REACT

#1 - Indecision APP
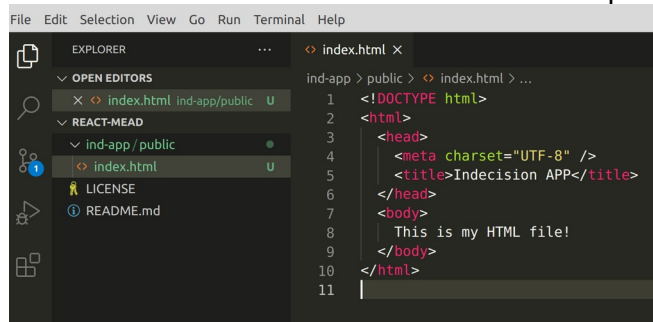
Table of Contents

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

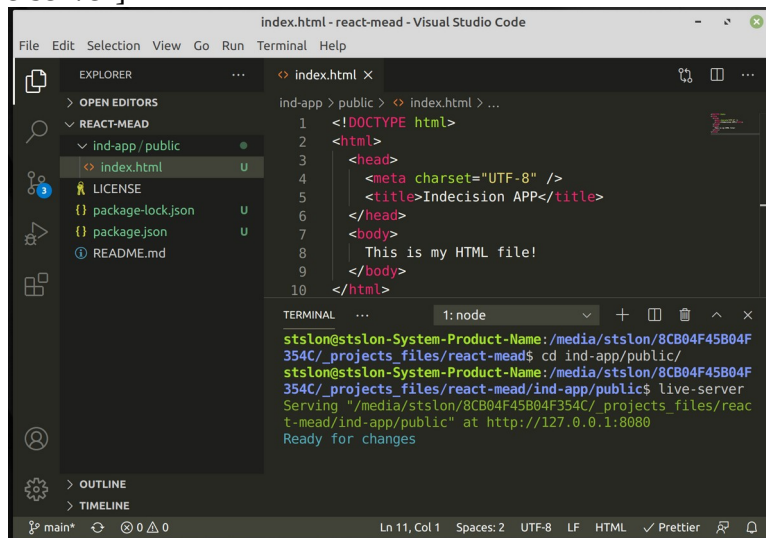Create a new repository on GitHub. Clone it to a local folder. Create a simple HTML file.
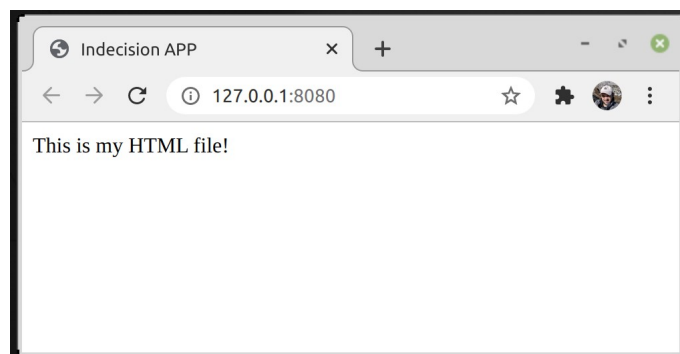
Npm init [CLI=> **npm init -y**]
Install live server [CLI=> **sudo npm install -g live-server**]
Go to a folder ind-app/public [CLI=> **cd ind-app/public**]
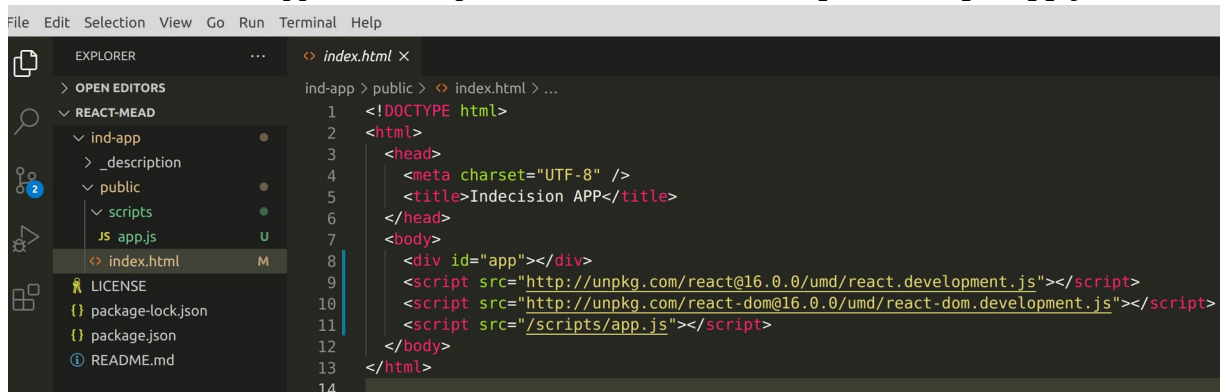Start a server [CLI=> **live-server**]

