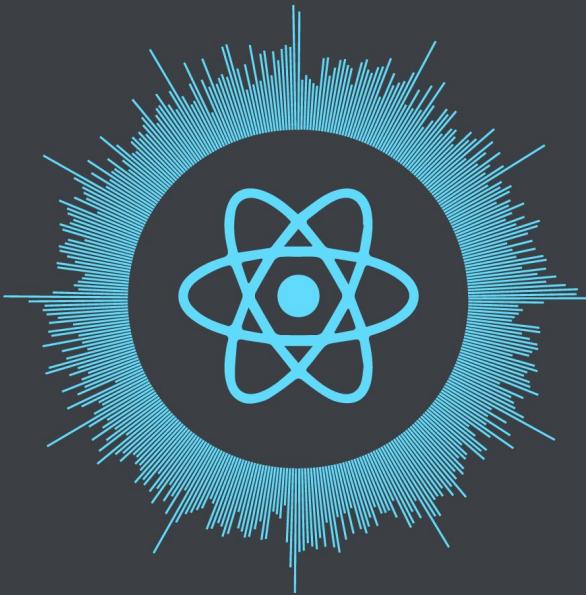


React

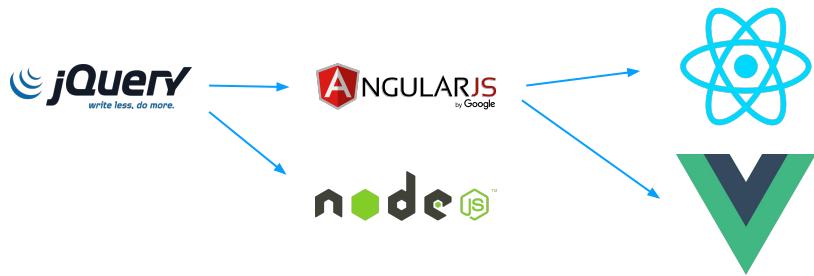
Back to Basics

publicis
sapient



Thomas Simonnet

Développeur depuis 10 ans - Expert JavaScript
Lead Developer & Manager @ Publicis Sapient



@SimonnetTom

publicis
sapient

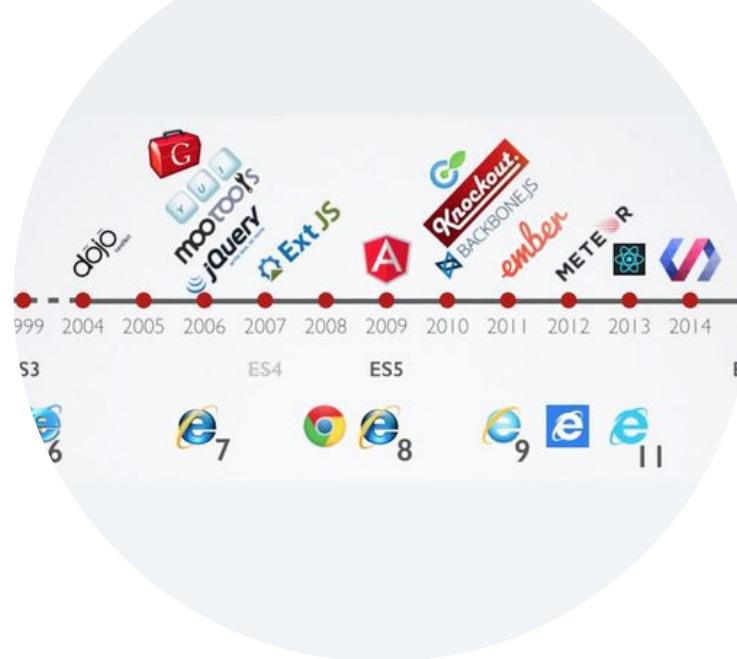


React : Origins

Son histoire, son utilité, sa place dans l'écosystème



Welcome to 2013...



Welcome to 2013...



publicis
sapient

Welcome to 2013...

AngularJS est sorti depuis 4 ans, acteur majeur du Front

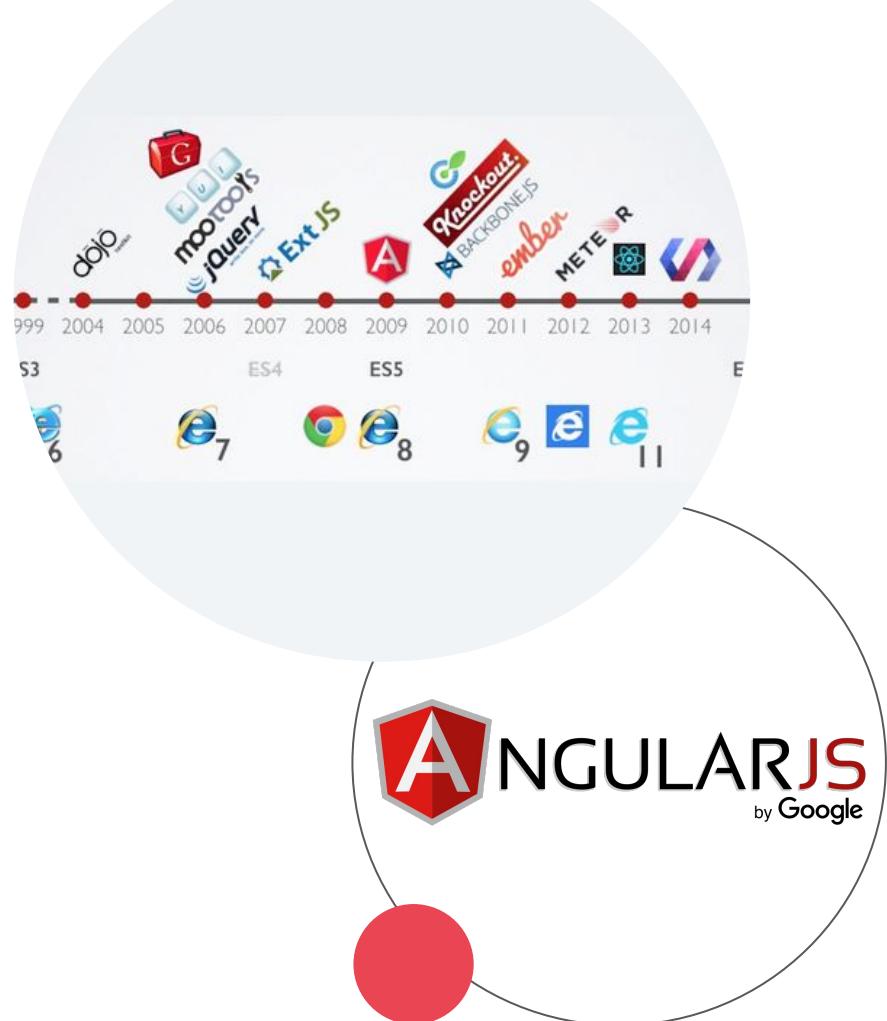
MVC, MVVM, MVW... Modèles en commun. Le but d'une interface est de représenter des modèles et de les modifier

