REACT

#1 - Indecision APP

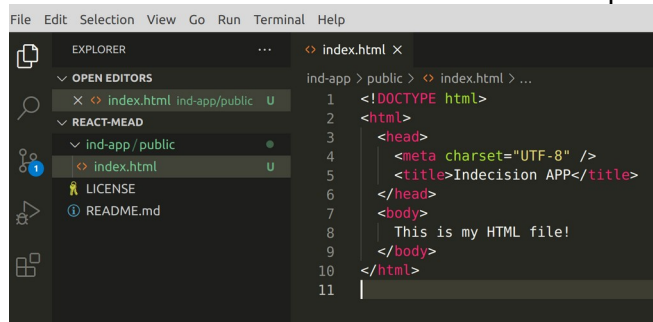# Table of Contents

3 Start with REACT
3.1 Basic setup (before REACT) - 007

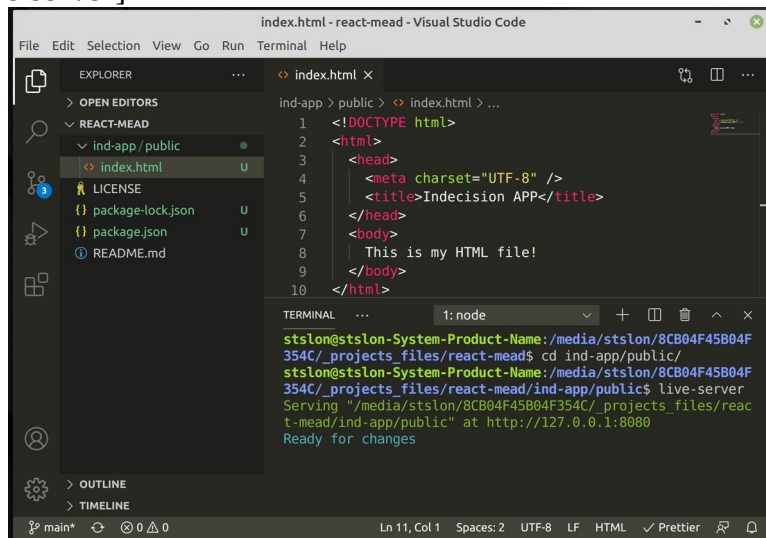Create a new repository on GitHub. Clone it to a local folder. Create a simple HTML file.
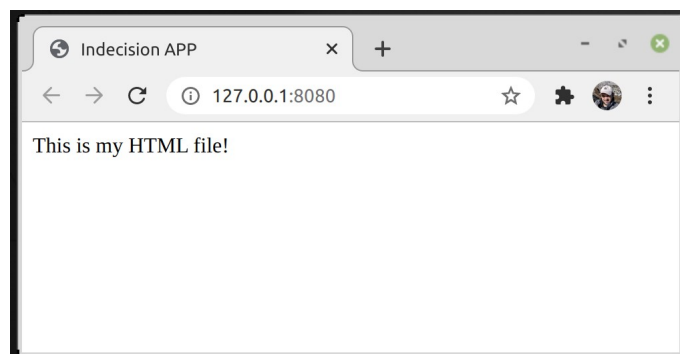

Npm init [CLI=> **npm init -y**]

