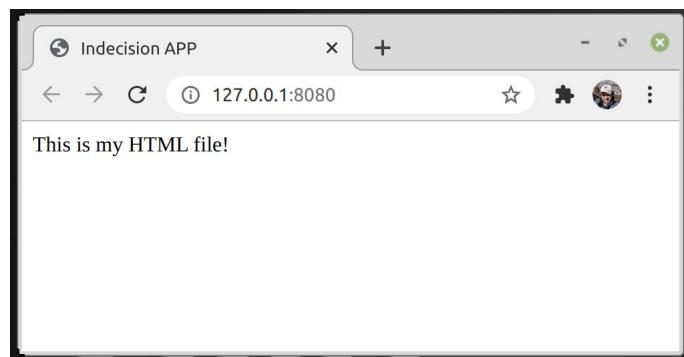
Go to a folder ind-app/public [CLI=> **cd ind-app/public**]

Start a server [CLI=> **live-server**]



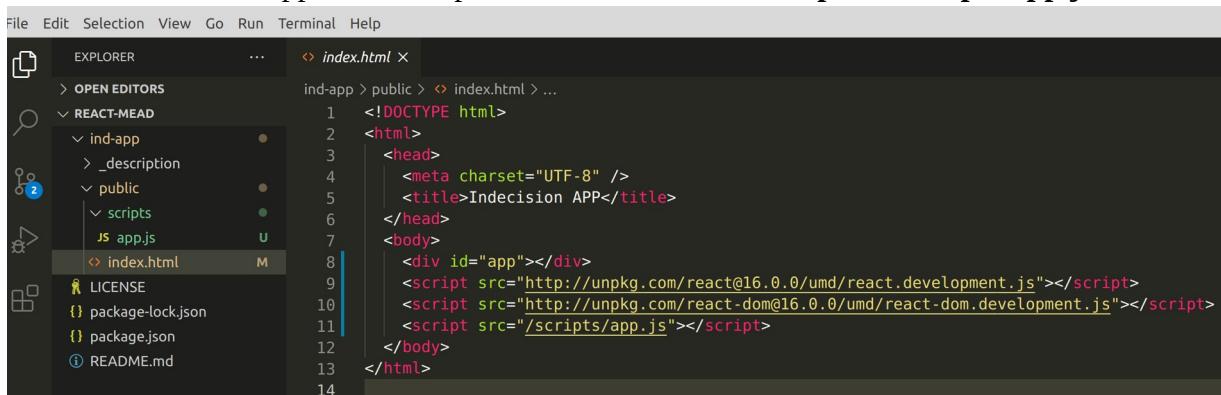
The screenshot shows the Visual Studio Code interface with the terminal tab active at the bottom. The terminal output shows the command 'live-server' being run and the server starting on port 8080. The output text is:

```
stslon@stslon-System-Product-Name:/media/stslon/8CB04F45B04F354C/_projects_files/react-mead$ cd ind-app/public/
stslon@stslon-System-Product-Name:/media/stslon/8CB04F45B04F354C/_projects_files/react-mead/ind-app/public$ live-server
Serving "/media/stslon/8CB04F45B04F354C/_projects_files/react-mead/ind-app/public" at http://127.0.0.1:8080
Ready for changes
```