"A component takes the description of the pattern solution and develops real code that can be used in a design." - [Steven Bradley](#)

"Les directives d'AngularJS sont à la fois la grande force et le point difficile à maîtriser quand on aborde le framework" -

[Matthieu Lux](#)

publicis
sapient



Les créateurs

Jordan Walke (Facebook) créateur original, Pete Hunt (Instagram) à la rescoussse pour rendre la bibliothèque indépendante de l'écosystème Facebook

React s'inspire de XHP, une bibliothèque également développée par Facebook, permettant l'inclusion de HTML au sein de PHP

React a été conçu comme étant une bibliothèque et non un framework MVC, comme peuvent l'être ses concurrents. Ainsi, React encourage la création de composants réutilisables, avec en entrée des données, pouvant changer au cours du temps



Les créateurs

Par ailleurs, React n'utilise pas de système de templates et ne fonctionne qu'avec du JavaScript, permettant une encapsulation complète du composant au sein d'une unique classe.

Sortie de React le **29 mai 2013**



JSConfUS - 5/08/2013

Jordan Walke + Tom Occhino - JS Apps at Facebook

Beaucoup de frameworks sur le marché, chacun avec sa façon de structurer les applications JavaScript



Yet another JS framework...

Agility.js • AngularJS • Aria Templates • Backbone.js
Batman.js • Bolt • CanJS • Chaplin + Brunch • Closure
cujo.js • Dart • Derby • dermis • Dijon • Dojo • DUEL
Ember.js • Epitome • Ext.js • Funnyface.js • GWT
Kendo UI • Knockback.js • KnockoutJS • Maria
Marionette • Meteor • Montage • Olives • PlastronJS
PureMVC • rAppid.js • Sammy.js • Serenade.js
SocketStream • soma.js • Spine • Stapes • Thorax • YUI

JSConfUS - 5/08/2013

Jordan Walke + Tom Occhino - JS Apps at Facebook

Beaucoup de frameworks sur le marché, chacun avec sa façon de structurer les applications JavaScript

Nouvelle façon de rendre les données du modèle et surtout de **réagir aux modifications des données**

Les composants sont déclaratifs et écrits grâce à une syntaxe inspirée de l'XML : **Le JSX**

publicis
sapient



Un composant React en 2013

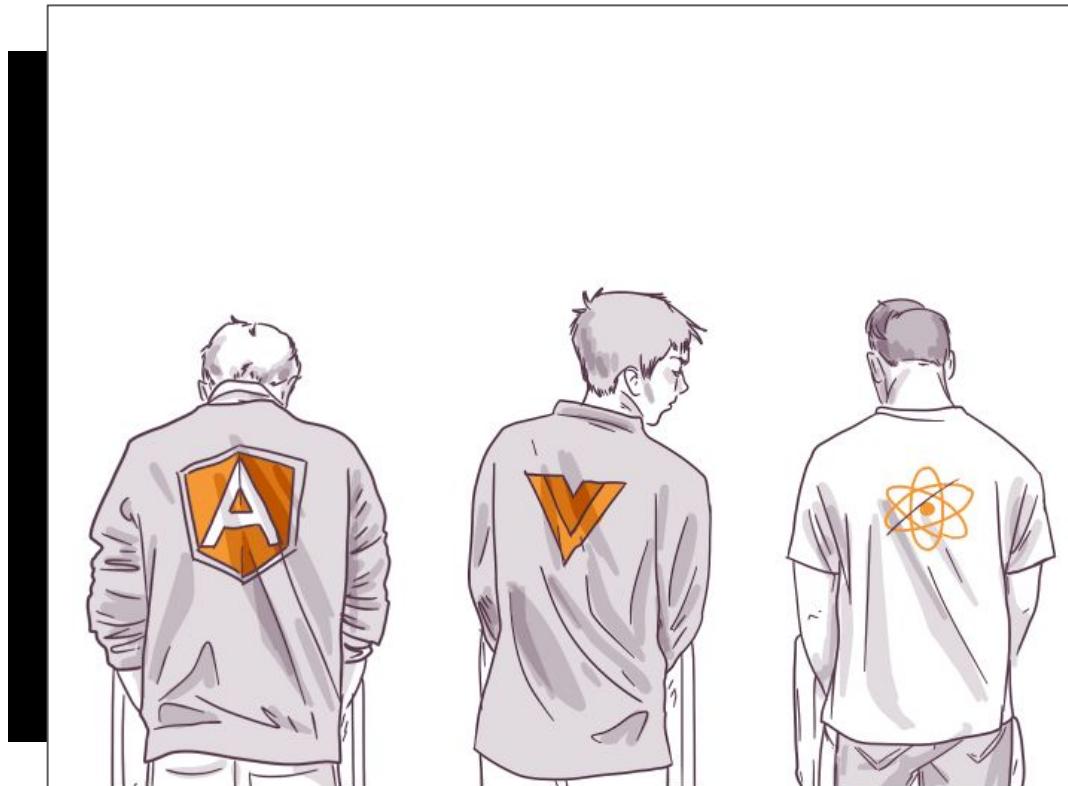
```
1  var ActionButton = React.createClass({
2    render: function() {
3      return (
4        <button class="ActionButton" onClick={this.props.onAction}>
5          <span>{this.props.text}</span>
6        </button>
7      );
8    }
9  });
10 <ActionButton text="Book flight" onAction={someFunc} />;
11
```

