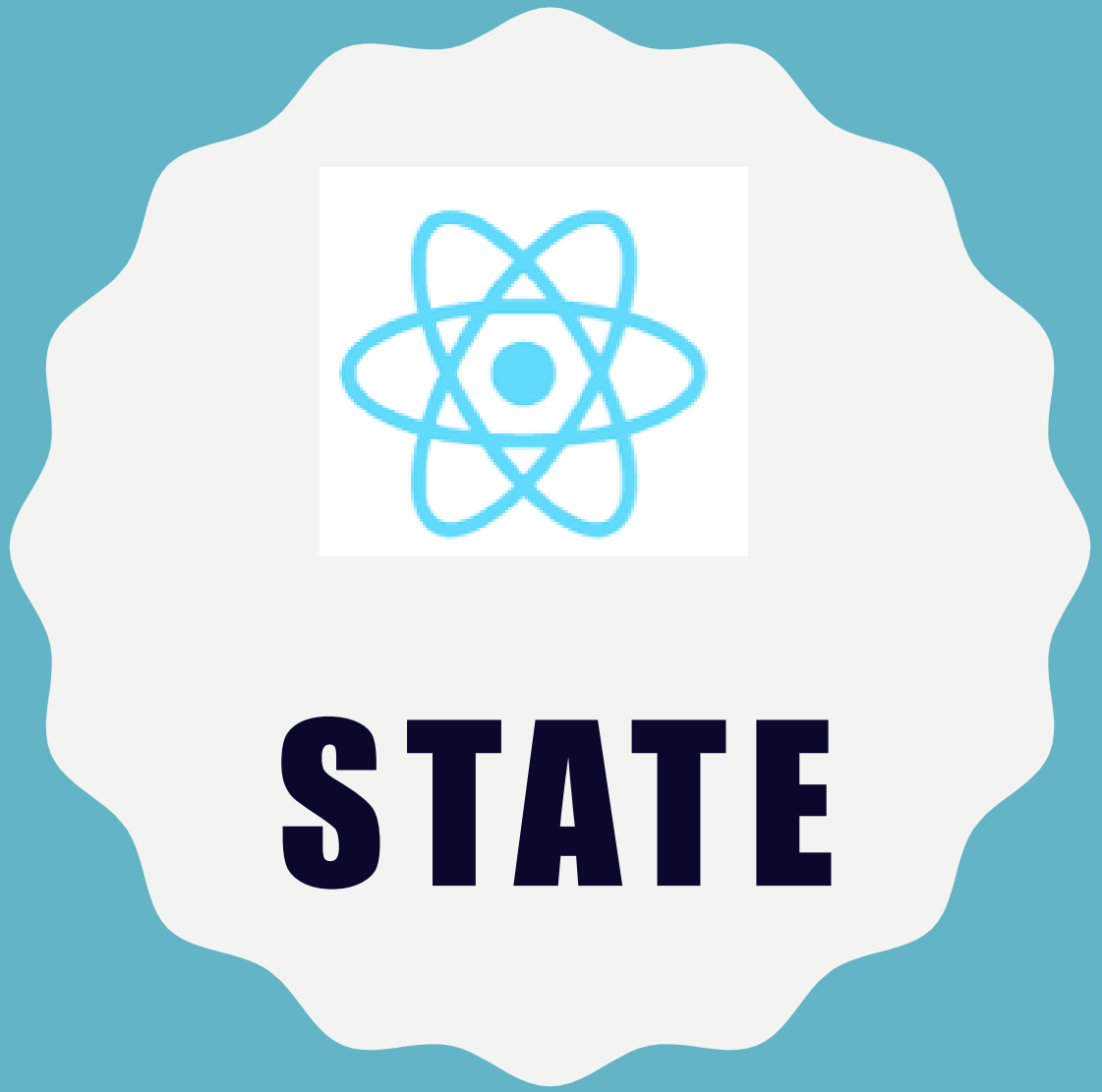

STATE

MONTAGE

1. `constructor()`
2. `getDerivedStateFromProps()`
3. `render()`
4. `componentDidMount()`

- Exemple : constructor()

```
import React from "react";

export default class App extends
React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "Iano"};
    }
    render() {
      return (
        <h1>Coucou {this.state.nom}</h1>
      );
    }
  }
```

MONTAGE

1. `constructor()`
2. `getDerivedStateFromProps()`
3. `render()`
4. `componentDidMount()`

- Exemple : getDerivedStateFromProps()

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "Iano"};
    }
    static getDerivedStateFromProps(props, state) {
      return {nom: props.nommod };
    }
    render() {
      return (
        <h1>Coucou {this.state.nom}</h1>
      );
    }
  }
```

```
/* index.js
ReactDOM.render(
      <App nommod="Iano modifie" />,
      document.getElementById('root')
);
```

MONTAGE

1. `constructor()`
2. `getDerivedStateFromProps()`
3. `render()`
4. `componentDidMount()`

- Exemple : render()

```
import React from "react";

export default class App extends React.Component {
    render() {
      return (
        <h1>Coucou Iano dans render</h1>
      );
    }
  }
```

MONTAGE

1. `constructor()`
2. `getDerivedStateFromProps()`
3. `render()`
4. `componentDidMount()`

- Exemple : componentDidMount()

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "Iano"};
    }
    componentDidMount() {
        setTimeout(() => {
          this.setState({nom: "Iano avec componentDidMount"})
        }, 1000)
    }
    render() {
      return (
        <h1>Coucou {this.state.nom}</h1>
      );
    }
  }
```