## 3.2 Create a React APP (scripts) - 008

Add div 'app' and 3 scripts to index.html and create **public/scripts/app.js**



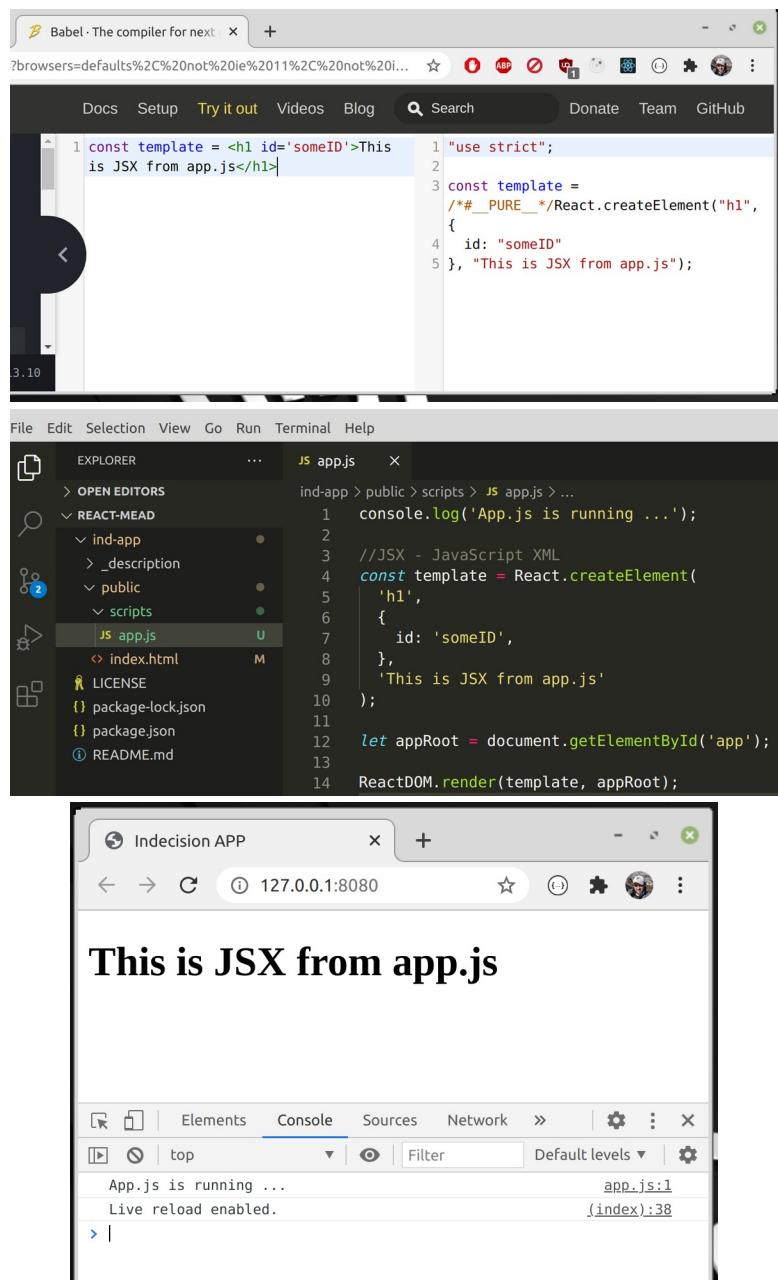
The screenshot shows the VS Code interface. The Explorer sidebar on the left lists files and folders: REACT-MEAD, ind-app, \_description, public, scripts, and index.html. The index.html file is currently open in the editor. Its code is as follows:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Indecision APP</title>
  </head>
  <body>
    <div id="app"></div>
    <script src="http://unpkg.com/react@16.0.0/umd/react.development.js"></script>
    <script src="http://unpkg.com/react-dom@16.0.0/umd/react-dom.development.js"></script>
    <script src="/scripts/app.js"></script>
  </body>
</html>

```

JSX is a JavaScript XML. BABEL is a JavaScript compiler (it converts modern simple ES6 or ES7 to ES5)



The screenshot illustrates the Babel compiler process. It includes:

- Babel - The compiler for next...:** A browser window showing the JSX code: 

```
const template = <h1 id='someID'>This is JSX from app.js</h1>
```

 and its equivalent ES5 output: 

```
"use strict";
const template =
/*#__PURE__*/React.createElement("h1",
{
  id: "someID"
}, "This is JSX from app.js");
```
- VS Code Editor:** Shows the file **JS app.js** with the following code:

```

console.log('App.js is running ...');

//JSX - JavaScript XML
const template = React.createElement(
  'h1',
  {
    id: 'someID',
  },
  'This is JSX from app.js'
);

let appRoot = document.getElementById('app');
ReactDOM.render(template, appRoot);

```

- Browser Preview:** A browser window titled "Indecision APP" at [127.0.0.1:8080](http://127.0.0.1:8080) displaying the rendered JSX: **This is JSX from app.js**.
- Console:** Shows the output of the console.log statement: 

