

REACT

#1 - Indecision APP

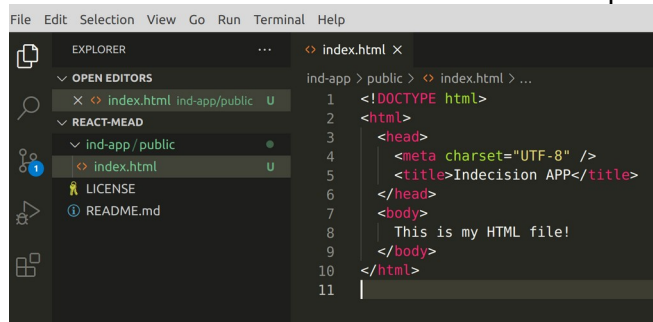
Table of Contents

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
.....	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 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 panel on the left shows a project structure with a folder 'ind-app' containing a subfolder 'public'. Inside 'public', there is a file 'index.html'. The main editor area shows the content of 'index.html' with the following code:

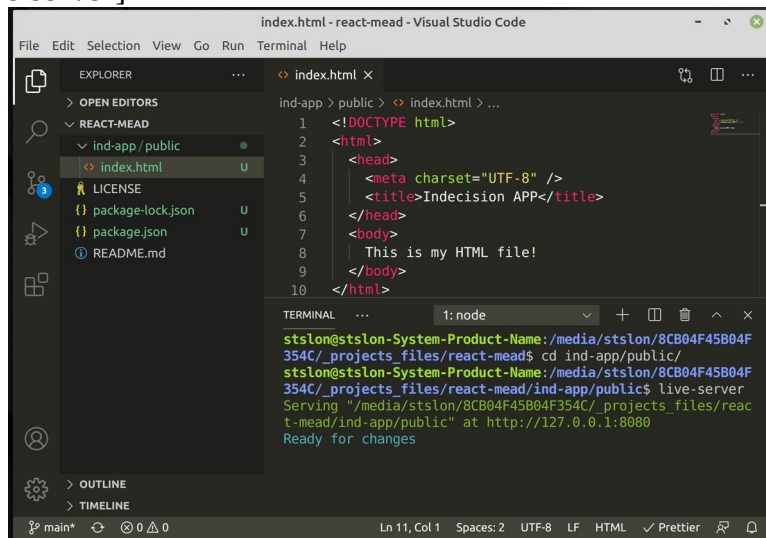
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>Indecision APP</title>
6   </head>
7   <body>
8     This is my HTML file!
9   </body>
10 </html>
```

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

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

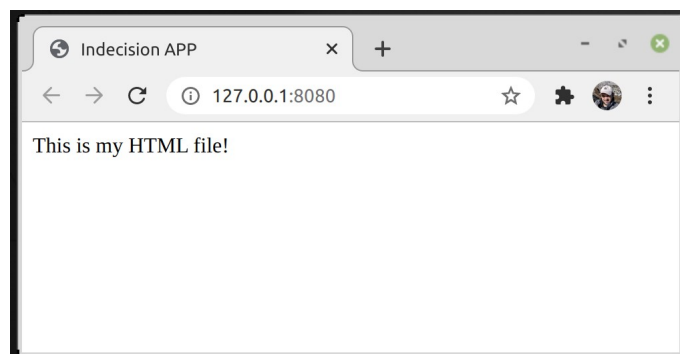
Go to a folder ind-app/public [CLI=> **cd ind-app/public**]

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



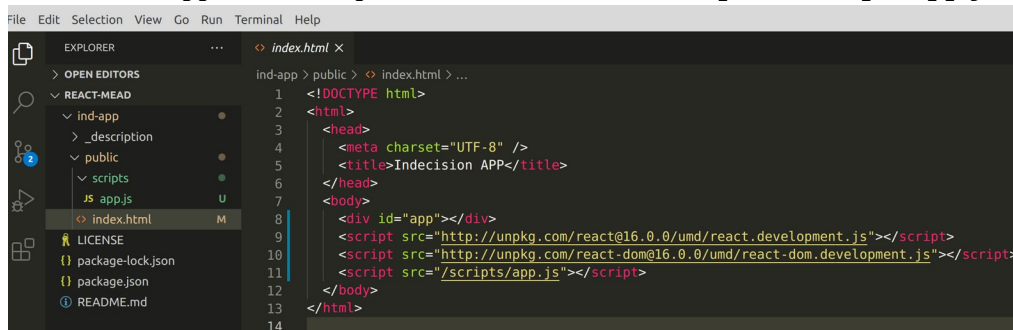
The screenshot shows the Visual Studio Code interface with the terminal open. The terminal output shows the command 'live-server' being executed and the server starting on http://127.0.0.1:8080. The Explorer panel on the left shows the project structure with a folder 'ind-app' containing a subfolder 'public'. Inside 'public', there is a file 'index.html'. The main editor area shows the content of 'index.html' with the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>Indecision APP</title>
6   </head>
7   <body>
8     This is my HTML file!
9   </body>
10 </html>
```