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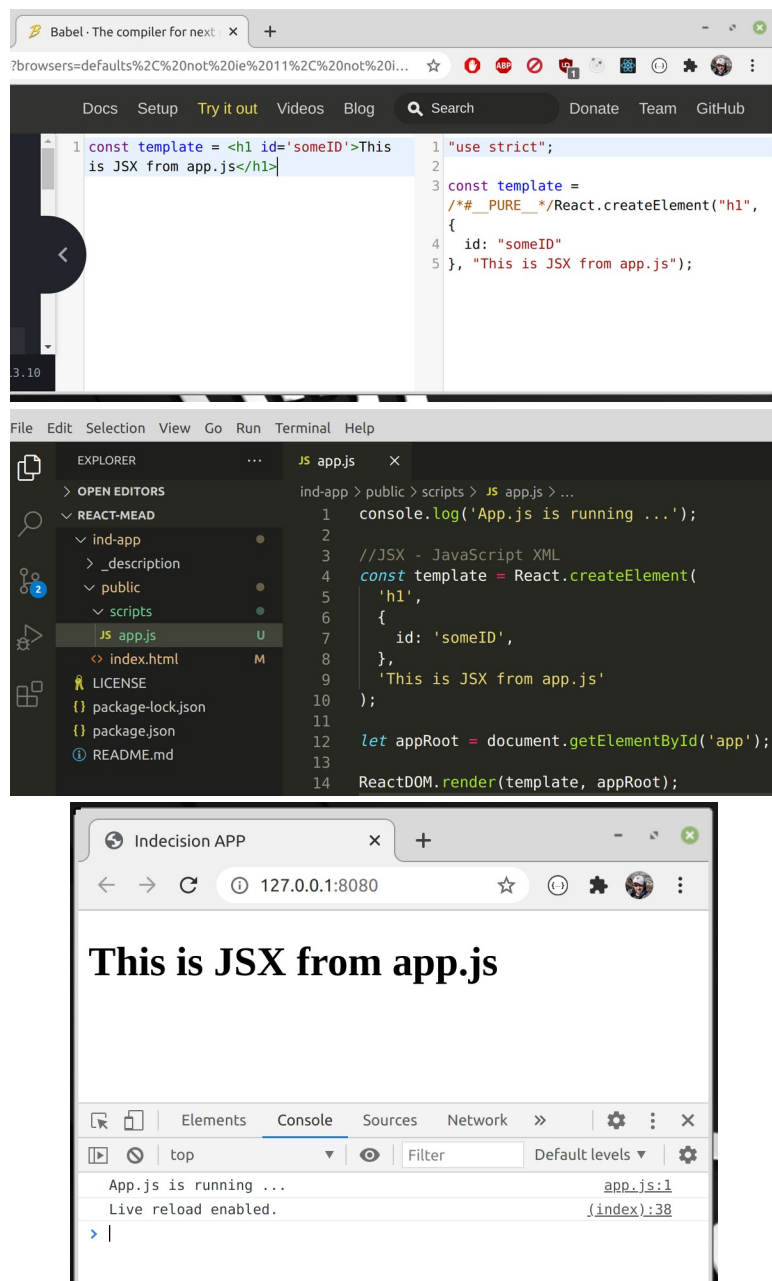
JSX is a JavaScript XML. BABEL is a JavaScript compiler (it converts modern simple ES6 or ES7 to ES5)

```
🅱 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");
   ⟨
  .3.10
```

```
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
 ❯ |
```
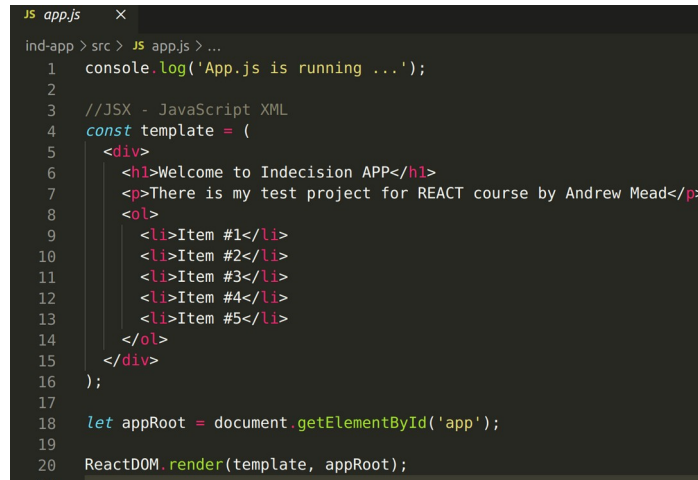
3.3 Using BABEL to compile JSX to ES5 - 009