```
App.js is running ...
```

 and a message indicating live reload: 

```
Live reload enabled.
```

### 3.3 Using BABEL to compile JSX to ES5 - 009

Install babel v.6.24.1 [CLI=> **npm i -g babel-cli@6.24.1**]

Install babel-preset-react v.6.24.1 and babel-preset-env v.1.5.2 [CLI=> **sudo npm i babel-preset-react@6.24.1 babel-preset-env@1.5.2**]

Now we are able to use JSX.

Create a file **src/app.js** with JSX

```
Js app.js x
ind-app > src > Js app.js > ...
1   console.log('App.js is running ...');
2
3 //JSX - JavaScript XML
4 const template = (
5   <div>
6     <h1>Welcome to Indecision APP</h1>
7     <p>There is my test project for REACT course by Andrew Mead</p>
8     <ol>
9       <li>Item #1</li>
10      <li>Item #2</li>
11      <li>Item #3</li>
12      <li>Item #4</li>
13      <li>Item #5</li>
14    </ol>
15  </div>
16);
17
18 let appRoot = document.getElementById('app');
19
20 ReactDOM.render(template, appRoot);
21
```

USE BABEL with CLI: [CLI=> **babel src/app.js --out-file=public/scripts/app.js –presets=env,react --watch**]

This command will create **public/scripts/app.js** with ES5 syntax of **src/app.js** and will track all the changes on **src/app.js** and immediately change **public/scripts/app.js**

```
Js app.js •
ind-app > public > scripts > Js app.js > ...
1 'use strict';
2
3 console.log('App.js is running ...');
4 //JSX - JavaScript XML
5 var template = React.createElement('div', null,
6
7   React.createElement('h1', null, 'Welcome to Indecision APP'),
8   React.createElement('p', null, 'There is my test project for REACT course by Andrew Mead'),
9
10  React.createElement('ol', null,
11    React.createElement('li', null, 'Item #1'),
12    React.createElement('li', null, 'Item #2'),
13    React.createElement('li', null, 'Item #3'),
14    React.createElement('li', null, 'Item #4'),
15    React.createElement('li', null, 'Item #5')
16  );
17
18 var appRoot = document.getElementById('app');
19
20 ReactDOM.render(template, appRoot);
21
```

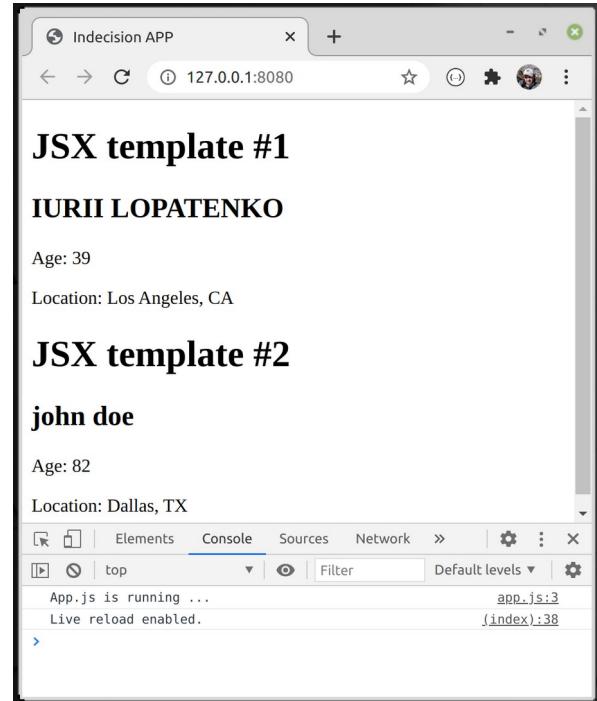
## 3.4 Exploring JSX - 010

Install VSCode extension BABEL ES6/ES7.

We have to pass to a RENDER all the information in **1 root tag**.

## 3.5 JS expressions inside JSX - 011

