

REACT

#1 - Indecision APP

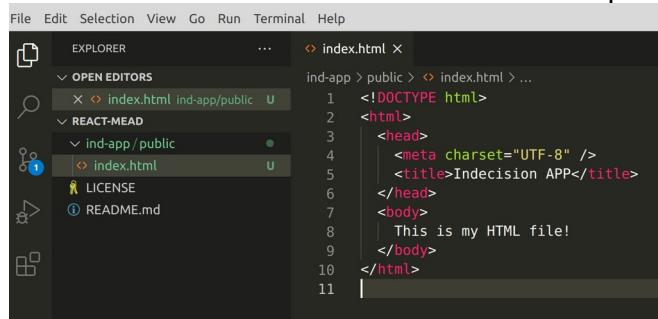
Table of Contents

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 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:

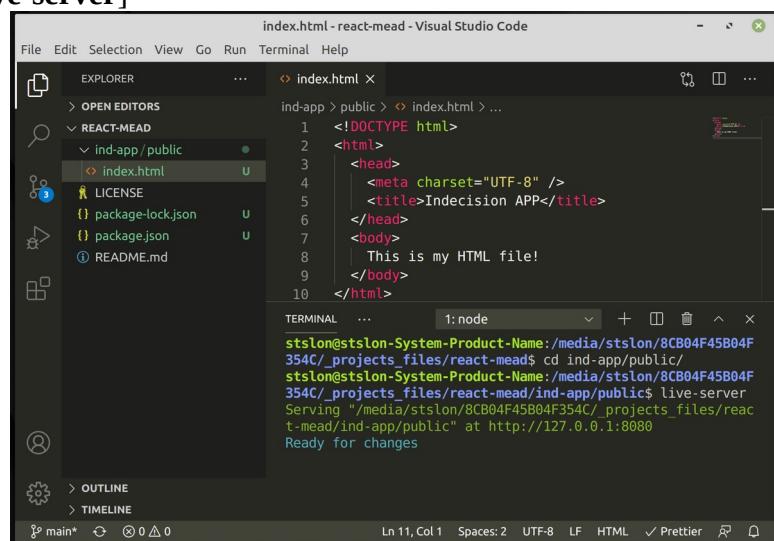
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Indecision APP</title>
</head>
<body>
    This is my HTML file!
</body>
</html>
```

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

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

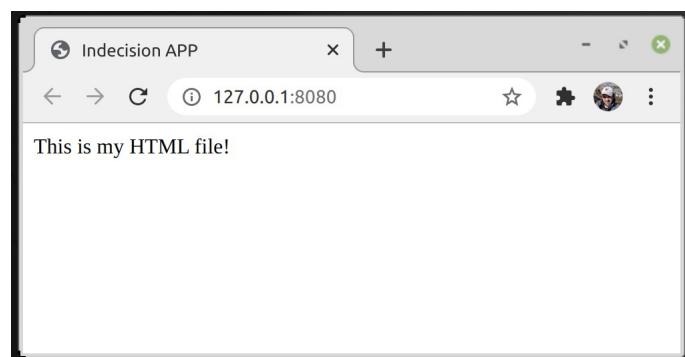
Go to a folder ind-app/public [CLI=> **cd ind-app/public**]

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



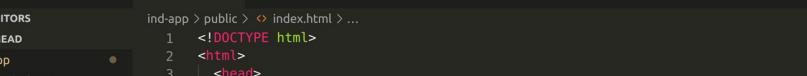
The screenshot shows the Visual Studio Code interface with the terminal tab active at the bottom. The terminal output shows the command 'live-server' being run and the server starting on port 8080. The output text is:

```
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 a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Explorer:** Shows the project structure:
 - OPEN EDITORS
 - REACT-MEAD
 - ind-app
 - _description
 - public
 - scripts
 - app.js
 - index.html
 - LICENSE
 - package-lock.json
 - package.json
 - README.md
- Content Area:** The file `index.html` is open, showing its source code:

```
<!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 shows the Babel IDE interface. On the left, the code editor displays the following JSX code:

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

On the right, the file tree and code editor show the generated JavaScript code:

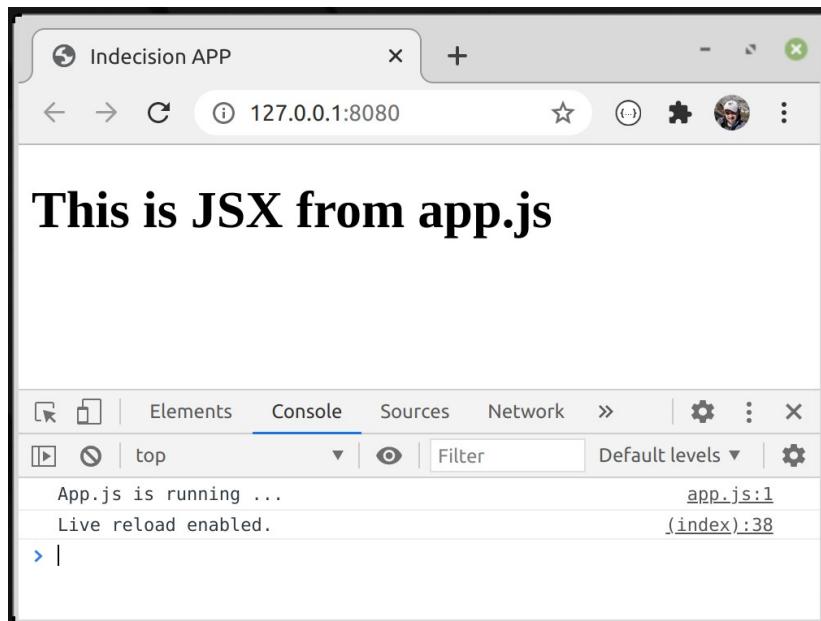
```
"use strict";  
  
const template =  
/*#__PURE__*/React.createElement("h1",  
, {  
id: "someID"  
}, "This is JSX from app.js");
```

The file tree shows the project structure:

- ind-app
- > _description
- > public
- > scripts
- > **JS app.js**
- > index.html
- > LICENSE
- > package-lock.json
- > package.json
- > README.md

The code editor on the right shows the generated JavaScript 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);
```



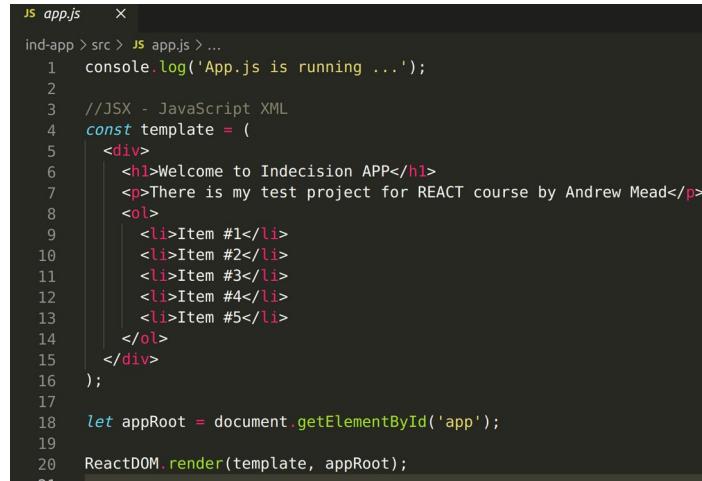
3.3 Using BABEL to compile JSX to ES5 - 009

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 bable-preset-react@6.24.1 babel-preset-env@1.5.2**]

Now we are able to use JSX.