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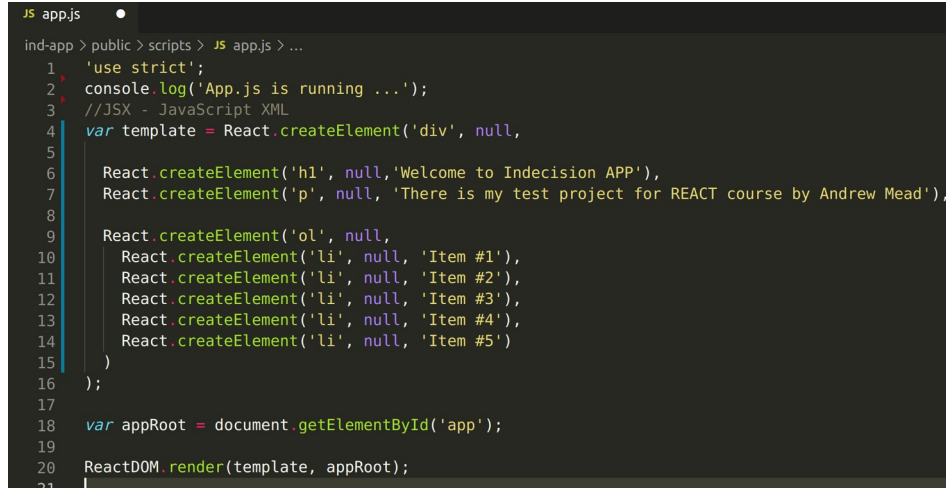
Create a file **src/app.js** with JSX

```
JS app.js    ×
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    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
```
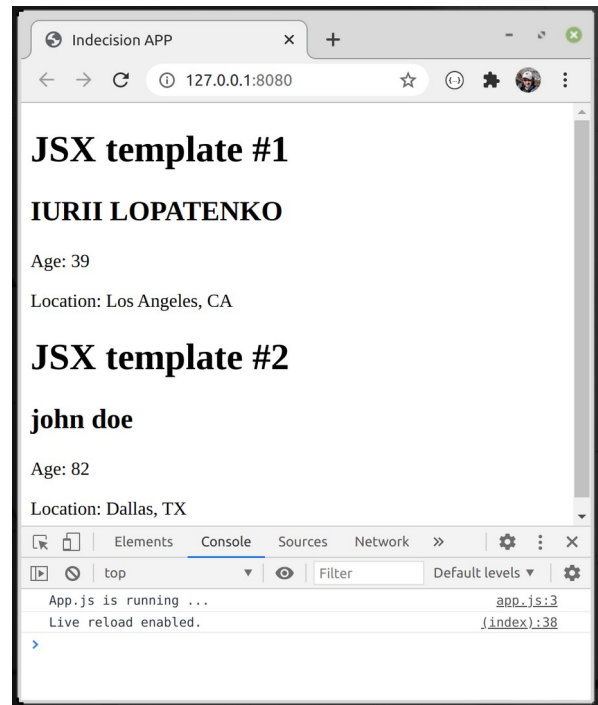
3.4 Exploring JSX - 010