```
JS app.js  X
ind-app > src > JS app.js ...
1  console.log('App.js is running ...');
2
3 //JSX - JavaScript XML
4
5 let userName = 'Iurii Lopatenko';
6 let userAge = 34;
7 let userLocation = 'Los Angeles, CA';
8
9 let user = {
10   name: 'John Doe',
11   age: 41,
12   location: 'Dallas, TX',
13 };
14 const templateTwo = (
15   <div>
16     <h1>JSX template #1</h1>
17     <h2>{userName.toUpperCase()}</h2>
18     <p>Age: {userAge + 5}</p>
19     <p>Location: {userLocation}</p>
20
21     <h1>JSX template #2</h1>
22     <h2>{user.name.toLowerCase()}</h2>
23     <p>Age: {user.age * 2}</p>
24     <p>Location: {user.location}</p>
25   </div>
26 );
27
28 let appRoot = document.getElementById('app');
29
30 ReactDOM.render(templateTwo, appRoot);
31
```



## 3.5

## Conditional Rendering in JSX - 012

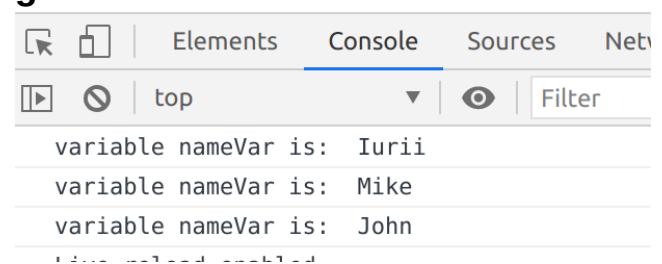
```
JS app.js  X
ind-app > src > JS app.js ...
1 //JSX - JavaScript XML
2 //012 - THEORY
3 let user = {
4   name: 'Iurii Lopatenko',
5   age: 34,
6   location: 'Los Angeles, CA, USA',
7 };
8
9 //Ternary operator
10 const getLocation = user.location ? (
11   <p>Location: {user.location}</p>
12 ) : undefined;
13
14 //IF
15 const checkUserName = (nameToCheck) => {
16   if (nameToCheck) {
17     return nameToCheck;
18   } else {
19     return 'ANONYMOUS';
20   }
21 };
22
23 const templateTheory = (
24   <div>
25     <h1>JSX template</h1>
26     <h2>{checkUserName(user.name)}</h2>
27     {/*Logical compare*/}
28     {user.age && user.age >= 18 && <p>Age: {user.age}</p>}
29     {getLocation}
30   </div>
31 );
```

```
JS app.js  X
ind-app > src > JS app.js ...
1 //012 PRACTICE
2 const app = {
3   title: 'Indecision App',
4   subtitle: 'There is something subtitle for an APP ...',
5   options: ['One', 'Two', 'Three'],
6 };
7
8 let templatePractice = (
9   <div>
10     <h1>{app.title}</h1>
11     {app.subtitle && <p>{app.subtitle}</p>}
12     {app.options.length > 0 ? 'Here are your options:' : 'No any options'}
13     {app.options.map((element, index) => (
14       <p key={index}>
15         Option #{index + 1} is: {element}
16       </p>
17     ))}
18   </div>
19 );
20
21 let appRoot = document.getElementById('app');
22 ReactDOM.render(templatePractice, appRoot);
23
```

## 3.6 ES6 aside: var vs const and let - 013

### 3.6.1 Declaring a variable

```
js es6-let-const.js ×
ind-app > src > playground > js es6-let-const.js > ...
1  var nameVar = 'Iurii';
2  console.log('variable nameVar is: ', nameVar);
3  var nameVar = 'Mike';
4  console.log('variable nameVar is: ', nameVar);
5  nameVar = 'John';
6  console.log('variable nameVar is: ', nameVar);
7
```



LET: we are **not able** to declare the same variable more than 1 time. We **can** declare a variable without any value: let variable; variable = 10;

```
7
8  let nameLet = 'Iurii';
9  console.log('variable nameLet is: ', nameLet);
10 let nameLet = 'Mike';
11 console.log('variable nameLet is: ', nameLet);
12 |
```