Back to 2019...

publicis
sapient



Les frameworks JS en 2019



Popularité théorique

[facebook / react](#) Used by ▾ 2.7m Watch ▾ 6.6k Star 139k Fork 26.4k

[vuejs / vue](#) Used by ▾ 1.1m Watch ▾ 6k Star 152k Fork 22.6k

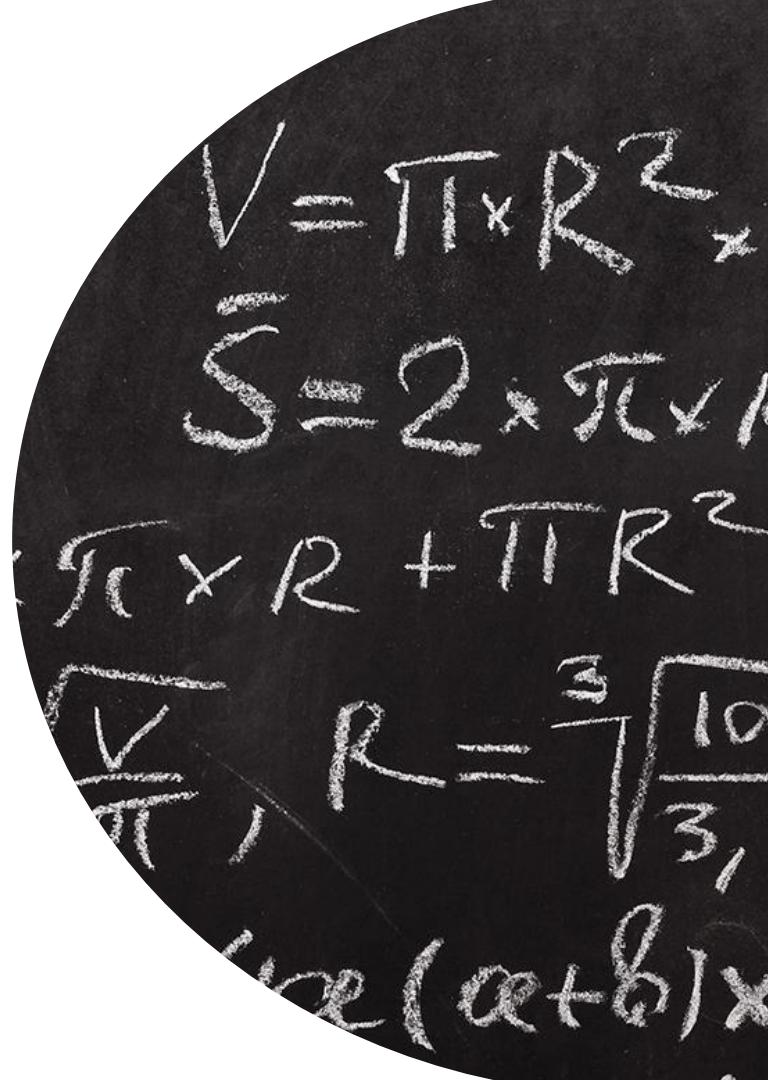
[angular / angular](#) Watch ▾ 3.2k Star 53.8k Fork 14.9k

React sorti en 2013

Vue sorti en 2014

Angular 2+ sorti en 2016

publicis
sapient



Popularité constatée

POCs : React et Vue car faciles d'accès et légers

Petits projets : un peu de tout

Gros projets : React et Angular, le premier pour sa versatilité, le deuxième pour toutes ses fonctionnalités embarquées

En France, beaucoup de projets React et Angular, assez peu de Vue

Transition AngularJS => Angular ratée...

Vue beaucoup plus populaire dans les pays asiatiques (Alibaba.com)



Angular 8

Les plus :

- Framework le plus complet, faible besoin de bibliothèques externes
- Rend l'utilisation de TypeScript obligatoire
- Système de templating
- Utilise des concepts déjà présents dans AngularJS, comme l'injection de dépendances et les modules

The Google logo, consisting of the word "Google" in its signature multi-colored font (blue, red, yellow, green) centered within a large white circle. A small red circle is positioned at the bottom left corner of the slide, pointing towards the Google logo.

Google

Angular 8

Les moins :

- Beaucoup d'outils non retirables, donc grosse taille de bundle
- Probablement la technologie Front du moment avec la courbe d'apprentissage la plus ardue
- Parfois très verbeux



The Google logo, consisting of the word "Google" in its signature multi-colored font (blue, red, yellow, green) located within a large white circle. A small red circle is positioned at the bottom left corner of the slide, pointing towards the Google logo.