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

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

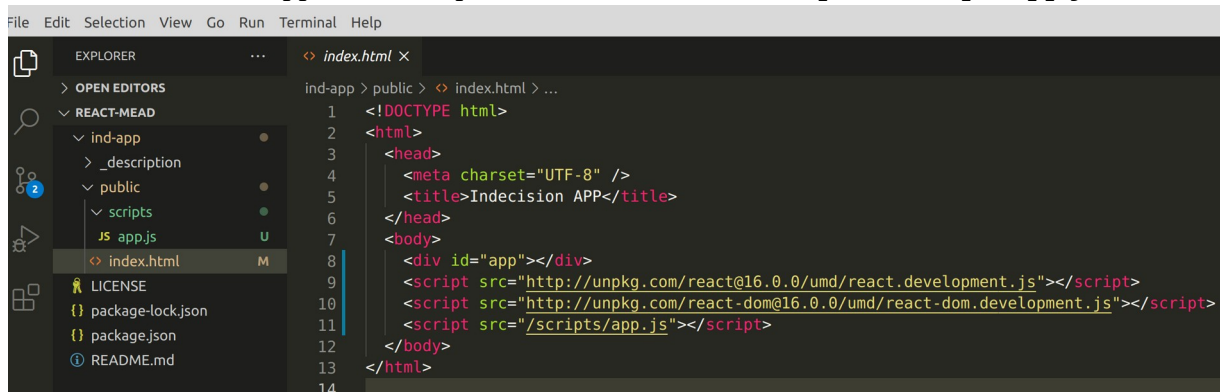Start a server [CLI=> **live-server**]

3.2 Create a React APP (scripts) - 008

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

```
File  Edit  Selection  View  Go  Run  Terminal  Help

EXPLORER                    ...        <> index.html ×

> OPEN EDITORS                          ind-app > public > <> index.html > ...
∨ REACT-MEAD                             1  <!DOCTYPE html>
   ∨ ind-app                  ●          2  <html>
      > _description                     3    <head>
      ∨ public                ●          4      <meta charset="UTF-8" />
         ∨ scripts            ●          5      <title>Indecision APP</title>
            JS app.js         U          6    </head>
         <> index.html        M          7    <body>
      ♀ LICENSE                          8      <div id="app"></div>
      {} package-lock.json               9      <script src="http://unpkg.com/react@16.0.0/umd/react.development.js"></script>
      {} package.json                   10      <script src="http://unpkg.com/react-dom@16.0.0/umd/react-dom.development.js"></script>
      ⓘ README.md                       11      <script src="/scripts/app.js"></script>
                                        12    </body>
                                        13  </html>
                                        14
```
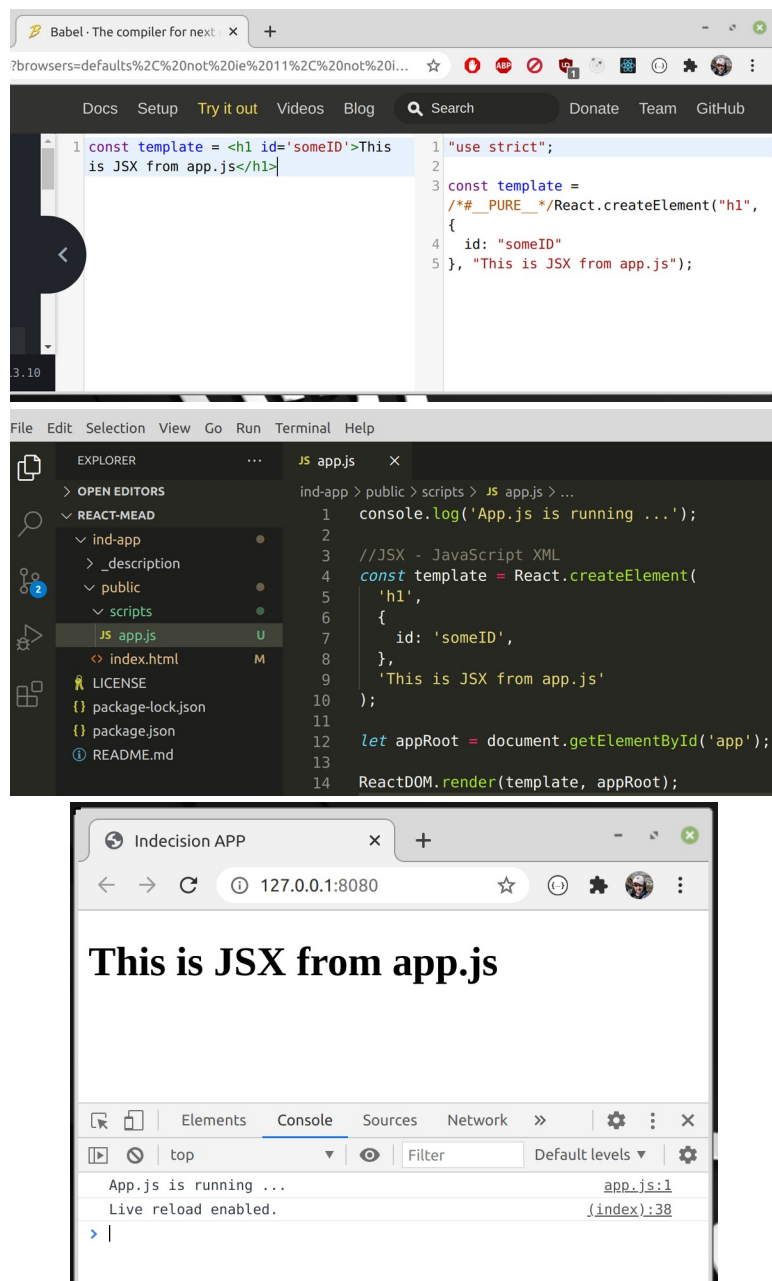
JSX is a JavaScript XML. BABEL is a JavaScript compiler (it converts modern simple ES6 or ES7 to ES5)

```
Babel · The compiler for next ×  +

?browsers=defaults%2C%20not%20ie%2011%2C%20not%20i...  ☆

     Docs   Setup   Try it out   Videos   Blog   🔍 Search        Donate   Team   GitHub

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