```
TypeError: src/playground/es6-let-const.js: Duplicate declaration "nameLet"
8  let nameLet = 'Iurii';
9  console.log('variable nameLet is: ', nameLet);
> 10 let nameLet = 'Mike';
   ^                                          |
11  console.log('variable nameLet is: ', nameLet);
```

CONST: we are **not able** to declare the same variable more than 1 time or change this variable. We **can not** declare a variable without any value: const variable; variable = 10;

```
17 //CONST
18 console.log('CONST');
19 const nameConst = 'Iurii';
20 console.log('variable nameConst is: ', nameConst);
21 nameConst = 'Frank';
22 console.log('variable nameConst is: ', nameConst);
23 const nameConst = 'Mike';
24 |
```

```
SyntaxError: src/playground/es6-let-const.js: "nameConst" is read-only
19 | const nameConst = 'Iurii';
20 | console.log('variable nameConst is: ', nameConst);
> 21 | nameConst = 'Frank';
   ^
22 | console.log('variable nameConst is: ', nameConst);
23 |
change src/playground/es6-let-const.js
TypeError: src/playground/es6-let-const.js: Duplicate declaration "nameConst"
21 | nameConst = 'Frank';
22 | console.log('variable nameConst is: ', nameConst);
> 23 | const nameConst = 'Mike';
   ^
```

### 3.6.2 Block and function scoping

```
25 //FUNCTION SCOPING
26 let functionLet = () => {
27   let justName = 'Mike';
28   return justName;
29 };
30 console.log(functionLet());
31 //console.log(justName); // justName is not defined
32
33 //BLOCK SCOPING
34 {
35   let variable = 'some text';
36 }
37 console.log(variable); //variable is not defined
```

## 3.7 ES6 aside: arrow functions (part 1) - 014

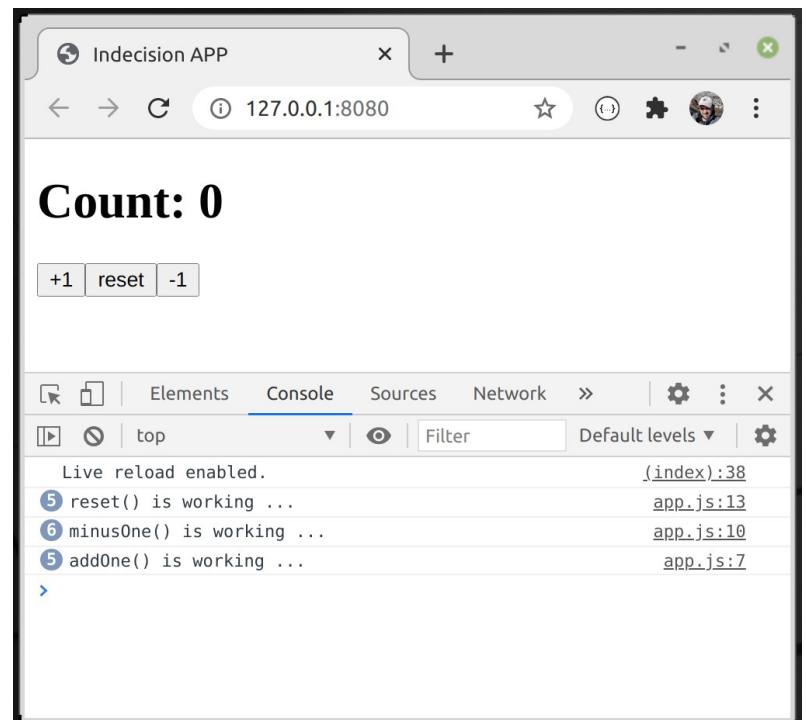
```
js es6-arrow-function.js ×
ind-app > src > playground > JS es6-arrow-function.js > ...
1  console.log('Hello from es6-arrow-function.js!');
2
3 //ES5 function
4 const square = function (n) {
5   return n * n;
6 };
7
8 function squareV2(x) {
9   return x * x;
10}
11
12 let result = square(25);
13 console.log('Result of ES5 function is: ', result);
14
15 //ES6 (ARROW) function
16 const square6 = (n) => n * n;
17 let result6 = square6(25);
18 console.log('Result of ES6 (ARROW) function is: ', result);
19
20 //test
21 let fullName = 'Iurii Lopatenko';
22 const getFirstName = (fullName) => fullName.split(' ')[0];
23 console.log(getFirstName(fullName));
```

## 3.8 ES6 aside: arrow functions (THIS and MAP()) (part 2) - 015