Hello World d'Angular 8

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-hello',
  templateUrl: './hello.component.html',
  styleUrls: ['./hello.component.css']
})
export class HelloComponent implements OnInit {

  movies;

  constructor() { }

  ngOnInit() {
    fetch('https://ghibliapi.herokuapp.com/films/?limit=10')
      .then((response) => response.json())
      .then((json) => {
        this.movies = json;
      });
  }
}
```

```
1   <div *ngFor="let movie of movies">
2     <h2>{{movie.title}}</h2>
3     <p>{{movie.description}}</p>
4   </div>
```

Vue

Les plus :

- Framework très performant
- Très simple à prendre à main
- Très bonne documentation
- Système de templating



Vue

Les moins :

- Probablement la plus petite communauté
- Peu de demandes sur le marché
- Un seul homme à la tête du framework



Hello World de Vue 2

```
1  <template>
2  ...
3  </template>
4
5  <script>
6  ...
7  </script>
8
9  <style>
10 ...
11 </style>
12
```

Hello World de Vue 2

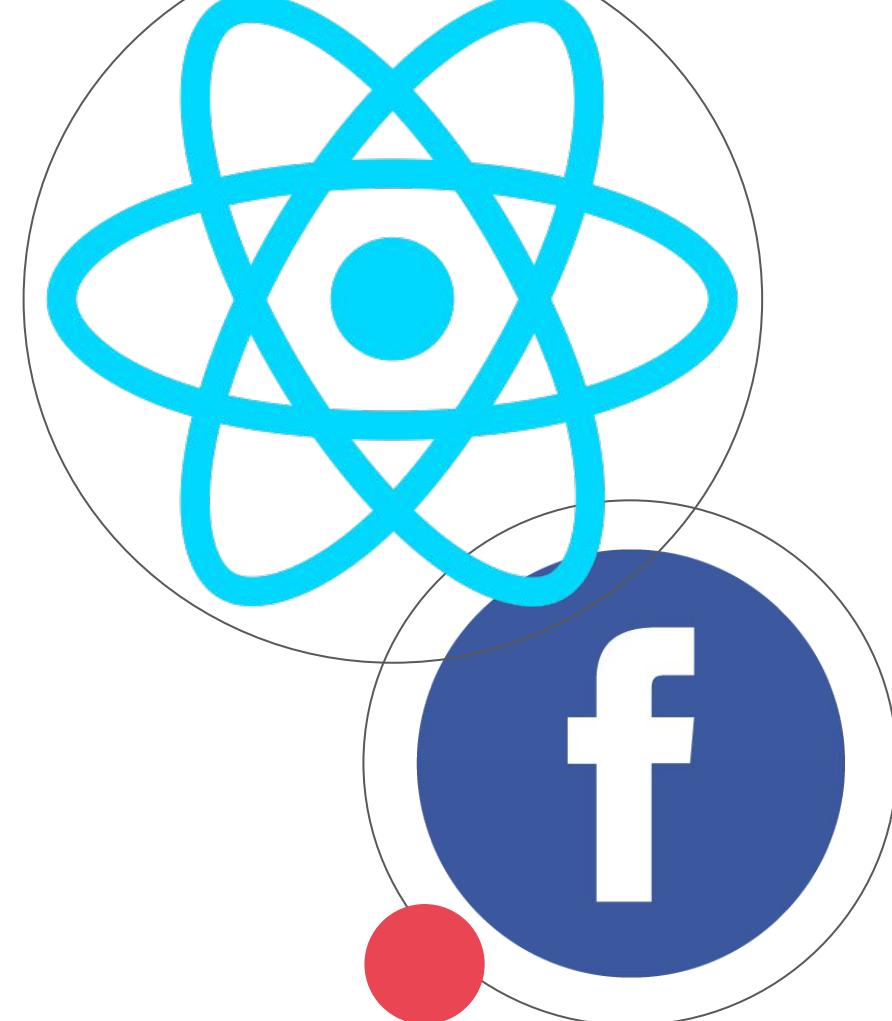
```
1 <template>
2   <div id="employee-table">
3     <p v-if="employees.length < 1" class="empty-table">No employees</p>
4     <table v-else>
5       <thead>
6         <tr>
7           <th>Name</th>
8           <th>Email</th>
9           <th>Actions</th>
10      </tr>
11    </thead>
12    <tbody>
13      <tr :key="employee.id" v-for="employee in employees">
14        <td v-if="editing === employee.id">
15          <input type="text" v-model="employee.name">
16        </td>
17        <td v-else>{{employee.name}}</td>
18        <td v-if="editing === employee.id">
19          <input type="text" v-model="employee.email">
20        </td>
21        <td v-else>{{employee.email}}</td>
22        <td v-if="editing === employee.id">
23          <button @click="editEmployee(employee)">Save</button>
24          <button class="muted-button" @click="editing = null">Cancel</button>
25        </td>
26        <td v-else>
27          <button @click="editMode(employee.id)">Edit</button>
28          <button @click="$emit('delete:employee', employee.id)">Delete</button>
29        </td>
30      </tr>
31    </tbody>
32  </table>
33</div>
34</template>
```

