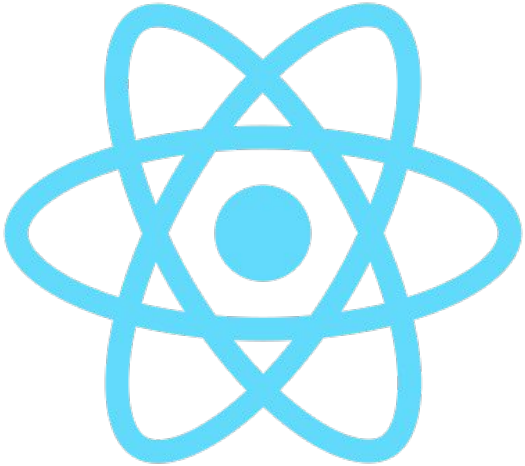


React js

« A declarative, efficient, and flexible JavaScript library for building user interfaces »

GitHub, about section.

Sommaire

- Démonstration
  - Qu’est-ce que React ?
  - Historique de la bibliothèque
  - Les grands principes de React
    - Composants
    - JSX
    - State et Props
    - Le DOM virtuel
  - Étude des performances
  - Démonstration

Qu’est-ce que React ?

➔ Bibliothèque JavaScript pour créer des interfaces utilisateurs

➔ Programmation déclarative

```
// Programmation impérative
if( user.likes() ) {
    if( hasBlue() ) {
        removeBlue();
        addGrey();
    } else {
        removeGrey();
        addBlue();
    }
}
```

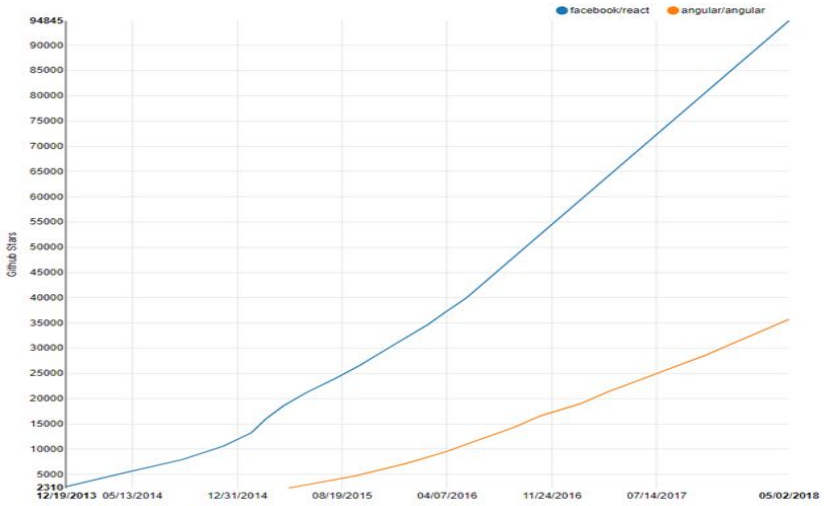
```
// Programmation declarative
if( this.state.liked ) {
    return <blueLike />;
} else {
    return <greyLike />;
}
```

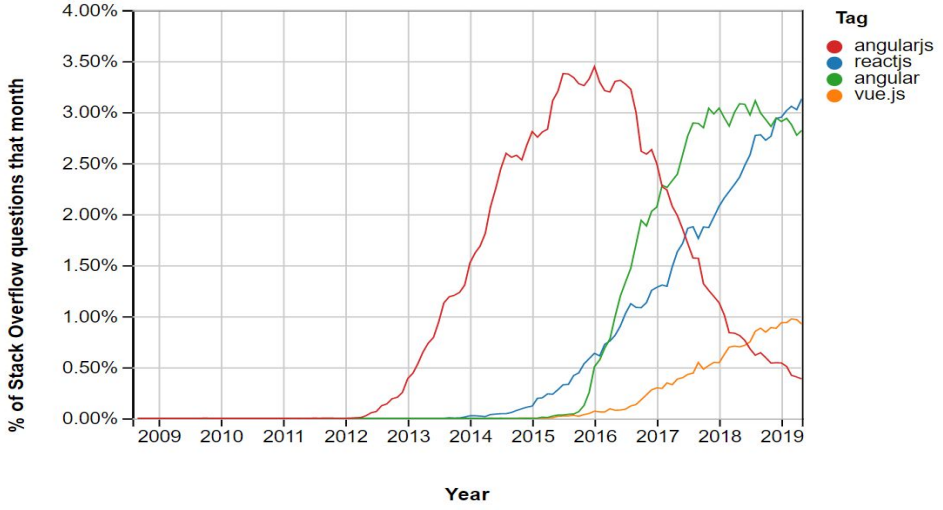
➔ À base de composants

➔ Utilisable dans n’importe quel application et intégrable facilement

Historique de la bibliothèque

2011	Création par Jordan Walke, ingénieur chez Facebook
2013	Première publication comme une bibliothèque open source
2014	Gain de popularité





Historique de la bibliothèque

2015	Première version stable
2021	Facebook, Instagram, Netflix, Airbnb, Cloudflare, Dropbox, Reddit… 166K GitHub stars (181K pour Vue et 72K pour Angular)

Les grands principes de React : Composants

- Morceaux indépendants et réutilisables de code
- Facilite le développement de l'interface utilisateur
- Fonctions ou classes ES6

```
export default function MyComponent() {
   return (
       <p>Hello World!</p>
   )
}
```

```
class MyComponent extends React.Component
{
   render() {
      return <AnotherComponent />;
   }
}
```

Les grands principes de React : JSX

De l'HTML dans le code JavaScript

RÉSULTAT

Hello World!