```
js es6-arrow-function-2.js ×
ind-app > src > playground > JS es6-arrow-function-2.js > ...
1  console.log('Hello from es6-arrow-function-2.js!');
2
3 //Arguments object - no longer bound with arrow functions
4 //
5 //ES5 function
6 const add = function (a, b) {
7   console.log(arguments);
8   console.log(
9     `You want to know how much will be ${a} plus ${b} .... I think it should be something like ${a + b}`);
10
11};
12
13);
14 add(5, 7);
15
16 //ES6 function
17 let add2 = (a, b) => {
18   //console.log(arguments); there will be an error!
19   console.log(
20     `You want to know how much will be ${a} plus ${b} .... I think it should be something like ${a + b}`);
21
22);
23
24);
25 add2(300, 505);
26
27 //THAT in ES5
28 console.log('THAT in ES5');
29 const userThat = {
30   name: 'Iurii',
31   cities: ['Moscow', 'Los Angeles'],
32   printPlaces: function () {
33     console.log(this.name);
34     console.log(this.cities);
35     const that = this;
36     this.cities.forEach(function (city) {
37       console.log(that.name + ' has lived in ' + city);
38     });
39   },
40 };
41 userThat.printPlaces();
42
43 /* //THIS in ES5
44 console.log('THIS in ES5');
45 const userThis = {
46   name: 'Iurii',
47   cities: ['Moscow', 'Los Angeles'],
48   printPlaces: function () {
49     console.log(this.name);
50     console.log(this.cities);
51     this.cities.forEach(function (city) {
52       console.log(this.name + ' has lived in ' + city); //there will be an error
53     });
54   },
55 };
56 userThis.printPlaces(); */
57
58 //THAT in ES6
59 console.log('THAT in ES6');
60 const userThisES6 = {
61   name: 'Iurii',
62   cities: ['Moscow', 'Los Angeles'],
63   printPlaces: function () {
64     console.log(this.name);
65     console.log(this.cities);
66     this.cities.forEach((city) => {
67       console.log(this.name + ' has lived in ' + city);
68     });
69   },
70 };
71 userThisES6.printPlaces();
72
73 /* //THIS in ES6 - all arrow - ERROR!!!
74 console.log('THIS in ES6 - all arrow');
75 const userThisES6Arrow = {
76   name: 'Iurii',
77   cities: ['Moscow', 'Los Angeles'],
78   printPlaces: () => {
79     console.log(this.name);
80     console.log(this.cities);
81     this.cities.forEach((city) => {
82       console.log(this.name + ' has lived in ' + city);
83     });
84   },
85 };
86 userThisES6Arrow.printPlaces(); */
87
88 //THIS in ES6 - no arrow - IT WORKS!!!
89 console.log('THAT in ES6 - no arrow');
90 const userThisES6NoArrow = {
91   name: 'Iurii',
92   cities: ['Moscow', 'Los Angeles'],
93   printPlaces() {
94     //TEST of MAP method
95     console.log('Test:');
96     const uppercase = this.cities.map(
97       (city) => `${this.name.toUpperCase()} has lived in ${city.toUpperCase()}`;
98     );
99     console.log(uppercase);
100    //END of TEST
101
102    console.log(this.name);
103    console.log(this.cities);
104    this.cities.forEach((city) => {
105      console.log(this.name + ' has lived in ' + city);
106    });
107  },
108};
109 userThisES6NoArrow.printPlaces();
110
111 //CHALLENGE
112 console.log('CHALLENGE');
113 const multiplier = {
114   numbers: [5, 4, 7, 45, 46, 97, 125, 10, 5, 456],
115   multiplyBy: 456,
116   multiply() {
117     return this.numbers.map((number) => number * this.multiplyBy);
118   },
119 };
120 console.log(multiplier.multiply());
```

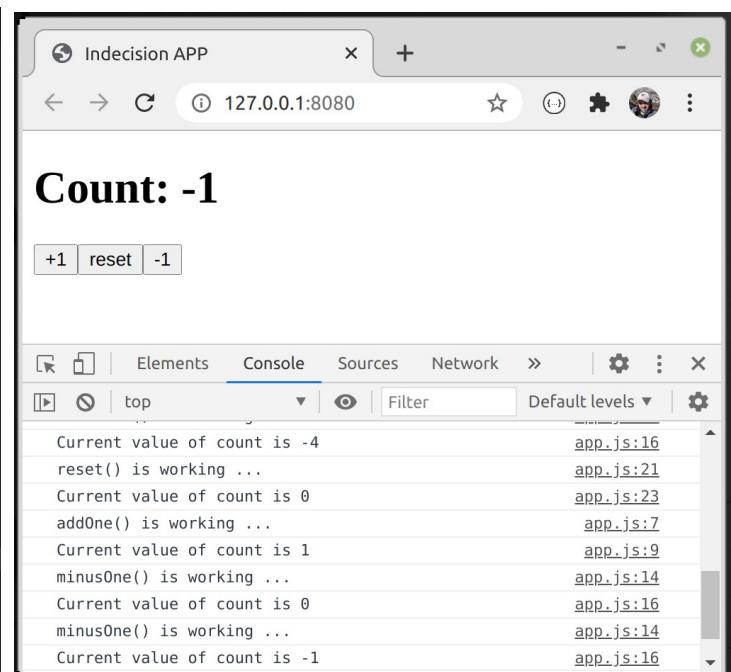
## 3.9 Events and Attributes - 016