```
36  <script>
37  export default {
38    name: "employee-table",
39    props: {
40      employees: Array
41    },
42    data() {
43      return {
44        editing: null
45      };
46    },
47    methods: {
48      editMode(id) {
49        this.editing = id;
50      },
51      editEmployee(employee) {
52        if (employee.name === "" || employee.email === "") return;
53        this.$emit("edit:employee", employee.id, employee);
54        this.editing = null;
55      }
56    }
57  };
58</script>
59<style scoped>
60 button {
61   margin: 0 0.5rem 0 0;
62 }
63 input {
64   margin: 0;
65 }
66 .empty-table {
67   text-align: center;
68 }
69</style>
70
```

React

Les plus :

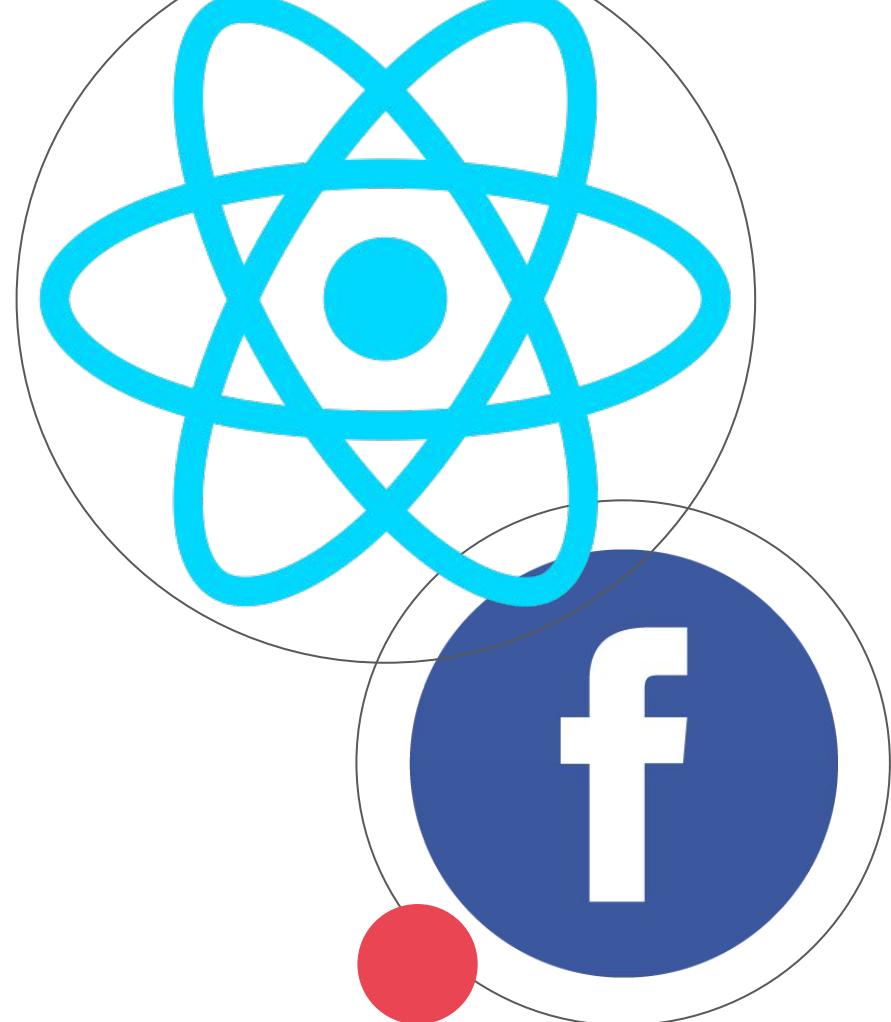
- Le Virtual DOM et la gestion du rendu
- Flot de données unidirectionnel
- Composants très simples à tester
- Une librairie de rendu, pas un framework
- Très grande communauté, et technologie
Front la plus recherchée sur le marché



React

Les moins :

- Librairies externes nécessaires
- Librairie en constante évolution
- Concepts assez différents des deux concurrents
- Beaucoup de libertés, donc beaucoup de place pour les erreurs



Hello World de React 16

```
1 import React, { useState } from 'react';
2
3 import { orderPizzas } from 'services/pizzaHut';
4
5 const PIZZA_CHOICES = ['calzone', 'pepperoni lovers', 'hawaiian'];
6
7 const HelloPizza = () => {
8   |
9   const [pizzas, setPizzas] = useState([])
10
11   const updateOrder = pizza => setPizzas([...pizzas, pizza])
12
13   const launchOrder = () => orderPizzas(pizzas)
14
15   return (
16     <>
17       {PIZZA_CHOICES.map(pizza => (
18         <div onClick={updateOrder}>{pizza}</div>
19       ))}
20       <button onClick={launchOrder}>Order!</button>
21     </>
22   );
23 }
24
25 export default HelloPizza;
```

Démarrage d'un projet Front

Un boilerplate pour chacune des trois technos, très simples à utiliser



```
> npm install -g @angular/cli  
> ng new my-dream-app  
> cd my-dream-app  
> ng serve
```



Démarrage d'un projet Front

Un boilerplate pour chacune des trois technos, très simples à utiliser



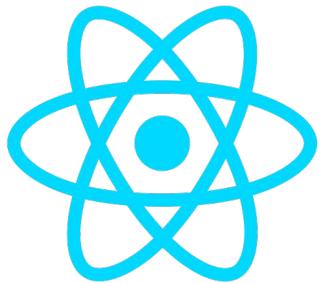
```
npm install -g @vue/cli  
vue create hello-world  
vue serve
```

publicis
sapient



Démarrage d'un projet Front

Un boilerplate pour chacune des trois technos, très simples à utiliser



publicis
sapient

```
npx create-react-app my-app
cd my-app
yarn start
```



Focus sur React

Les bases pour démarrer

publicis
sapient



C'est ici que tout commence

HTML

```
<div id="root"></div>
```

JavaScript

```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```

publicis
sapient



C'est ici que tout commence

HTML

```
1 <div id="root"></div>
```

CSS

JS (Babel)

```
1 const element = <h1>Hello, world!</h1>;
2 ReactDOM.render(element,
  document.getElementById('root'));
```

Hello, world

Le JSX, une syntaxe intuitive

- Extension syntaxique de JavaScript
- Très proche du HTML rendu
- Produit des éléments React
- Non compréhensible par le navigateur
 - Transcodage obligatoire du JSX en JS

```
const element = (
  <h1 className="greeting">
    Bonjour, monde !
  </h1>
);
```

=>

```
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Bonjour, monde !'
);
```

1 Composant === 1 Fonction

```
7  const HelloPizza = () => {
8    |
9    const [pizzas, setPizzas] = useState([])
10   |
11   const updateOrder = pizza => setPizzas([...pizzas, pizza])
12   |
13   const launchOrder = () => orderPizzas(pizzas)
14   |
15   return (
16     <>
17       {PIZZA_CHOICES.map(pizza => (
18         <div onClick={updateOrder}>{pizza}</div>
19       ))}
20       <button onClick={launchOrder}>Order!</button>
21     </>
22   );
23 }
24 }
```

“Au lieu de séparer artificiellement les technologies en mettant le balisage et la logique dans des fichiers séparés, React sépare les préoccupations via des unités faiblement couplées appelées « composants », qui contiennent les deux.”

1 Composant === 1 Fonction

```
7  const HelloPizza = () => {  
8  |  
9   const [pizzas, setPizzas] = useState([])      Gestion de l'état  
10 |  
11 |  
12 |  
13 |  
14 |  
15 |  
16 |  
17 |  
18 |  
19 |  
20 |  
21 |  
22 |  
23 |  
24 |
```

“Au lieu de séparer artificiellement les technologies en mettant le balisage et la logique dans des fichiers séparés, React sépare les préoccupations via des unités faiblement couplées appelées « composants », qui contiennent les deux.”