ÉDITEUR JSX INTERACTIF ☐ JSX ?

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name,
      "!"
    );
  }
}
ReactDOM.render(
    React.createElement(HelloMessage, { name: "World" }),
    document.getElementById('hello-example')
);
```

ÉDITEUR JSX INTERACTIF ☑ JSX ?

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}!
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="World" />,
  document.getElementById('hello-example')
);
```

Les grands principes de React : State

Permet de stocker des données associées à un composant

```
class MyComponent extends React.Component {
   constructor() {
       this.state = { couleur: 'bleu' };
   }

   render() {
       return (
           <div>
               <p>Je stocke la couleur {this.state.couleur}</p>
               <button onClick={() => this.setState({ couleur: 'rouge' })}>
                   Changer la couleur
               </button>
           </div>
       );
   }
}
```

Props

Permet de transmettre de l'information entre composants de manière unidirectionnelle

```
class ParentComponent extends React.Component {
   render() {
       <ChildComponent couleur="vert" />
   }
}
```

Props

```
class ChildComponent extends React.Component {
    constructor(props) {
        super(props);
        this.state = { couleur: props.couleur };
    }

    render() {
        return (
            <div>
                <p>J'ai reçu la couleur {props.couleur}</p>
                <p>Je stocke la couleur {this.state.couleur}</p>
                <button onClick={() => this.setState({ couleur: 'rouge' })}>
                    Changer la couleur
                </button>
            </div>
        );
    }
}
```

Classes, fonctions et hooks

- Introduction des hooks en 2019
- Fonctions permettant d'utiliser les fonctionnalités avancés d'un composant de type classe dans un composant de type fonction

```
class MyComponent extends React.Component {
 constructor(props) {
  super(props);
  this.state = { couleur: 'bleu' };
 }
 render() {
  return (
   <div>
    <p>Je stocke la couleur {this.state.couleur}</p>
    <button onClick={() => this.setState({ couleur: 'rouge'
})}>
     Changer la couleur
    </button>
  </div>
  );
 }
}
```

```
function MyComponent(props) {
 const [couleur, setCouleur] = useState('bleu');

 return (
   <div>
     <p>Je stocke la couleur
{this.state.couleur}</p>
     <button onClick={() => setCouleur('rouge')}>
       Changer la couleur
     </button>
   </div>
 );
}
```

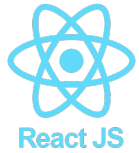
Classes, fonctions et hooks

React JS

Exemple de custom hook

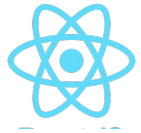
```
function useMyHook(max) {
    const [myNumber, setMyNumber] = useState(0);

    useEffect(() => {
        fetch(`https://example.com/number/${max}`)
        .then(r => setMyNumber(r));
    }, []);

    return myNumber;
}
```

Les grands principes de React : Le DOM virtuel



- DOM : arbre d'éléments HTML (texte), lent à parcourir
- DOM virtuel : arbre de ReactElement (objets), rapide à parcourir
- Un ReactElement est statique et convertissable en élément HTML
- Un ReactComponents est dynamique et convertissable en ReactElement
- Un diff est réalisé entre les ReactComponents et les ReactElement du DOM virtuel afin de décider si une mise à jour est nécessaire

Étude des performances



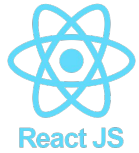
Startup metrics (lighthouse with mobile simulation)

Name	angular-v8.0.1-keyed	react-v16.8.6-keyed	vue-v2.6.2-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,703.7 ± 0.4 (1.24)	2,353.2 ± 0.5 (1.08)	2,177.5 ± 0.3 (1.00)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	52.0 ± 70.5 (3.25)	16.0 ± 0.0 (1.00)	16.0 ± 0.0 (1.00)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	295.5 ± 0.0 (1.40)	260.7 ± 0.0 (1.24)	211.0 ± 0.0 (1.00)
slowdown geometric mean	1.78	1.10	1.00

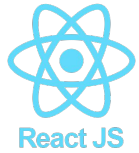
Memory allocation in MBs ± 95% confidence interval

Name	angular-v8.0.1-keyed	react-v16.8.6-keyed	vue-v2.6.2-keyed
ready memory Memory usage after page load.	4.8 ± 0.0 (2.22)	2.2 ± 0.0 (1.03)	2.2 ± 0.1 (1.00)
run memory Memory usage after adding 1000 rows.	9.2 ± 0.0 (1.34)	6.9 ± 0.0 (1.00)	7.0 ± 0.0 (1.03)
update eatch 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	9.5 ± 0.0 (1.28)	8.0 ± 0.0 (1.08)	7.4 ± 0.0 (1.00)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	9.8 ± 0.1 (1.29)	8.8 ± 0.0 (1.15)	7.6 ± 0.0 (1.00)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	6.5 ± 0.0 (1.75)	4.6 ± 0.0 (1.25)	3.7 ± 0.0 (1.00)
slowdown geometric mean	1.54	1.10	1.01

https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/



Démonstration



Merci