```
js app.js  x
ind-app > src > js app.js > ...
1 //016 - Events and Attributes
2 let count = 0;
3
4 const addOne = () => {
5   console.log('addOne() is working ...');
6 };
7 const minusOne = () => {
8   console.log('minusOne() is working ...');
9 };
10 const reset = () => {
11   console.log('reset() is working ...');
12 };
13
14 const template016 = (
15   <div>
16     <h1>Count: {count}</h1>
17     <button onClick={addOne}> +1 </button>
18     <button onClick={reset}> reset </button>
19     <button onClick={minusOne}> -1 </button>
20   </div>
21 );
22
23 const appRoot = document.getElementById('app');
24 ReactDOM.render(template016, appRoot);
25
```



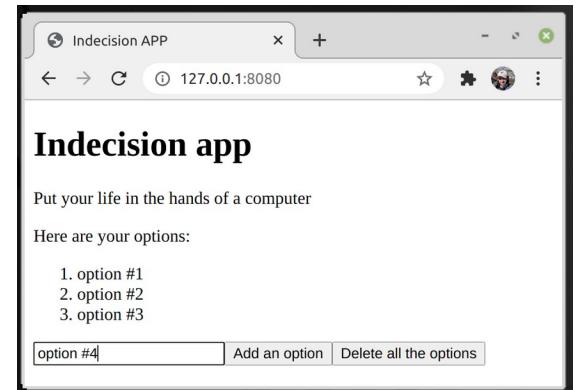
## 3.10 Manual data binding - 017

```
js app.js  x
ind-app > src > js app.js > [e] reset
1 //016 - Events and Attributes
2 let count = 0;
3
4 const addOne = () => {
5   console.log('addOne() is working ...');
6   count += 1;
7   console.log(`Current value of count is ${count}`);
8   renderCounter();
9 };
10
11 const minusOne = () => {
12   console.log('minusOne() is working ...');
13   count -= 1;
14   console.log(`Current value of count is ${count}`);
15   renderCounter();
16 };
17
18 const reset = () => {
19   console.log('reset() is working ...');
20   count = 0;
21   console.log(`Current value of count is ${count}`);
22   renderCounter();
23 };
24
25 const appRoot = document.getElementById('app');
26
27 const renderCounter = () => {
28   const template016 = (
29     <div>
30       <h1>Count: {count}</h1>
31       <button onClick={addOne}> +1 </button>
32       <button onClick={reset}> reset </button>
33       <button onClick={minusOne}> -1 </button>
34     </div>
35   );
36   ReactDOM.render(template016, appRoot);
37 };
38
39 renderCounter();
```



## 3.11 Forms and Inputs - 018