```
File  Edit  Selection  View  Go  Run  Terminal  Help

EXPLORER                    ...        JS app.js       ×

> OPEN EDITORS                          ind-app > public > scripts > JS app.js > ...
∨ REACT-MEAD                             1   console.log('App.js is running ...');
   ∨ ind-app                  ●          2
      > _description                     3   //JSX - JavaScript XML
      ∨ public                ●          4   const template = React.createElement(
         ∨ scripts            ●          5     'h1',
            JS app.js         U          6     {
         <> index.html        M          7       id: 'someID',
      ♀ LICENSE                          8     },
      {} package-lock.json               9     'This is JSX from app.js'
      {} package.json                   10   );
      ⓘ README.md                       11
                                        12   let appRoot = document.getElementById('app');
                                        13
                                        14   ReactDOM.render(template, appRoot);
```

```
🌐 Indecision APP          ×  +

←  →  C  ⓘ 127.0.0.1:8080              ☆

This is JSX from app.js

▣ ▢  | Elements  Console  Sources  Network  »  | ⚙ ⋮ ✕
▶ ⊘ | top          ▼ | 👁 Filter      | Default levels ▼ ⚙
   App.js is running ...                            app.js:1
   Live reload enabled.                            (index):38
 > |
```
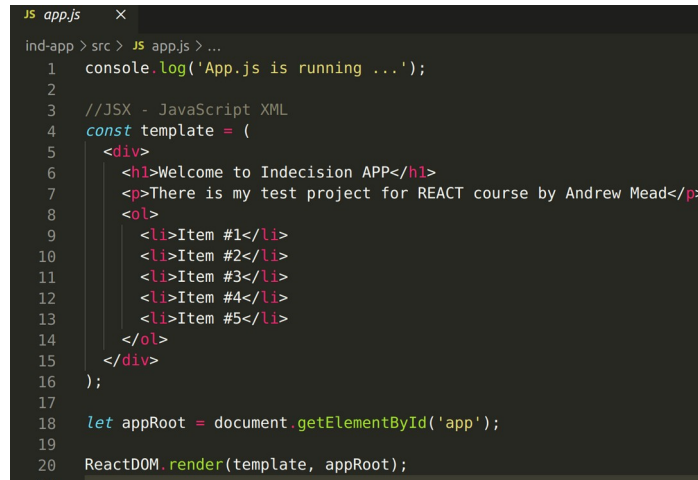
3.3 Using BABEL to compile JSX to ES5 - 009

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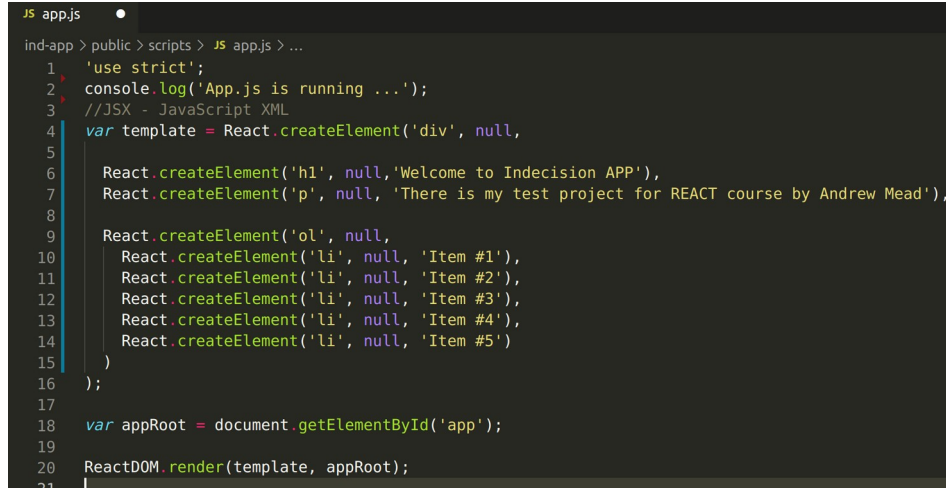
Create a file **src/app.js** with JSX

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

//JSX - JavaScript XML
const template = (
  <div>
    <h1>Welcome to Indecision APP</h1>
    <p>There is my test project for REACT course by Andrew Mead</p>
    <ol>
      <li>Item #1</li>
      <li>Item #2</li>
      <li>Item #3</li>
      <li>Item #4</li>
      <li>Item #5</li>
    </ol>
  </div>
);

let appRoot = document.getElementById('app');

ReactDOM.render(template, appRoot);
```

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
'use strict';
console.log('App.js is running ...');
//JSX - JavaScript XML
var template = React.createElement('div', null,

  React.createElement('h1', null,'Welcome to Indecision APP'),
  React.createElement('p', null, 'There is my test project for REACT course by Andrew Mead'),

  React.createElement('ol', null,
    React.createElement('li', null, 'Item #1'),
    React.createElement('li', null, 'Item #2'),
    React.createElement('li', null, 'Item #3'),
    React.createElement('li', null, 'Item #4'),
    React.createElement('li', null, 'Item #5')
  )
);

var appRoot = document.getElementById('app');