1 Composant === 1 Fonction

```
7  const HelloPizza = () => {
8    |
9    const [pizzas, setPizzas] = useState([])      Gestion de l'état
10   |
11   const updateOrder = pizza => setPizzas([..pizzas, pizza])
12   |
13   const launchOrder = () => orderPizzas(pizzas) Logique
14   |
15   return (
16   |
17   |
18   |
19   |
20   |
21   |
22   |
23   |
24   )
```

“Au lieu de séparer artificiellement les technologies en mettant le balisage et la logique dans des fichiers séparés, React sépare les préoccupations via des unités faiblement couplées appelées « composants », qui contiennent les deux.”

1 Composant === 1 Fonction

```
7  const HelloPizza = () => {
8    |
9    const [pizzas, setPizzas] = useState([])  Gestion de l'état
10   const updateOrder = pizza => setPizzas([...pizzas, pizza])
11
12   const launchOrder = () => orderPizzas(pizzas) Logique
13
14   return (
15     <>
16       {PIZZA_CHOICES.map(pizza => (
17         <div onClick={updateOrder}>{pizza}</div>
18       ))}
19       <button onClick={launchOrder}>Order!</button>
20     </>
21   );
22 };
23
24 
```

Balisage pour rendu

“Au lieu de séparer artificiellement les technologies en mettant le balisage et la logique dans des fichiers séparés, React sépare les préoccupations via des unités faiblement couplées appelées « composants », qui contiennent les deux.”

1 Composant === 1 Fonction

```
27 const Welcome = props => {
28   return <h1>Bonjour, {props.name}</h1>;
29 }
30
31 <Welcome name="Sara" />
```

“Tout composant React doit agir comme une fonction pure vis-à-vis de ses props.”

1 Composant === 1 Fonction

```
27 const Welcome = props => {
28   props.name = 'Jean Michel';
29   return <h1>Bonjour, {props.name}</h1>;
30 }
31
32 <Welcome name="Sara" />
```

*"Tout composant React
doit agir comme une
fonction pure vis-à-vis de
ses props."*

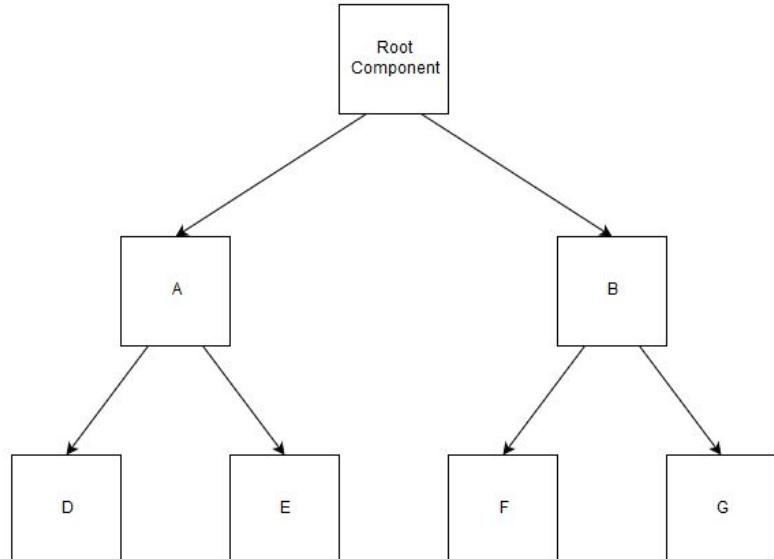
1 Composant === 1 Fonction



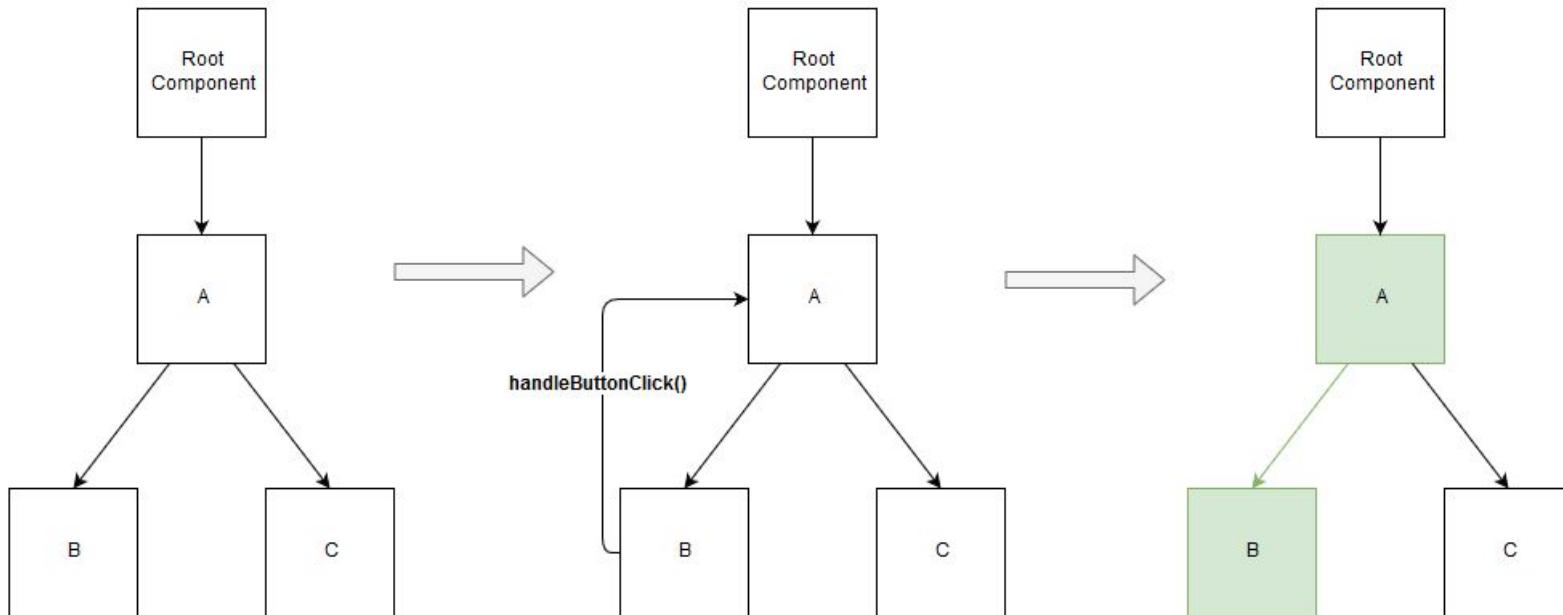
“Tout composant React doit agir comme une fonction pure vis-à-vis de ses props.”