3.2 Create a React APP (scripts) - 008

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

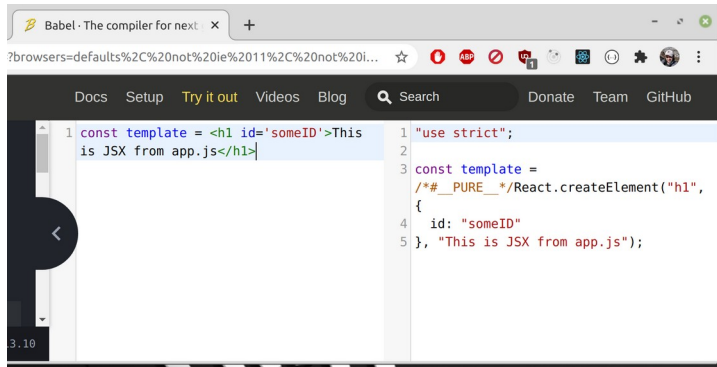


```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8" />
5     <title>Indecision APP</title>
6   </head>
7   <body>
8     <div id="app"></div>
9     <script src="http://unpkg.com/react@16.0.0/umd/react.development.js"></script>
10    <script src="http://unpkg.com/react-dom@16.0.0/umd/react-dom.development.js"></script>
11    <script src="/scripts/app.js"></script>
12  </body>
13 </html>
14

```

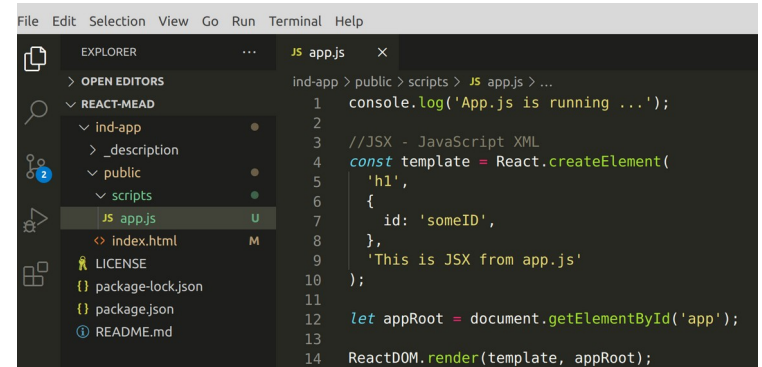
JSX is a JavaScript XML. BABEL is a JavaScript compiler (it converts modern simple ES6 or ES7 to ES5)



```

1 const template = <h1 id='someID'>This
  is JSX from app.js</h1>
2
3 const template =
4   /* _PURE */ React.createElement("h1",
5     { id: "someID" }, "This is JSX from app.js");

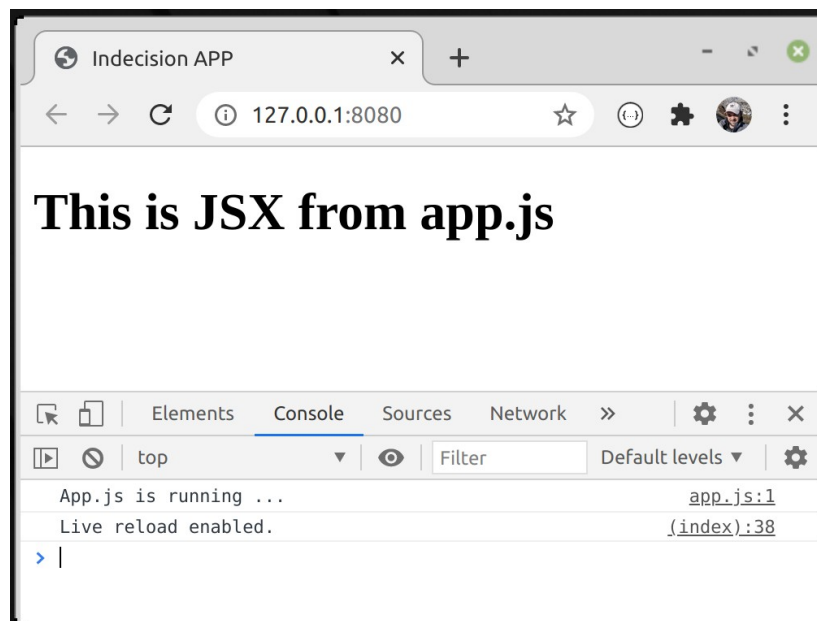
```



```

1 console.log('App.js is running ...');
2
3 //JSX - JavaScript XML
4 const template = React.createElement(
5   'h1',
6   {
7     id: 'someID',
8   },
9   'This is JSX from app.js'
10 );
11
12 let appRoot = document.getElementById('app');
13
14 ReactDOM.render(template, appRoot);