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
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);
```

Indecision APP — 127.0.0.1:8080

JSX template #1

IURII LOPATENKO

Age: 39

Location: Los Angeles, CA

JSX template #2

john doe

Age: 82

Location: Dallas, TX

```
App.js is running ...                    app.js:3
Live reload enabled.                    (index):38
```

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
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
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 ${
  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();
```

```js
  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(); */
```

```js
  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());
```
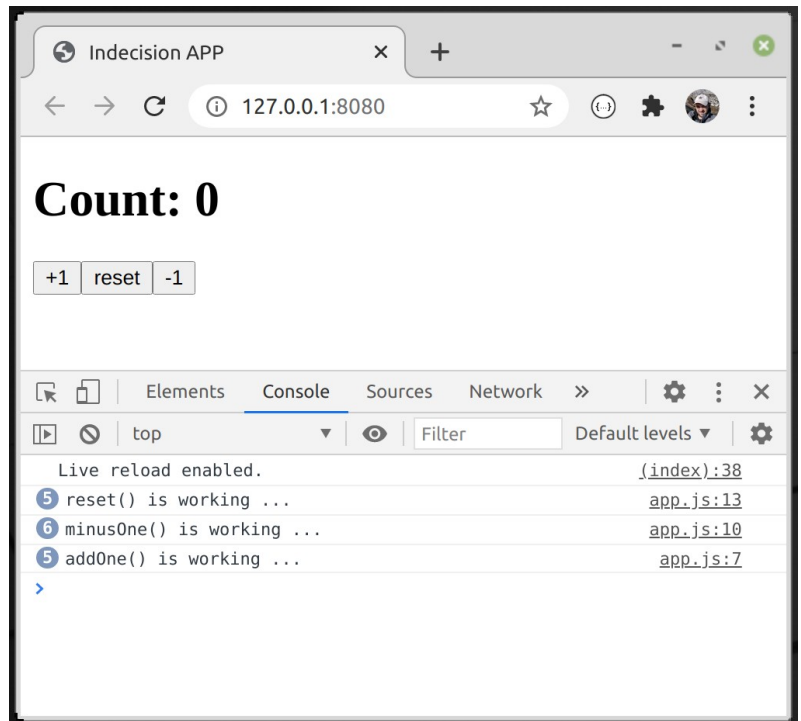
3.9 Events and Attributes - 016

```js
//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);
```
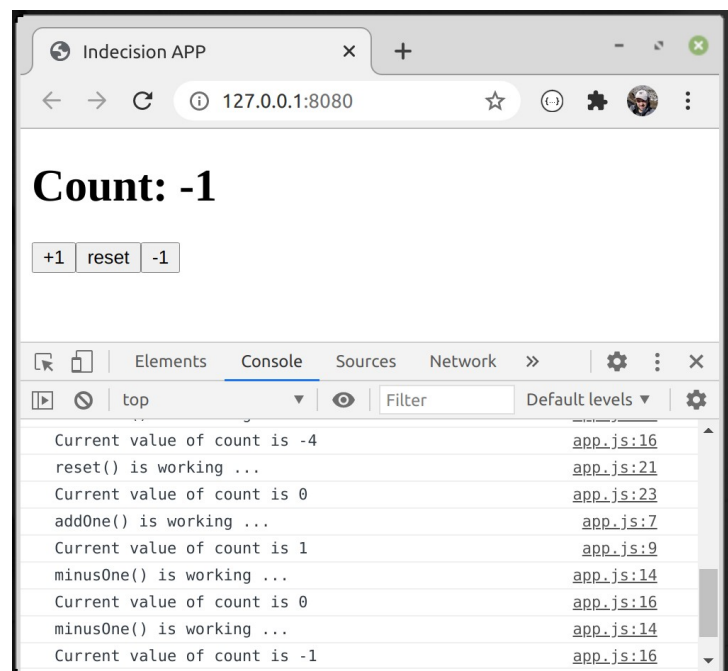
Count: 0