```
JS app.js
ind-app > src > JS app.js > ...
1  console.log('App.js is running ...');
2  const app = {
3    title: 'Indecision app',
4    subtitle: 'Put your life in the hands of a computer',
5    options: [],
6  };
7  const deleteOptions = () => {
8    app.options = [];
9    renderApp();
10 };
11 const onFormSubmit = (e) => {
12   e.preventDefault();
13   const option = e.target.elements.option.value;
14   if (option) {
15     app.options.push(option);
16     e.target.elements.option.value = '';
17     renderApp();
18   }
19 };
20 const renderApp = () => {
21   const template = (
22     <div>
23       <h1>{app.title}</h1>
24       {app.subtitle && <p>{app.subtitle}</p>}
25       <p>{app.options.length > 0 ? 'Here are your options: ' : 'No any options'}</p>
26       <ol>{app.options.map((eachOption, index) => (<li key={index}>{eachOption}</li>))}</ol>
27       <form onSubmit={onFormSubmit}>
28         <input type='text' name='option'>
29         <button>Add an option</button>
30         <button onClick={deleteOptions}>Delete all the options</button>
31       </form>
32     </div>
33   );
34   ReactDOM.render(template, appRoot);
35 };
36 const appRoot = document.getElementById('app');
37 renderApp();
38 
```



## 3.12 Arrays in JSX (Iteration with map()) - 019

## 3.13 A random number. Conditional rendering a button - 020

```
JS app.js
ind-app > src > JS app.js > ...
1  const app = {
2    title: 'Indecision app',
3    subtitle: 'Put your life in the hands of a computer',
4    options: [],
5  };
6  const deleteOptions = () => {
7    app.options = [];
8    renderApp();
9  };
10 const onMakeDecision = () => {
11   const randomNum = Math.floor(Math.random() * app.options.length);
12   const newOption = app.options[randomNum];
13   alert(newOption);
14 };
15 const onFormSubmit = (e) => {
16   e.preventDefault();
17   const option = e.target.elements.option.value;
18   if (option) {
19     app.options.push(option);
20     e.target.elements.option.value = '';
21     renderApp();
22   }
23 };
24 const renderApp = () => {
25   const template = (
26     <div>
27       <h1>{app.title}</h1>
28       {app.subtitle && <p>{app.subtitle}</p>}
29       <p>{app.options.length > 0 ? 'Here are your options: ' : 'No any options'}</p>
30       <ol>{app.options.map((eachOption, index) => (<li key={index}>{eachOption}</li>))}</ol>
31       <button disabled={app.options.length === 0} onClick={onMakeDecision}>What should I do?</button>
32       <button onClick={deleteOptions}>Delete all the options</button>
33       <form onSubmit={onFormSubmit}>
34         <input type='text' name='option'>
35         <button>Add option</button>
36       </form>
37     </div>
38   );
39   ReactDOM.render(template, appRoot);
40 };
41 const appRoot = document.getElementById('app');
42 renderApp();
43 
```

## 3.14 Built a final app - 021

App will contain a simple h1 header with a name of app and a button. Text of button will change between ‘show details’ and ‘hide details’ (after each click). When button text will be ‘show detail’ next to button will be created a paragraph with a short string. If button text will be ‘hide details’ paragraph will be hided.

```
JS build-it-visible.js X
ind-app > src > playground > JS build-it-visible.js > ...
1  let isVisible = false;
2  const changeIsVisible = () => {
3    isVisible = !isVisible;
4    appRender();
5  };
6
7  const appRoot = document.getElementById('app');
8
9  const appRender = () => {
10    const template = (
11      <div>
12        <h1>Visibiliti Toggle</h1>
13        <button onClick={changeIsVisible}>
14          {isVisible ? 'Hide details' : 'Show details'}
15        </button>
16        {isVisible && <p>There are some details ...</p>}
17      </div>
18    );
19    ReactDOM.render(template, appRoot);
20  };
21  appRender();
```