```



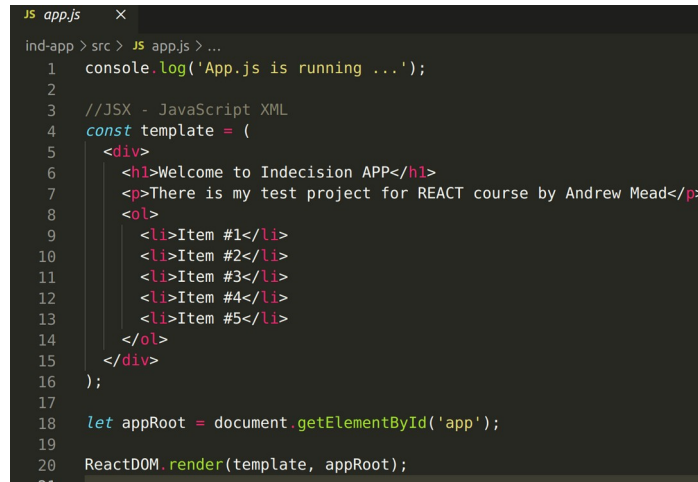
3.3 Using BABEL to compile JSX to ES5 - 009

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



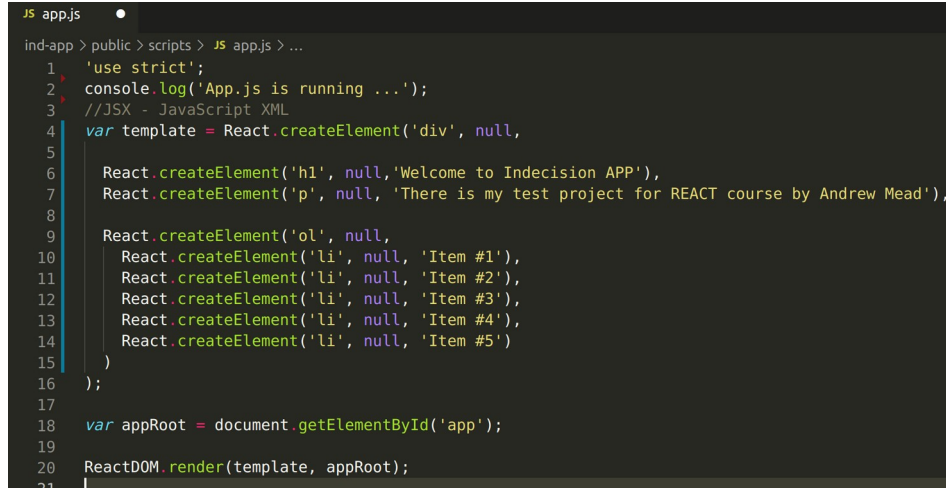
```

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**



