Breizh C@mp
Mix de technologies

React, une autre manière de penser vos composants graphiques

Mathieu ANCELIN
@TrevorReznik
SERLI

Mathieu ANCELIN

- Développeur @SERLI
- Scala, Java, JS, web & OSS
  - ReactiveCouchbase, Weld-OSGi, etc ...
  - Poitou-Charentes JUG
- Membre de l'expert group MVC 1.0
- @TrevorReznik

Serli

- Société de conseil et d'ingénierie
- Développement, expertise, R&D, formation

- 70 personnes
- Contribution à des projets OSS
- Membre du JCP

DEVOXX the java community conference

JavaOne MAKE THE FUTURE JAVA

Training by SERLI

Java Community Process MEMBER

Weld

GlassFish

Google Cloud Platform

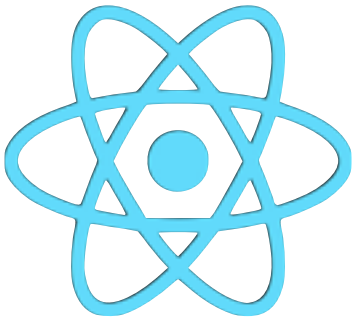
sonarsource

codenvy

Breizh C@mp

Mix de technologies

React



React by Facebook

- Librairie Javascript pour créer des interfaces graphiques
  - rend des vues et répond à des évènements
  - le V de MVC ;-)
  - pas de full stack
- Open source depuis 2013
- Cible essentiellement les grosses applications JS avec des données qui changent dans le temps
  - Utilisé en prod. sur votre mur Facebook (et surement ailleurs) et pour toute l'application web Instagram

Déclaratif

Exprimez à quoi doit ressembler votre app. à n'importe quel moment dans le temps

et React va automatiquement gérer toutes les mises à jour de l'UI quand les données sous-jacentes sont modifiées.

Simple

Quand vos données changent, conceptuellement

React appuie sur F5 (la vue est entièrement re-rendue)

et sait mettre à jour que les parties de l'UI qui ont réellement changées

Approche composant

Avec React

on fabrique des composants, pas des templates !!!

Les composants sont autonomes, réutilisables, composables et savent se rendre eux-même dans le DOM

Les composants sont des briques hautement cohésives et peu couplées avec les autres briques

Utilisateurs

Virtual DOM

- React rend entièrement la vue à chaque changement de son modèle
  - il très très compliqué de patcher la vue à la main quand les données changent en permanence => effets de bord
- Pour que ce soit performance, React s'appuie sur un DOM virtuel
  - React calcule la nouvelle vue
  - React la compare avec l'ancienne
    - calcule une liste de différences minimale
  - React batch tous les changements nécessaires vers le DOM

Synthetic events

- React ré-implémente un système d'évènement au dessus du DOM
  - bubbleling
  - Capture
  - etc …
- Conforme aux recommandations W3C
- Fonctionne de manière identique sur tous les navigateurs

Breizh C@mp
Mix de technologies

Demo (unleash the troll)

Votre premier composant

```
const Clicker = React.createClass({
  getInitialState() {
    return { count: 0 };
  },
  handleClick() {
    this.setState({ count: this.state.count + 1 });
  },
  render() {
    return (
      React.createElement("div", {style: { display: 'flex'}},
        React.createElement("span", null,
              "You have clicked ", this.state.count, " times"),
        React.createElement("button",
              {type: "button", onClick: this.handleClick},
                "Click me"
        )
      )
    );
  }
});
React.render(React.createElement(Clicker, null), mountNode);
```