Le flot de données est unidirectionnel

- Les props se diffusent toujours des parents vers les enfants
- Les enfants ne peuvent pas directement modifier leurs parents
- Généralement, les actions de l'UI sont les seules à modifier les états des composants
- Quand les props ou l'état d'un composant est modifié, il est re-rendu, à l'instar de ses enfants

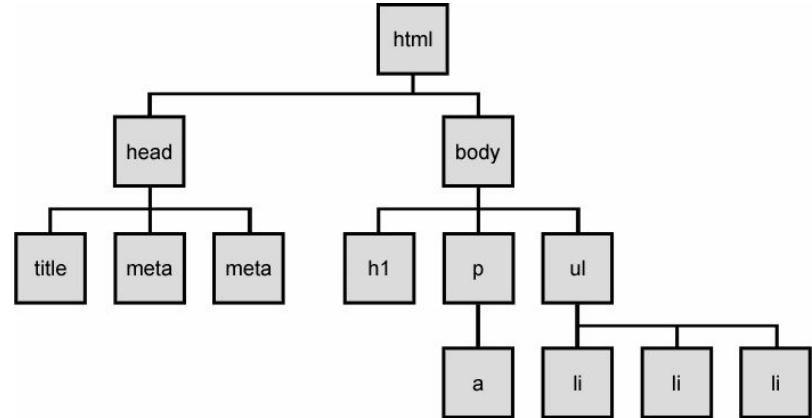


Le flux de données est unidirectionnel



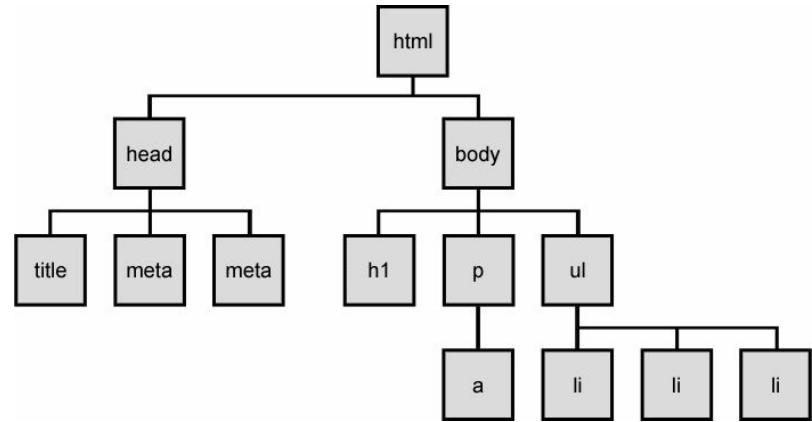
La vraie force de React, son moteur de rendu

- Lors d'un nouveau rendu, React ne met à jour que ce qui est nécessaire
- Combinaison d'un algorithme de réconciliation et du Virtual DOM (Document Object Model)



La vraie force de React, son moteur de rendu

- Lors d'un nouveau rendu, React ne met à jour que ce qui est nécessaire
- Combinaison d'un algorithme de réconciliation et du Virtual DOM (Document Object Model)
- A chaque update d'un composant, tout le Virtual DOM est modifié
- React détecte les différences entre l'ancienne copie du Virtual DOM et la nouvelle
- React ne met à jour que cette différence dans le vrai DOM



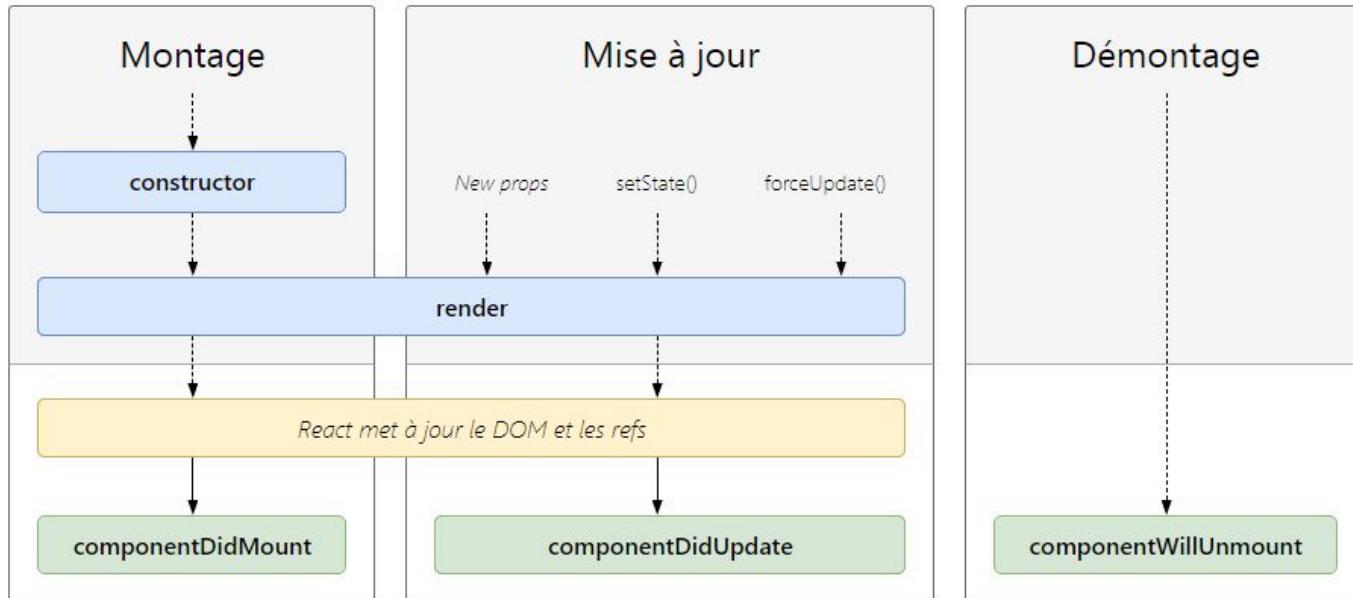
Le lifecycle des composants

```
7  class HelloPizza extends React.Component {
8
9    constructor(props) {
10      super(props);
11      this.state = {pizzas: []}
12    }
13
14   componentDidMount() {}
15
16   componentDidUpdate() {}
17
18   componentWillUnmount() {}
19
20   updateOrder = pizza => {
21     this.setState(state => {pizzas : [...pizzas, pizza]});
22   }
23
24   launchOrder = () => orderPizzas(this.state.pizzas)
25
26   render() {
27     return (
28       <>
29         {PIZZA_CHOICES.map(pizza => (
30           <div onClick={updateOrder}>{pizza}</div>
31         )));
32         <button onClick={launchOrder}>Order!</button>
33       </>
34     );
35   }
36 }
```