```

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

```

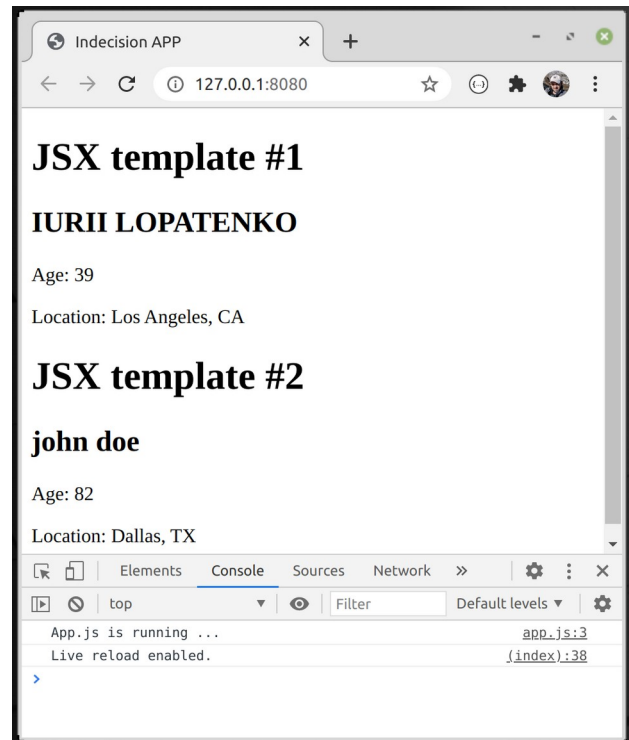
3.4 Exploring JSX - 010

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 );
32
```

```
JS app.js x
ind-app > src > JS app.js > ...
33 //012 PRACTICE
34 const app = {
35   title: 'Indecision App',
36   subtitle: 'There is something subtitle for an APP ...',
37   options: ['One', 'Two', 'Three'],
38 };
39
40 let templatePractice = (
41   <div>
42     <h1>{app.title}</h1>
43     {app.subtitle && <p>{app.subtitle}</p>}
44     {app.options.length > 0 ? 'Here are your options:' : 'No any options'}
45     {app.options.map((element, index) => (
46       <p key={index}>
47         Option #{index + 1} is: {element}
48       </p>
49     ))}
50   </div>
51 );
52 let appRoot = document.getElementById('app');
53 ReactDOM.render(templatePractice, appRoot);
54
```

3.6 ES6 aside: var vs const and let - 013

3.6.1 Declaring a variable

```
JS es6-let-const.js X
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 X
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 X
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 ${
10    |   a + b
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 ${
21     |   a + b
22     | }');
23   | };
24 };
25 add2(300, 505);
26
27 //THAT in ES5
28 console.log('THAT in ES5');
29 const userThat = {
30   | name: 'Iurii',
31   | cities: ['Moscow', 'Los Angeles'],
32   | printPlaces: function () {
33     | console.log(this.name);
34     | console.log(this.cities);
35     | const that = this;
36     | this.cities.forEach(function (city) {
37       | console.log(that.name + ' has lived in ' + city);
38     | });
39   | },
40 };
41 userThat.printPlaces();
42
43 //THIS in ES6 - no arrow - IT WORKS!!!
44 console.log('THAT in ES6 - no arrow');
45 const userThisES6NoArrow = {
46   | name: 'Iurii',
47   | cities: ['Moscow', 'Los Angeles'],
48   | printPlaces() {
49     | //TEST of MAP method
50     | console.log('Test:');
51     | const upperCase = this.cities.map(
52     |   | (city) => `${this.name.toUpperCase()} has lived in ${city.toUpperCase()}`
53     | );
54     | console.log(upperCase);
55     | //END of TEST
56   | },
57   | console.log(this.name);
58   | console.log(this.cities);
59   | this.cities.forEach((city) => {
60     | console.log(this.name + ' has lived in ' + city);
61   | });
62   | },
63 };
64 userThisES6NoArrow.printPlaces();
65
66 //CHALLENGE
67 console.log('CHALLENGE');
68 const multiplier = {
69   | numbers: [5, 4, 7, 45, 46, 97, 125, 10, 5, 456],
70   | multiplyBy: 456,
71   | multiply() {
72     | return this.numbers.map((number) => number * this.multiplyBy);
73   | },
74 };
75 console.log(multiplier.multiply());

```

```

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 //THIS 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('THAT 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(); */

```


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
```

