ES6 Javascript

1 Lý thuyết:

ES6 (ES2015) là viết tắt của ECMAScript 6, là một phiên bản mới nhất của Javascript theo chuẩn ECMAScript.

ECMAScript là chuẩn quốc tế trong lập trình Javascript, là tập hợp những chuẩn mực, quy tắc trong lập trình Javascript như cách khai báo biến, định nghĩa method, định nghĩa class...

ECMAScript được công bố vào tháng 11 năm 1996, được phát triển bởi **Brendan Eich** và **Netscape**.

1.1 Javascript let

let statement cho phép khai báo biến trong scope {}, phạm vi hoạt động của biến đó chỉ thuộc scope {}.

Giá trị của biến x ngoài scope {} sẽ thay đổi khi ta khai báo lại trong scope {} khi dung **var**.

```
 var x = 10;
// Now x is 10
{
    var x = 2;
    // Now x is 2
}
// Now x is 2
```

Giá trị của biến x ngoài scope {} sẽ không thay đổi khi ta khai báo lại trong scope {} khi dùng **let**.

```
 let x = 10;
// Now x is 10
{
    let x = 2;
    // Now x is 2
}
// Now x is 10
```

Giá trị của biến x sẽ không thay đổi khi ta khai báo lại trong scope {} khi dùng **let**.

1.2 Javascript const

const statement dùng để khai báo một biến với giá trị không thay đổi.

Biến x là một constant trong scope {}, nhưng nó là một biến thông thường ở ngoài scope {}.

```
 let x = 10;
// Now x is 10
{
    const x = 2;
    // Now x is 2
}
// Now x is 10
```

1.3 Default Parameter Value

Trong chuẩn ES6 cho phép gán giá trị mặc định trong tham số truyền vào của một method.

```
function myFunction(x, y = 10) {
    // y is 10 if not passed or undefined
    return x + y;
}
myFunction(5); // will return 15
```

1.4 Array method

Array.find: Trả về giá trị được tìm thấy đầu tiên của mảng trong function kiểm tra.

```
var numbers = [4, 9, 16, 25, 29];
var first = numbers.find(myFunction);
// myFunction is test function
function myFunction(value, index, array) {
    return value > 18;
}
// first is 25
```

Array.findIndex: Trả về index của phần tử được tìm thấy đầu tiên của mảng trong function kiểm tra.

```
var numbers = [4, 9, 16, 25, 29];
var first = numbers.find(myFunction);
// myFunction is test function
function myFunction(value, index, array) {
    return value > 18;
}
// first is 3
```

1.5 New Number Properties

ES6 thêm một vài property cho đối tượng **Number**.

- **EPSILON**

- **MIN_SAFE_INTEGER**
- **MAX_SAFE_INTEGER**

```
// Hằng số điện môi
var x = Number.EPSILON;

// Giá trị nhỏ nhất của kiểu integer
var x = Number.MIN_SAFE_INTEGER;

// Giá trị lớn nhất của kiểu integer
var x = Number.MIN_SAFE_INTEGER;
```

1.6 New Number Method

ES6 cung cấp thêm một vài method mới cho đối tượng **Number**.

- Number.isInteger()
- Number.isSafeInteger()

```
// return true if the argument is an integer
Number.isInteger(10);        // returns true
Number.isInteger(10.5);      // returns false
// return true if the argument is a safe integer
Number.isSafeInteger(10);      // returns true
Number.isSafeInteger(12345678901234567890);      //
returns false
```

1.7 New Global Methods

ES6 cung cấp thêm một vài method global.

- **isFinite**()
- **isNan**()

```
// return false if the argument is infinity or NaN
isFinite(10/0);       // returns false
isFinite(10/1);       // returns true

// return true if the argument is NaN
isNaN("Hello");       // returns true
```

1.8 Arrow Functions

Arrow Function là phương thức định nghĩa một function trong Javascript một cách ngắn gọn.

```
// ES5
```

```
var x = function(x, y) {
     return x * y;
}

// ES6
const x = (x, y) => x * y;
```

Trong Arrow Function không sở hữu đối tượng **this**.