ReactDOM.render(template, appRoot);
```
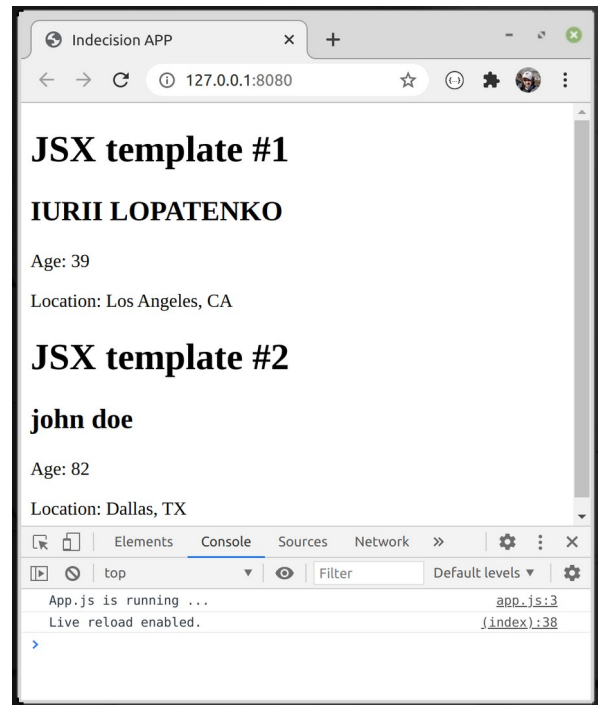
3.4 Exploring JSX - 010

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

//JSX - JavaScript XML

let userName = 'Iurii Lopatenko';
let userAge = 34;
let userLocation = 'Los Angeles, CA';

let user = {
  name: 'John Doe',
  age: 41,
  location: 'Dallas, TX',
};
const templateTwo = (
  <div>
    <h1>JSX template #1</h1>
    <h2>{userName.toUpperCase()}</h2>
    <p>Age: {userAge + 5}</p>
    <p>Location: {userLocation}</p>

    <h1>JSX template #2</h1>
    <h2>{user.name.toLowerCase()}</h2>
    <p>Age: {user.age * 2}</p>
    <p>Location: {user.location}</p>
  </div>
);

let appRoot = document.getElementById('app');

ReactDOM.render(templateTwo, appRoot);
```

3.5 Conditional Rendering in JSX - 012

```js
//JSX - JavaScript XML
//012 - THEORY
let user = {
  name: 'Iurii Lopatenko',
  age: 34,
  location: 'Los Angeles, CA, USA',
};

//Ternary operator
const getLocation = user.location ? (
  <p>Location: {user.location}</p>
) : undefined;

//IF
const checkUserName = (nameToCheck) => {
  if (nameToCheck) {
    return nameToCheck;
  } else {
    return 'ANONYMOUS';
  }
};

const templateTheory = (
  <div>
    <h1>JSX template</h1>
    <h2>{checkUserName(user.name)}</h2>
    {/*Logical compare*/}
    {user.age && user.age >= 18 && <p>Age: {user.age}</p>}
    {getLocation}
  </div>
);
```

```js
//012 PRACTICE
const app = {
  title: 'Indecision App',
  subtitle: 'There is something subtitle for an APP ...',
  options: ['One', 'Two', 'Three'],
};

let templatePractice = (
  <div>
    <h1>{app.title}</h1>
    {app.subtitle && <p>{app.subtitle}</p>}
    {app.options.length > 0 ? 'Here are your options:' : 'No any options'}
    {app.options.map((element, index) => (
      <p key={index}>
        Option #{index + 1} is: {element}
      </p>
    ))}
  </div>
);
let appRoot = document.getElementById('app');
ReactDOM.render(templatePractice, appRoot);
```

3.6 ES6 aside: var vs const and let - 013
3.6.1 Declaring a variable

```js
var nameVar = 'Iurii';
console.log('variable nameVar is: ', nameVar);
var nameVar = 'Mike';
console.log('variable nameVar is: ', nameVar);
nameVar = 'John';
console.log('variable nameVar is: ', nameVar);
```

```
variable nameVar is:  Iurii
variable nameVar is:  Mike
variable nameVar is:  John
```

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;

```js
let nameLet = 'Iurii';
console.log('variable nameLet is: ', nameLet);
let nameLet = 'Mike';
console.log('variable nameLet is: ', nameLet);
```

```
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;

```js
//CONST
console.log('CONST');
const nameConst = 'Iurii';
console.log('variable nameConst is: ', nameConst);
nameConst = 'Frank';
console.log('variable nameConst is: ', nameConst);
const nameConst = 'Mike';
```

```
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

```js
//FUNCTION SCOPING
let functionLet = () => {
   let justName = 'Mike';
   return justName;
};
console.log(functionLet());
//console.log(justName); // justName is not defined

//BLOCK SCOPING
{
   let variable = 'some text';
}
console.log(variable); //variable is not defined
```