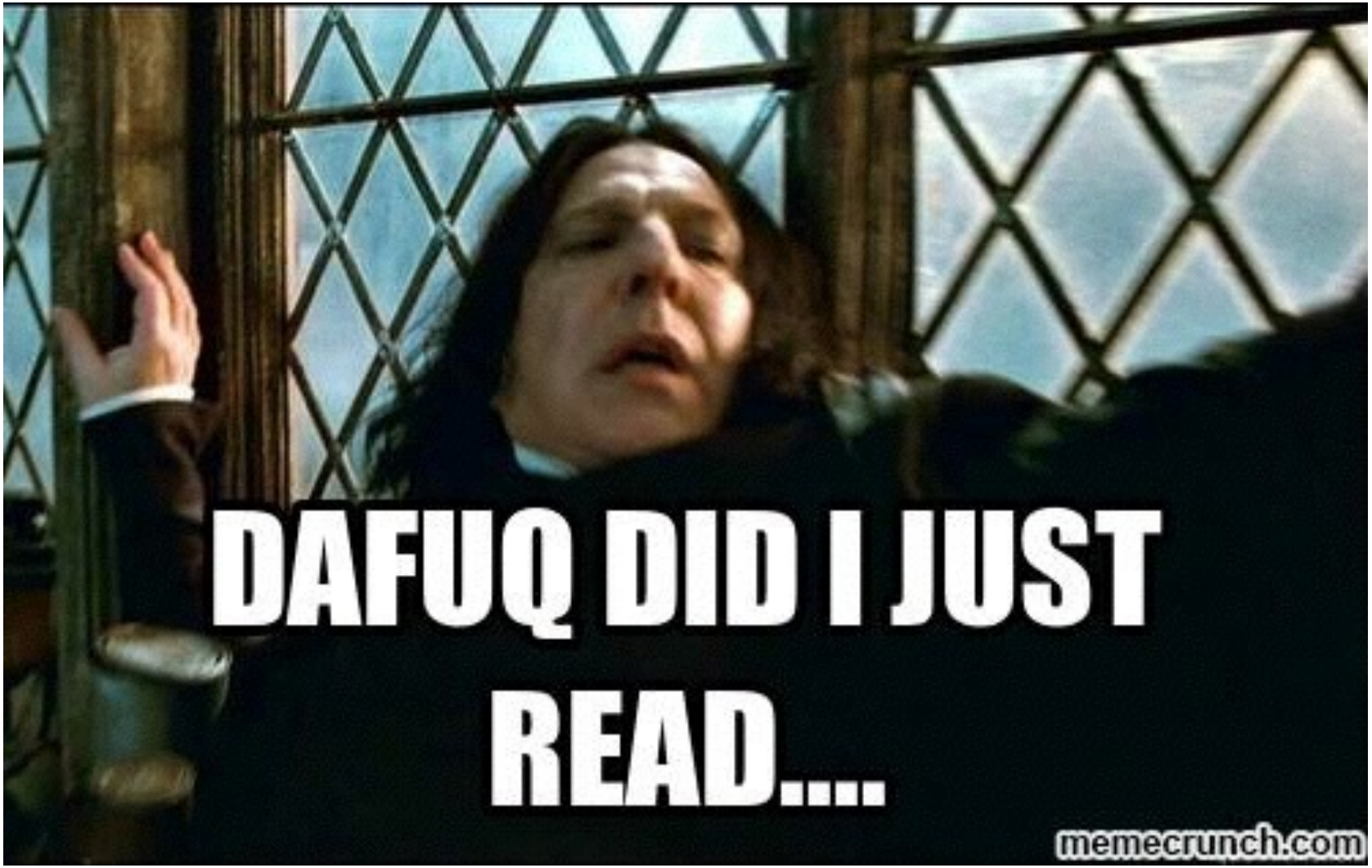
JSX

```
const Clicker = React.createClass({
  getInitialState() {
    return { count: 0 };
  },
  handleClick() {
    this.setState({ count: this.state.count + 1 });
  },
  render() {
    return (
      <div style={{ display: 'flex' }}>
        <span>You have clicked {this.state.count} times</span>
        <button type="button" onClick={this.handleClick}>
          Click me
        </button>
      </div>
    );
  }
});
React.render(React.createElement(Clicker, null), mountNode);
```

JSX

```
const Clicker = React.createClass({
  getInitialState() {
    return { count: 0 };
  },
  handleClick() {
    this.setState({ count: this.state.count + 1 });
  },
  render() {
    return (
      <div style={{ display: 'flex' }}>
        <span>You have clicked {this.state.count} times</span>
        <button type="button" onClick={this.handleClick}>
          Click me
        </button>
      </div>
    );
  }
});
React.render(React.createElement(Clicker, null), mountNode);
```

DAFUQ !!!



DAFUQ !!!

```
  return (
     <div>Seconds Elapsed: {this.state.secondsElapsed}</div>
  );
```

```
render: function() {
    return (
      <div>
        <h3>TODO</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit}>
          <input onChange={this.onChange} value={this.state.text} />
          <button>{'Add #' + (this.state.items.length + 1)}</button>
        </form>
      </div>
    );
  }
```

DAFUQ !!!

```
render() {
  return (
    React.createElement("div", null, "Seconds Elapsed: ", this.state.secondsElapsed)
  );
}

render() {
  return (
    React.createElement("div", null,
      React.createElement("h3", null, "TODO"),
      React.createElement(TodoList, {items: this.state.items}),
      React.createElement("form", {onSubmit: this.handleSubmit},
        React.createElement("input", {onChange: this.onChange, value: this.state.text}),
        React.createElement("button", null, 'Add #' + (this.state.items.length + 1))
      )
    )
  );
}
```

DAFUQ !!!