[+1] [reset] [-1]

```
Live reload enabled.                          (index):38
5 reset() is working ...                       app.js:13
6 minusOne() is working ...                     app.js:10
5 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
```
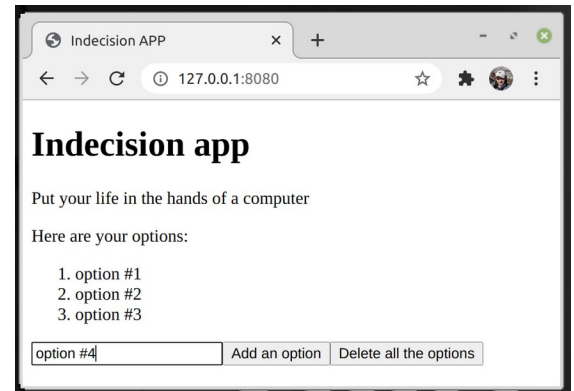
9

3.11 Forms and Inputs - 018

```js
console.log('App.js is running ...');
const app = {
  title: 'Indecision app',
  subtitle: 'Put your life in the hands of a computer',
  options: [],
};
const deleteOptions = () => {
  app.options = [];
  renderApp();
};
const onFormSubmit = (e) => {
  e.preventDefault();
  const option = e.target.elements.option.value;
  if (option) {
    app.options.push(option);
    e.target.elements.option.value = '';
    renderApp();
  }
};
const renderApp = () => {
  const template = (
    <div>
      <h1>{app.title}</h1>
      {app.subtitle && <p>{app.subtitle}</p>}
      <p>{app.options.length > 0 ? 'Here are your options: ' : 'No any options'}</p>
      <ol>{app.options.map((eachOption, index) => (<li key={index}>{eachOption}</li>))}</ol>
      <form onSubmit={onFormSubmit}>
        <input type='text' name='option'></input>
        <button>Add an option</button>
        <button onClick={deleteOptions}>Delete all the options</button>
      </form>
    </div>
  );
  ReactDOM.render(template, appRoot);
};
const appRoot = document.getElementById('app');
renderApp();
```

3.12 Arrays in JSX (Iteration with map()) - 019
# 3.13 A random number. Conditional rendering a button - 020

```js
const app = {
  title: 'Indecision app',
  subtitle: 'Put your life in the hands of a computer',
  options: [],
};
const deleteOptions = () => {
  app.options = [];
  renderApp();
};
const onMakeDecision = () => {
  const randomNum = Math.floor(Math.random() * app.options.length);
  const newOption = app.options[randomNum];
  alert(newOption);
};
const onFormSubmit = (e) => {
  e.preventDefault();
  const option = e.target.elements.option.value;
  if (option) {
    app.options.push(option);
    e.target.elements.option.value = '';
    renderApp();
  }
};
const renderApp = () => {
  const template = (
    <div>
      <h1>{app.title}</h1>
      {app.subtitle && <p>{app.subtitle}</p>}
      <p>{app.options.length > 0 ? 'Here are your options: ' : 'No any options'}</p>
      <ol>{app.options.map((eachOption, index) => (<li key={index}>{eachOption}</li>))}</ol>
      <button disabled={app.options.length === 0} onClick={onMakeDecision}>What should I do?</button>
      <button onClick={deleteOptions}>Delete all the options</button>
      <form onSubmit={onFormSubmit}>
        <input type='text' name='option'></input>
        <button>Add option</button>
      </form>
    </div>
  );
  ReactDOM.render(template, appRoot);
};
const appRoot = document.getElementById('app');
renderApp();
```