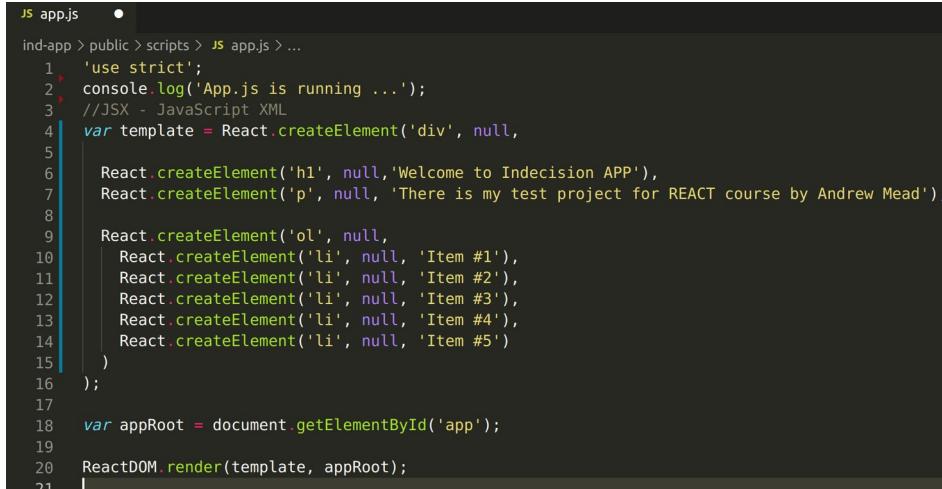
Create a file **src/app.js** with JSX



```
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
```

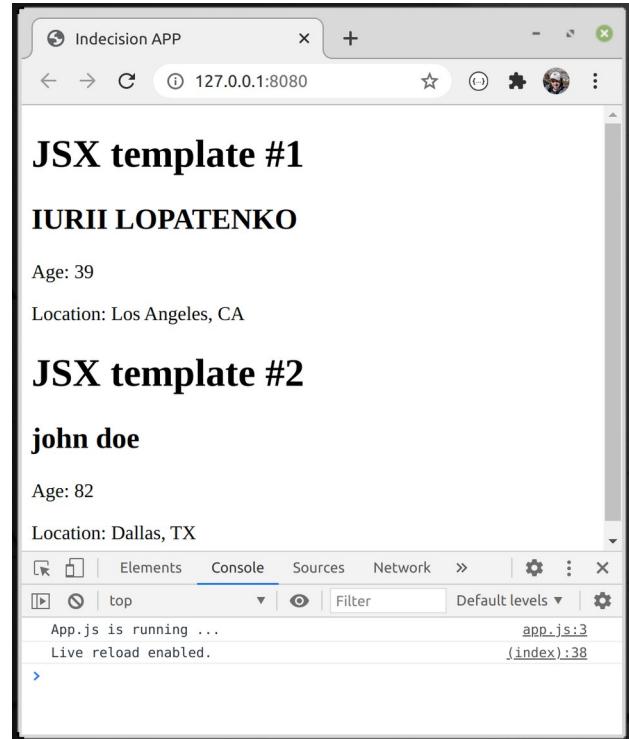
3.4 Exploring JSX - 010

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

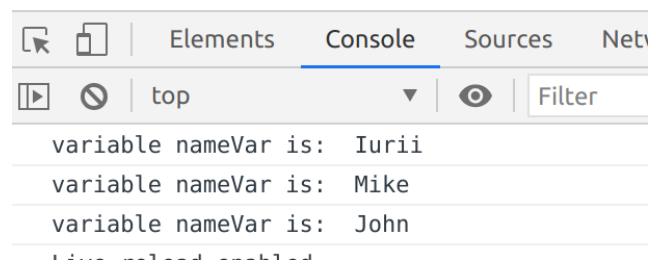
```
JS app.js  X
ind-app > src > JS app.js > ...
1 //JSX - JavaScript XML
2 //012 - THEORY
3 let user = {
4   name: 'Iurii Lopatenko',
5   age: 34,
6   location: 'Los Angeles, CA, USA',
7 };
8
9 //Ternary operator
10 const getLocation = user.location ? (
11   <p>Location: {user.location}</p>
12 ) : undefined;
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 > ...
32
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);
```

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
```