2 Bài Tập:

2.1 BÀI TẬP 1

2.1.1 Thời lượng: 10 phút

2.1.2 Mô tả bài toán:

Cách sử dụng **let** trong ES6.

2.1.3 Các bước thực hiện:

```
let x = 10;
document.writeln("<pre>");
document.writeln("The fist value is" + x);

function test() {
   let x = 5;
   document.writeln("The second value is" + x);
}
test();
document.writeln("The last value is" + x);
document.write("</pre>");
```

2.2 BÀI TẬP 2

2.2.1 Thời lượng: 10 phút

2.2.2 Mô tả bài toán:

Định nghĩa một function có với tham số có giá trị mặc định.

2.2.3 Các bước thực hiện:

```
function myFunction(x, y = 10) {
    return x + y;
}
document.writeln("Result is" + myFunction(5));
```

2.3 BÀI TẬP 3

2.3.1 Thời lượng: 10 phút

2.3.2 Mô tả bài toán:

Dùng method **find** và **findIndex** để tìm giá trị và vị trí của phần tử trong một mảng.

2.3.3 Các bước thực hiện:

```
var numbers = [4, 9, 16, 25, 29];
var first = numbers.find(myFunction);
// myFunction is test function
function firstValueFunction(value, index, array) {
    return value > 18;
}
document.writeln("First number over 18 is " +
firstValueFunction(5));


var firstIndex = numbers.findIndex(myFunction);
// myFunction is test function
function firstIndexFunction(value, index, array) {
    return value > 18;
}
document.writeln("First number over 18 has index " +
firstIndexFunction(5));
```

2.4 BÀI TẬP 4

2.4.1 Thời lượng: 10 phút

2.4.2 Mô tả bài toán:

Định nghĩa một Arrow function.

2.4.3 Các bước thực hiện:

```
const myFunction = (x, y) => x * y;
document.writeln("Result is " + myFunction(5, 5));
```

React Component Basic

1 Lý thuyết:

Trong ReactJS, Component được định nghĩa như class hoặc functions.

Component có những đặc tính nổi bậc như:

- ❏ Cho phép tách UI thành những phần nhỏ không phụ thuộc.
- ❏ Khả năng tái sử dụng (reusable).

Một React Application được tạo nên bởi nhiều Component khác nhau, mỗi Component thể hiện những chức năng UI khác nhau.

Để tạo một Component, ta cần kế thừa từ React.Component.

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Rendering React Component

Ví dụ dưới đây thể hiện cách mà React render một Component lên màn hình.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="React" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

1. Gọi tới `ReactDOM.render()` với element `<Welcome name="React" />`
2. React sẽ khởi tạo `Welcome` element với props {name: "React"}
3. Welcome component sẽ trả về kết quả là `<h1>Hello, React</h1>` element.
4. **React DOM** update element `<h1>Hello, React</h1>` vào **DOM HTML**.

Composing Component

Các Component trong React có thể được tái sử dụng và sử dụng lồng giống như cách sử dụng các element trong **HTML**.

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="React" />
      <Welcome name="JQuery" />
      <Welcome name="Javascript" />
    </div>
  );
}
```

Như ví dụ trên, Component Welcome được gọi 3 lần trong tag div, kết quả React sẽ render 3 element h1 với những nội dung khác nhau dựa vào tham số name được truyền vào.

Extracting Component

Các Component trong React nó độc lập và có thể liên kết với nhau một cách dễ dàng, Chia nhỏ Component thành những Component nhỏ hơn để tăng sự tương tác và đặc tính tái sử dụng của Component React.

```
class UserInfo extends Component {
  render() {
    return (
      <div className="UserInfo">
        <img className="Avatar"
        src={this.props.user.avatarUrl}
        alt={this.props.user.name}
        />
        <div className="UserInfo-name">
          {this.props.user.name}
        </div>
      </div>
    );
  }
}
```

Trong Component **UserInfo**, mỗi user sẽ có một hình avatar và mỗi avatar sẽ có những thuộc tính riêng. Avatar của user sẽ phụ thuộc vào Component UserInfo. Khi ta muốn thay đổi hoặc thêm chức năng cho Avatar (như thay đổi avatar), ta phải chỉnh sửa Component **UserInfo** làm cho Component **UserInfo** sẽ trở nên cồng kềnh và khả năng tái sử dụng lại không cao.

Chia nhỏ component UserInfo để tăng tính tương tác, dễ chỉnh sữa.