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

```js
console.log('hello from es6-arrow-function.js!');

//ES5 function
const square = function (n) {
  return n * n;
};

function squareV2(x) {
  return x * x;
}

let result = square(25);
console.log('Result of ES5 function is: ', result);

//ES6 (ARROW) function
const square6 = (n) => n * n;
let result6 = square6(25);
console.log('Result of ES6 (ARROW) function is: ', result);

//test
let fullName = 'Iurii Lopatenko';
const getFirstName = (fullNAme) => fullName.split(' ')[0];
console.log(getFirstName(fullName));
```

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

```js
console.log('hello from es6-arrow-function-2.js!');

//Arguments object - no longer bound with arrow functions
//
//ES5 function
const add = function (a, b) {
  console.log(arguments);
  console.log(
    `You want to know how much will be ${a} plus ${b} .... I think it should be something like ${
      a + b
    }`
  );
};
add(5, 7);

//ES6 function
let add2 = (a, b) => {
  //console.log(arguments); there will be an error!
  console.log(
    `You want to know how much will be ${a} plus ${b} .... I think it should be something like ${
      a + b
    }`
  );
};
add2(300, 505);

//THAT in ES5
console.log('THAT in ES5');
const userThat = {
  name: 'Iurii',
  cities: ['Moscow', 'Los Angeles'],
  printPlaces: function () {
    console.log(this.name);
    console.log(this.cities);
    const that = this;
    this.cities.forEach(function (city) {
      console.log(that.name + ' has lived in ' + city);
    });
  },
};
userThat.printPlaces();
```

```js
/* //THIS in ES5
console.log('THIS in ES5');
const userThis = {
  name: 'Iurii',
  cities: ['Moscow', 'Los Angeles'],
  printPlaces: function () {
    console.log(this.name);
    console.log(this.cities);
    this.cities.forEach(function (city) {
      console.log(this.name + ' has lived in ' + city); //there will be an error
    });
  },
};
userThis.printPlaces(); */

//THIS in ES6
console.log('THAT in ES6');
const userThisES6 = {
  name: 'Iurii',
  cities: ['Moscow', 'Los Angeles'],