Le lifecycle des composants

"Phase de Render"
Méthodes pures, sans effets secondaires.
Peuvent être interrompues, annulées ou redémarrées par React.

"Phase de Commit"
Peuvent opérer sur le DOM, engendrer des effets secondaires, programmer des mise à jour.



Les Hooks, une autre façon de se brancher au lifecycle

- Utilisables uniquement dans un composant sous forme de fonction
- useState, pour gérer l'état d'un composant
- useEffect, pour déclencher des effets après le rendu
- Possibilité de construire ses propres Custom Hooks

```
import React, { useState, useEffect } from 'react';

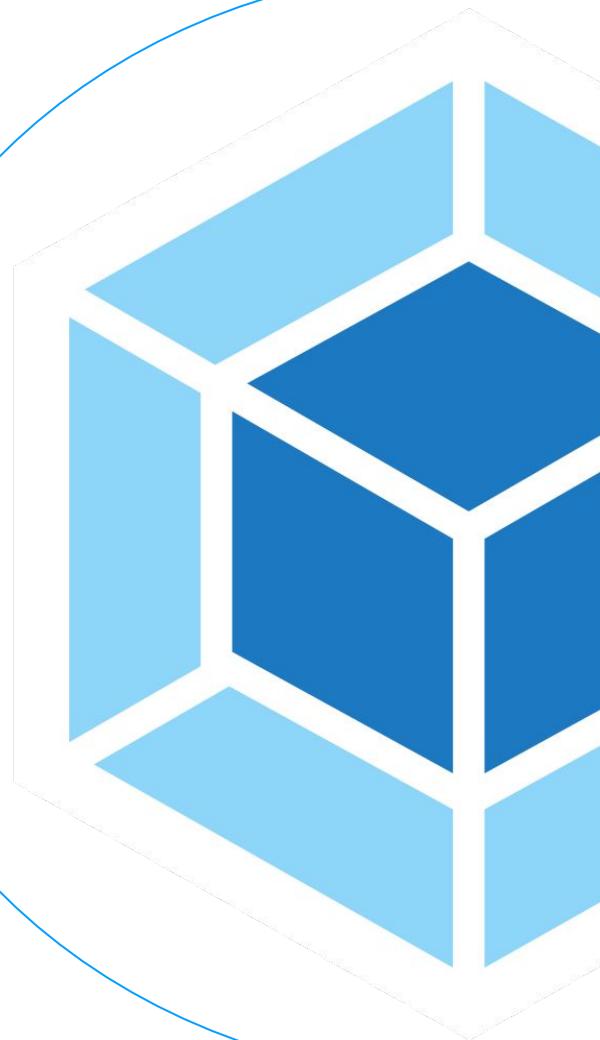
function Example() {
  const [count, setCount] = useState(0);

  // Équivalent à componentDidMount plus componentDidUpdate :
  useEffect(() => {
    // Mettre à jour le titre du document en utilisant l'API du navigateur
    document.title = `Vous avez cliqué ${count} fois`;
  });

  return (
    <div>
      <p>Vous avez cliqué {count} fois</p>
      <button onClick={() => setCount(count + 1)}>
        Cliquez ici
      </button>
    </div>
  );
}
```

Un facteur essentiel : Webpack

- Bundler Web le plus connu et le plus utilisé
- Configuration assez verbeuse et complexe
- Permet d'identifier et de regrouper les ressources importées dans le projet (CSS, images, fonts, etc...)
- Sert également de serveur web en développement



Place à l'exercice !



<https://github.com/keabard/react-basics-molkky>



Des questions ?

publicis
sapient



Biblio le Hobbit

- [Design Patterns, Components, And Frameworks](#)
- [\[JSConfUS 2013\] Tom Occhino and Jordan Walke: JS Apps at Facebook](#)
- [AngularJS : Les directives](#)
- [Web Components VS Frameworks](#)
- [Why did we build React?](#)



Biblio le Hobbit

- [Javascript Framework Comparison: Vue, React and Angular \(2019\)](#)
- [React: The Virtual DOM](#)
- [The difference between Virtual DOM and DOM](#)
- [React Documentation](#)
- [React lifecycle methods diagram](#)
- [Pourquoi devoir compiler nos apps React](#)



Biblio le Hobbit

- [An Introduction To Rendering In React](#)
- [Angular vs. React vs. Vue: A performance comparison](#)