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
let isVisible = false;
const changeIsVisible = () => {
  isVisible = !isVisible;
  appRender();
};

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

const appRender = () => {
  const template = (
    <div>
      <h1>Visibiliti Toggle</h1>
      <button onClick={changeIsVisible}>
        {isVisible ? 'Hide details' : 'Show details'}
      </button>
      {isVisible && <p>There are some details ...</p>}
    </div>
  );
  ReactDOM.render(template, appRoot);
};
appRender();
```
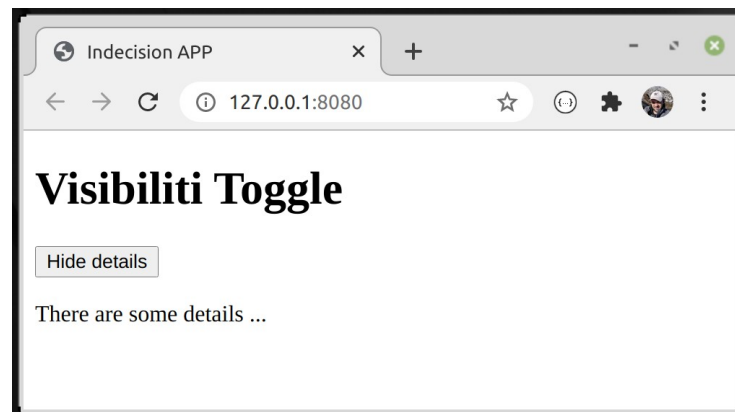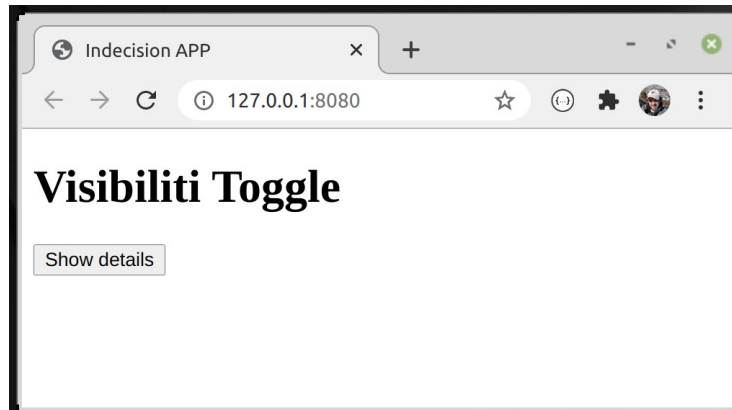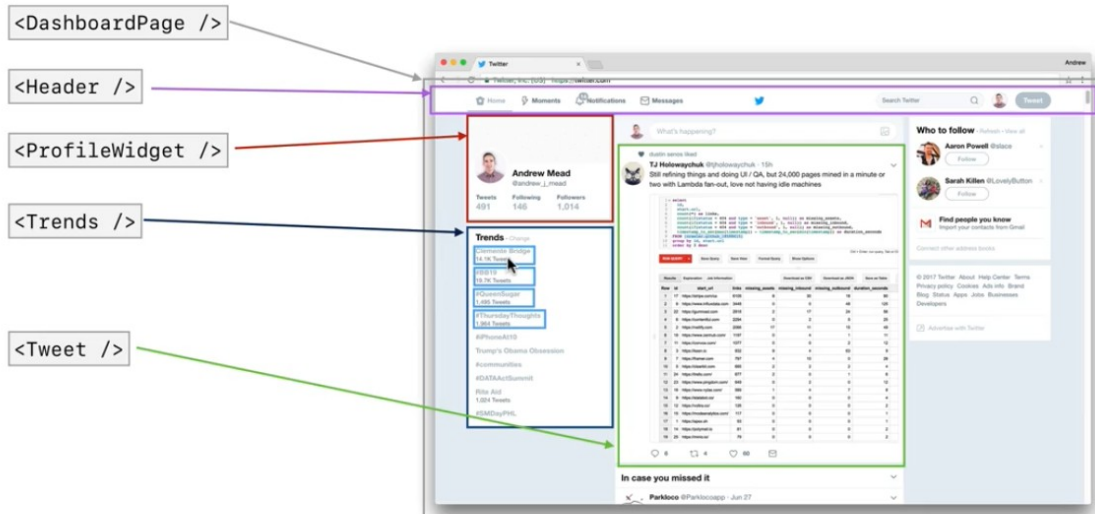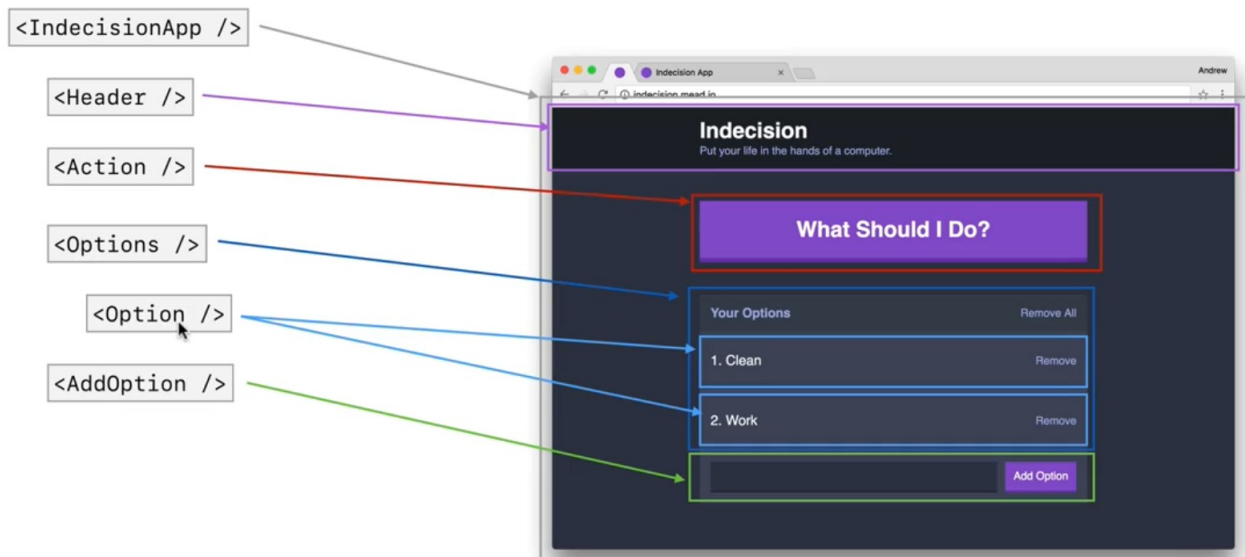
4 React Components
4.1 Intro - 022
4.2 Think in terms of React - 023
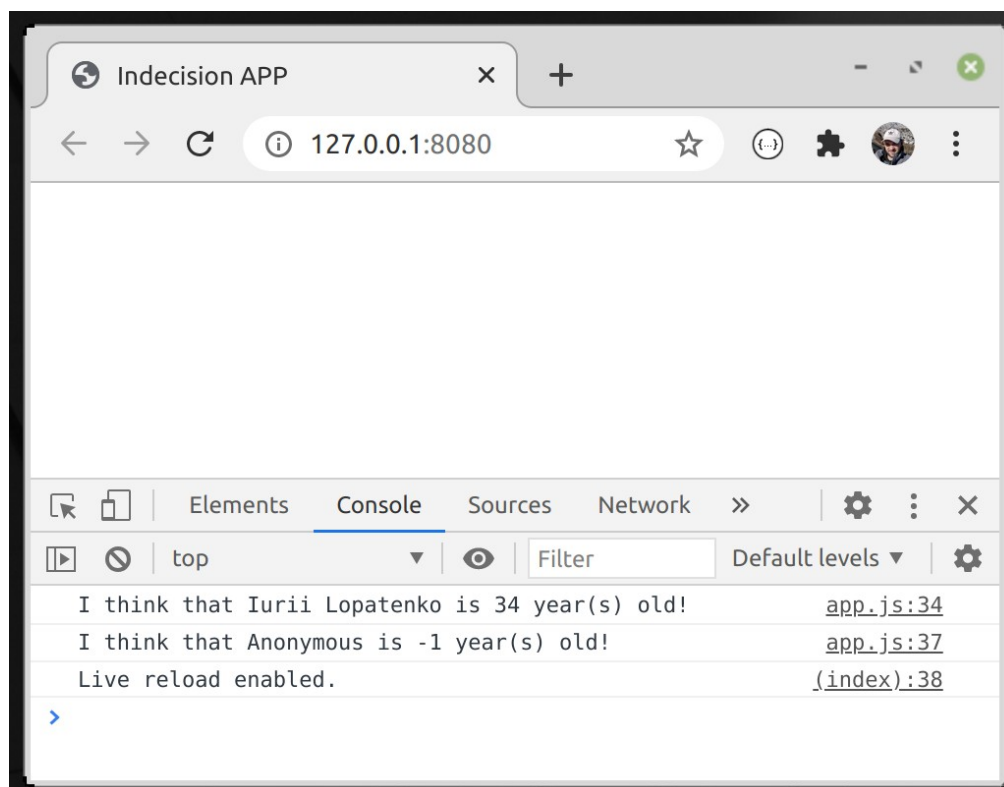
4.3 ES6 Classes (part 1) - 024

Create a simple class (with constructor, default values for properties and few of methods)