```
class Avatar extends Component {
  render() {
    return (
      <img className="Avatar"
        src={this.props.user.avatarUrl}
        alt={this.props.user.name} />
    );
  }
}

class UserInfo extends Component {
  render() {
    return (
      <div className="UserInfo">
        <Avatar user={this.props.user} />
        <div className="UserInfo-name">
          {this.props.user.name}
        </div>
      </div>
    );
  }
}
```

Đối tượng Avatar của user sẽ độc lập với **UserInfo**, mọi thay đổi trong tính năng Avatar sẽ được thực hiện trên Component **Avatar**.

Props

Props là một thuộc tính read-only của Component, được dùng để tương tác dữ liệu giữa các Component.

Bản thân một Component là một thực thể độc lập, ta không thể truy cập trực tiếp vào những thuộc tính hoặc method của một Component từ một Component khác. Các Component sẽ tương tác với nhau thông qua thuộc tính Props của nó.

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}

class App extends Component {
  render() {
    return (
      <div>
        <Welcome name="React" />
    </div>
    );
  }
}
```

Trong ví dụ, Component **App** sẽ gọi tới Component **Welcome** với giá trị truyền vào ***name=”React”***

2 Bài Tập:

1. BÀI TẬP 1

1.1. Thời lượng: 15 phút

1.2. Mô tả bài toán:

Tạo một Component Welcome sử dụng chuẩn ES6.

1.3. Các bước thực hiện:

Chỉnh file ***src/index.js*** như sau:

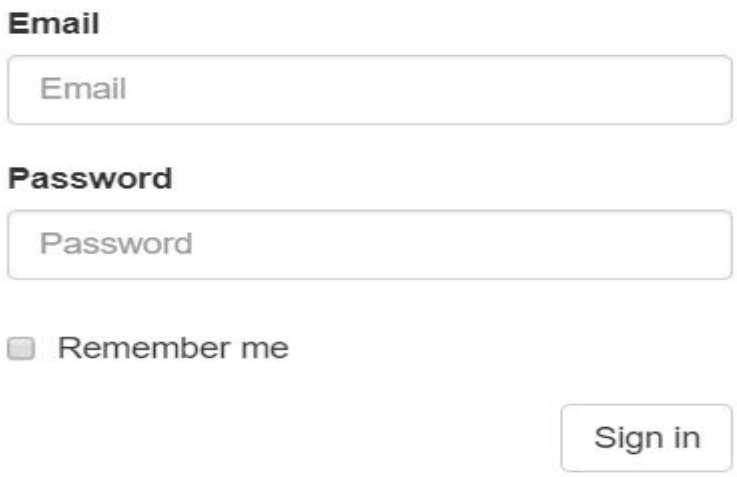
```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import registerServiceWorker from
'./registerServiceWorker';
const Welcome = (props) =>{  return <h1>Hello,
{props.name}</h1>;}
const element = <Welcome name="Sara" />;
ReactDOM.render( element,
document.getElementById('root'));
registerServiceWorker();
```

2. BÀI TẬP 2

1.1. Thời lượng: 20 phút

1.2. Mô tả bài toán:

Tạo một Component LoginForm, hiển thị form login vào một hệ thống như hình dưới



Lưu ý: Component LoginForm sẽ sử dụng react-bootstrap (https://react-bootstrap.github.io/components/forms/)

State and Lifecycle

1 Lý thuyết:

State

State là một thuộc tính đặc biệt của component, lưu trữ những thuộc tính dữ liệu của Component, xác định trạng thái của chính Component đó.

Khi một giá trị thuộc tính của State thay đổi, React sẽ tự động render lại trạng thái (Nội dung) của Component đó. State của component được cập nhật thông qua method ***setState.***

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

Ví dụ trên, Component **Clock** có thuộc tính state được gán bằng đối tượng {date: new Date()}

Lifecycle: Vòng đời của Component

Vòng đời của Component là khoản thời gian tồn tại hay hoạt động của Component đó, nó bắt đầu từ lúc đối tượng Component được render lên màn hình cho tới khi nó được remove DOM của HTML.