  printPlaces: function () {
    console.log(this.name);
    console.log(this.cities);
    this.cities.forEach((city) => {
      console.log(this.name + ' has lived in ' + city);
    });
  },
};
userThisES6.printPlaces();

/* //THIS in ES6 - all arrow - ERROR!!!
console.log('THAT in ES6 - all arrow');
const userThisES6Arrow = {
  name: 'Iurii',
  cities: ['Moscow', 'Los Angeles'],
  printPlaces: () => {
    console.log(this.name);
    console.log(this.cities);
    this.cities.forEach((city) => {
      console.log(this.name + ' has lived in ' + city);
    });
  },
};
userThisES6Arrow.printPlaces(); */
```

```js
//THIS in ES6 - no arrow - IT WORKS!!!
console.log('THAT in ES6 - no arrow');
const userThisES6NoArrow = {
  name: 'Iurii',
  cities: ['Moscow', 'Los Angeles'],
  printPlaces() {
    //TEST of MAP method
    console.log('Test:');
    const upperCase = this.cities.map(
      (city) => `${this.name.toUpperCase()} has lived in ${city.toUpperCase()}`
    );
    console.log(upperCase);
    //END of TEST

    console.log(this.name);
    console.log(this.cities);
    this.cities.forEach((city) => {
      console.log(this.name + ' has lived in ' + city);
    });
  },
};
userThisES6NoArrow.printPlaces();

//CHALLENGE
console.log('CHALLENGE');
const multiplier = {
  numbers: [5, 4, 7, 45, 46, 97, 125, 10, 5, 456],
  multiplyBy: 456,
  multiply() {
    return this.numbers.map((number) => number * this.multiplyBy);
  },
};
console.log(multiplier.multiply());
```
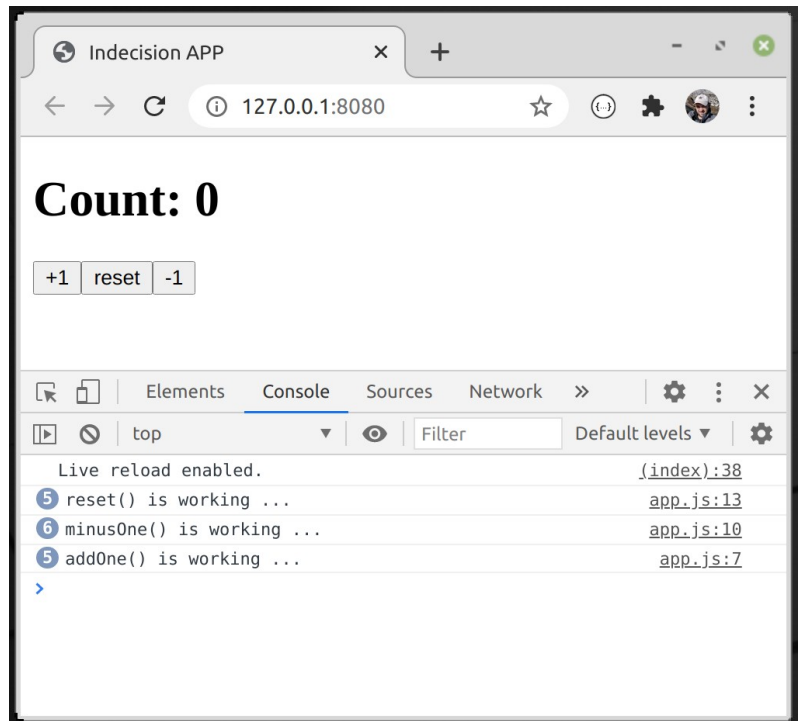
3.9 Events and Attributes - 016

```js
//016 - Events and Attributes
let count = 0;

const addOne = () => {
  console.log('addOne() is working ...');
};
const minusOne = () => {
  console.log('minusOne() is working ...');
};
const reset = () => {
  console.log('reset() is working ...');
};

const template016 = (
  <div>
    <h1>Count: {count}</h1>
    <button onClick={addOne}> +1 </button>
    <button onClick={reset}> reset </button>
    <button onClick={minusOne}> -1 </button>
  </div>
);

const appRoot = document.getElementById('app');
ReactDOM.render(template016, appRoot);
```