UPDATE

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

- Exemple : getDerivedStateFromProps()

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "Iano"};
    }
    static getDerivedStateFromProps(props, state) {
      return {nom: props.nommod };
    }
    changeNom = () => {
      this.setState({nom: "Iano change"});
    }
    render() {
      return (
        <div>
        <h1>Coucou {this.state.nom}</h1>
        <button type="button" onClick={this.changeNom}>Changer nom</button>
        </div>
      );
    }
  }
```

```
/* index.js
ReactDOM.render(
      <App nommod="Iano modifie" />,
      document.getElementById('root')
);
```

UPDATE

1. `getDerivedStateFromProps()`
2. `shouldComponentUpdate()`
3. `render()`
4. `getSnapshotBeforeUpdate()`
5. `componentDidUpdate()`

- Exemple : `shouldComponentUpdate()`

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "Iano"};
    }
    shouldComponentUpdate() {
      // return false;
      return true;
    }
    changeNom = () => {
      this.setState({nom: "Iano change"});
    }
    render() {
      return (
        <div>
        <h1>Coucou {this.state.nom}</h1>
        <button type="button" onClick={this.changeNom}>Changer nom</button>
        </div>
      );
    }
  }
```

UPDATE

1. `getDerivedStateFromProps()`
2. `shouldComponentUpdate()`
3. `render()`
4. `getSnapshotBeforeUpdate()`
5. `componentDidUpdate()`

- Exemple : `render()`

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "Iano"};
    }
    changeNom = () => {
      this.setState({nom: "Iano change"});
    }
    render() {
      return (
        <div>
        <h1>Coucou {this.state.nom}</h1>
        <button type="button" onClick={this.changeNom}>Changer nom</button>
        </div>
      );
    }
  }
```

UPDATE

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

- Exemple : getSnapshotBeforeUpdate()

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "iano"};
    }
    componentDidMount() {
      setTimeout(() => {
        this.setState({nom: "iano modifie avec componentDidMount"})
      }, 1000)
    }
    getSnapshotBeforeUpdate(prevProps, prevState) {
        document.getElementById("div1").innerHTML =
        "Avant update, Coucou " + prevState.nom;
    }
    componentDidUpdate() {
      document.getElementById("div2").innerHTML =
      "Avec update, Coucou " + this.state.nom;
    }
```

```
    render() {
      return (
        <div>
          <h1>Coucou {this.state.nom}</h1>
          <div id="div1"></div>
          <div id="div2"></div>
        </div>
      );
    }
}
```

UPDATE

1. `getDerivedStateFromProps()`
2. `shouldComponentUpdate()`
3. `render()`
4. `getSnapshotBeforeUpdate()`
5. `componentDidUpdate()`

- Exemple : `componentDidUpdate()`

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {nom: "iano"};
    }
    componentDidMount() {
      setTimeout(() => {
        this.setState({nom: "iano modifie avec componentDidMount"})
      }, 1000)
    }
    componentDidUpdate() {
      document.getElementById("div2").innerHTML =
      "Avec update, Coucou " + this.state.nom;
    }
    render() {
      return (
        <div>
          <h1>Coucou {this.state.nom}</h1>
```

DEMONTAGE

- Exemple : `componentWillUnmount()`

```
export default class App extends React.Component {
    constructor(props) {
      super(props);
      this.state = {show: true};
    }
    delHeader = () => {
      this.setState({show: false});
    }
    render() {
      let myheader;
      if (this.state.show) {
        myheader = <Child />;
      };
      return (
        <div>
        {myheader}
        <button type="button" onClick={this.delHeader}>Supprimer Header</button>
        </div>
      );
    }
  }
```

```
class Child extends React.Component {
  componentWillUnmount() {
    alert("Ce component va detruire.");
  }
  render() {
    return (
      <h1>Coucou!!</h1>
    );
  }
}
```

MÉTHODES DE CYCLE DE VIE

```
componentDidMount() {
    //exécutée après que la sortie du composant a été injectée dans le DOM.
    //ici on va mettre en place le minuteur
}

componentWillUnmount() {
    //ici on va détruire le minuteur dans la méthode de cycle de vie
}
```

MÉTHODES DE CYCLE DE VIE

```
class Horloge extends React.Component {
    constructor(props) {···
    }

    componentDidMount() {
        this.minuteur = setInterval(
          () => this.tic(),
          1000
        );
    }

    componentWillUnmount() {
        clearInterval(this.minuteur);
    }

    tic() {
        this.setState({
          date: new Date()
        });
    }
```

MISE A JOUR DE L'ETAT

Exemple avec plusieurs variables

```
constructor(props) {
    super(props);
    this.state = {
      posts: [],
      comments: []
    };
}
```

Mise a jour indépendamment

```
componentDidMount() {
    fetchPosts().then(response => {
      this.setState({
        posts: response.posts
      });
    });

    fetchComments().then(response => {
      this.setState({
        comments: response.comments
      });
    });
  }
```