Component cung cấp một số method (event) tương tác với những trạng thái khác nhau trong Lifecycle của nó.

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
  componentDidMount() {
  }

  componentWillUnmount() {
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

- **componentDidMount**
  Sự kiện diễn ra khi Component được render lên màn hình
- **componentWillUnmont**
  Sự kiện diễn ra khi kết thúc lifecycle của Component
- **componentDidUpdate**
  Sự kiện diễn ra khi Component được cập nhật và render nội dung.

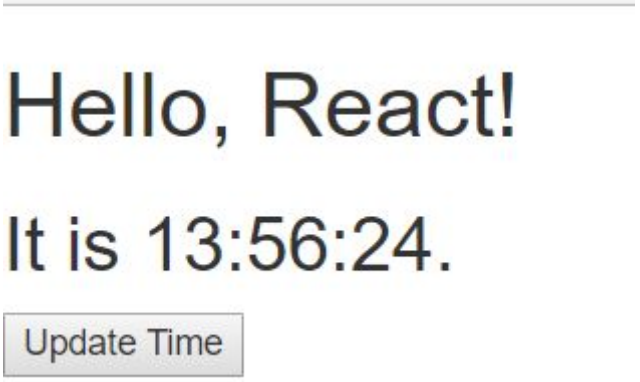
2 Bài Tập:

2.1 BÀI TẬP 1

2.1.1 Thời lượng: 15 phút

2.1.2. Mô tả bài toán:

Định nghĩa và thay đổi trạng thái thuộc tính trong Component.



2.1.3. Các bước thực hiện:

Tạo Component **Clock**

```
import React from 'react';
class Clock extends React.Component {
    constructor(props) {
        super(props);
        //Khởi tạo một trạng thái thuộc tính date
        this.state = {date: new Date()};
    }
    onClick() {
        //Thay đổi trạng thái của thuộc tính date
        this.setState({date: new Date()});
    }
    render() {
        return (
            <div>
                <h1>Hello, {this.props.name}!</h1>
                <h2>It is
{this.state.date.toLocaleTimeString()}.</h2>
                <button
onClick={this.onClick.bind(this)}>Update Time</button>
            </div>
        );
    }
}
export default Clock;
```

Trong file ***src/App.js***

```
import React from 'react';
import Clock from './Components/Clock';
import './App.css';
class App extends React.Component {
  render() {
    return (
      <div>
        <Clock name="React" />
      </div>
    );
  }
}
export default App;
```

2.2 BÀI TẬP 2

2.2.1 Thời lượng: 20 phút

2.2.3 Mô tả bài toán:

Kiểm tra trạng thái thay đổi thuộc tính state của Component trong vòng đời của nó.

2.3.3. Các bước thực hiện:

Trong Component Clock thêm 2 sự kiên ***componentDidMount*** và ***componentDidUpdate*** như sau

```
/**...*//
componentDidMount() {
        alert("Component already rendered.");
    }
    componentDidUpdate(props, state) {
        if(state.date != this.state.date) {
            alert("Current date: " +
this.state.date.toLocaleTimeString());
        }
    }
/**......**/
```

2.3 BÀI TẬP 3

2.3.1 Thời lượng: 20 phút

2.3.3 Mô tả bài toán:

Thêm action ***onClick*** cho component **LoginForm** trong bài tập 2 phần **React Component Basic**.

2.3.3. Các bước thực hiện:

Trong Component LoginForm thêm những nội dung sau

```
/**...**/
constructor(props) {
  super(props);
  this.state = {data: {loginname: "", password: ""}}
}
onSubmit() {
  if(this.state.loginname == "" ||
this.state.data.password == "")  {
        alert("Invalid data");
   } else  {
        alert(
               "Login Name: " + this.state.data.loginname
+
               "\nPassword: " + this.state.data.password
        );
   }
}
…
<Col xs={12} md={3} sm={6} lg={3}>
        <Button className="btn-signin" type="button"
onClick={this.onSubmit.bind(this)}>Sign in
        </Button>
</Col>
...
/**...**/
```

References

1. https://www.w3schools.com/js/js_es6.asp