Count: 0

+1 reset -1

```
Live reload enabled.                    (index):38
reset() is working ...                  app.js:13
minusOne() is working ...               app.js:10
addOne() is working ...                 app.js:7
```

3.10 Manual data binding - 017

```js
//016 - Events and Attributes
let count = 0;

const addOne = () => {
  console.log('addOne() is working ...');
  count += 1;
  console.log(`Current value of count is ${count}`);
  renderCounter();
};

const minusOne = () => {
  console.log('minusOne() is working ...');
  count -= 1;
  console.log(`Current value of count is ${count}`);
  renderCounter();
};

const reset = () => {
  console.log('reset() is working ...');
  count = 0;
  console.log(`Current value of count is ${count}`);
  renderCounter();
};

const appRoot = document.getElementById('app');

const renderCounter = () => {
  const template016 = (
    <div>
      <h1>Count: {count}</h1>
      <button onClick={addOne}> +1 </button>
      <button onClick={reset}> reset </button>
      <button onClick={minusOne}> -1 </button>
    </div>
  );
  ReactDOM.render(template016, appRoot);
};

renderCounter();
```

Count: -1

+1 reset -1

```
Current value of count is -4            app.js:16
reset() is working ...                  app.js:21
Current value of count is 0             app.js:23
addOne() is working ...                 app.js:7
Current value of count is 1             app.js:9
minusOne() is working ...               app.js:14
Current value of count is 0             app.js:16
minusOne() is working ...               app.js:14
Current value of count is -1            app.js:16
```

9