```js
class Person {
  constructor(name = 'Anonymous', age = -1) {
    this.name = name;
    this.age = age;
  }
  getGretting() {
    return `Hello, ${this.name}`;
  }
  getDescription() {
    return `I think that ${this.name} is ${this.age} year(s) old!`;
  }
}

const me = new Person('Iurii Lopatenko', 34);
console.log(me.getDescription());

const other = new Person();
console.log(other.getDescription());
```

Indecision APP
127.0.0.1:8080

```
I think that Iurii Lopatenko is 34 year(s) old!        app.js:34
I think that Anonymous is -1 year(s) old!              app.js:37
Live reload enabled.                                   (index):38
```

4.4 ES6 Classes (part 2) - 025

We have learned how to create a sub classes, how to change methods of parent class, how to add new methods

```js
//Create a class Persone
class Person {
  constructor(name = 'Anonymous', age = -1) {
    this.name = name;
    this.age = age;
  }
  getGreeting() {
    return `Hello, ${this.name}`;
  }
  getDescription() {
    return `I think that ${this.name} is ${this.age} year(s) old!`;
  }
}
console.log('PERSONS');
//Create a new instance of class Person
const me = new Person('Iurii Lopatenko', 34);
//LOG result of executing method getDescription() of me
console.log(me.getDescription());

console.log('STUDENTS');
class Student extends Person {
  constructor(name, age, major) {
    //add all the data from parent constructor
    super(name, age);
    //add additional properties
    this.major = major;
  }
  //Add a new method
  hasMajor() {
    return !!this.major;
  }
  //Change a method from parent
  getDescription() {
    let description = super.getDescription();
    if (this.hasMajor()) description += ` Their major is ${this.major}`;
    return description;
  }
}
const meStudent = new Student('Iurii Lopatenko', 34, 'mining science');
console.log(meStudent.getDescription());
```

```js
console.log('TEST TASK: TRAVELER');
class Traveler extends Person {
  constructor(name, age, homeLocation) {
    super(name, age);
    this.homeLocation = homeLocation;
  }

  getGreeting() {
    let greet = super.getGreeting();
    if (!!this.homeLocation) {
      greet += `. You are located at ${this.homeLocation}`;
    }
    return greet;
  }
}

const testTraveler = new Traveler('Iurii Lopatenko', 34, 'Los Angeles');
console.log(testTraveler.getGreeting());
```