- Séparation des préoccupations ???
  - la vue ne devrait pas être avec le « contrôleur / composant »
  - Ce n'est pas un problème de couplage mais de cohésion
    - de toute façon votre template est fortement couplé à votre « viewModel »
    - la logique de rendu et le markup sont inévitablement couplés
  - Les templates séparent les technos, pas les préoccupations
    - Souvent limités, et en plus ils faut les apprendre ;-)
      - {{>}}, {{#each}}, etc …
      - Et on en invente tous les jours

Props & state

- Les deux sources de données d'un composant
- les propriétés, définies à la création d'une instance de composant
    - Idéalement immutable
  - peuvent avoir des valeurs par défaut pour faciliter la réutilisation
  - peuvent être validée pour faciliter la réutilisation

```
<Todolist title="Tasks" maxItems="10"
                handleDestroy={this.destroyTask} style={{ heigh: 250 }} />

this.props.maxItems;
this.props.destroyTask(task.id);
```

Props & state

- l'état, qui représente l'état interne d'une instance de composant
  - l'état est mutable, idéalement seulement par le composant lui-même
- C'est le changement de l'état qui lancera la mise à jour de la vue. Le double data binding est supporté mais non encouragé

```
this.setState({
  tasks: []
});

this.state.tasks.map(item => <Task key={task.id} task={item} />);
```

Cycle de vie

```
const Timer = React.createClass({
  getInitialState() {
    return {secondsElapsed: 0};
  },
  tick() {
    this.setState({secondsElapsed: this.state.secondsElapsed + 1});
  },
  componentDidMount() {
    this.interval = setInterval(this.tick, 1000);
  },
  componentWillUnmount() {
    clearInterval(this.interval);
  },
  render() {
    return (
      <div>Seconds Elapsed: {this.state.secondsElapsed}</div>
    );
  }
});
React.render(<Timer />, mountNode);
```

Cycle de vie

- componentWillMount
- componentDidMount
- componentWillReceiveProps
- shouldUpdateComponent
- componentWillUpdate
- componentDidUpdate
- componentWillUnmount

Validation

```
var MyComponent = React.createClass({
  propTypes: {
    tasks: React.PropTypes.array,
    displayDate: React.PropTypes.bool,
    formatDate: React.PropTypes.func,
    nrbOfItems: React.PropTypes.number,
    config: React.PropTypes.object,
    title: React.PropTypes.string,
    required: React.PropTypes.*.isRequired,
    ...
  },
  render() {
    ...
  }
});
```

Mixins

```
const SetIntervalMixin = {
  componentWillMount() {
    this.intervals = [];
  },
  setInterval(cb, delay) {
    this.intervals.push(setInterval(cb, delay));
  },
  componentWillUnmount() {
    this.intervals.map(clearInterval);
  }
};
```

Mixins

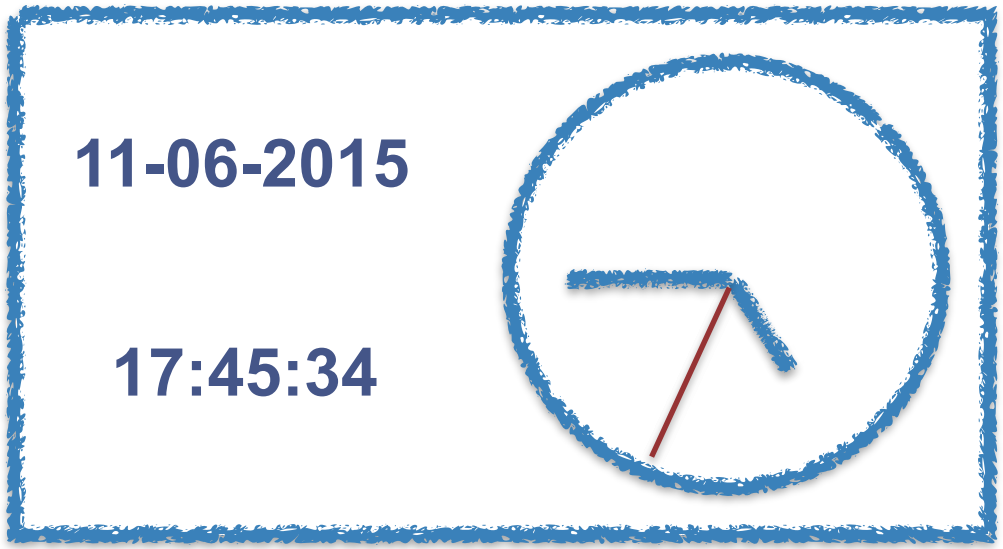
```
const TickTock = React.createClass({
  mixins: [SetIntervalMixin],
  getInitialState() {
    return {seconds: 0};
  },
  componentDidMount() {
    this.setInterval(this.tick, 1000);
  },
  tick() {
    this.setState({seconds: this.state.seconds + 1});
  },
  render() {
    return (
      <p>React has been running for
                          {this.state.seconds} seconds.</p>
    );
  }
});
```

Breizh C@mp
Mix de technologies

Demo

Breizh C@mp
Mix de technologies

Demo



Tester vos composants

```
import React from 'react/addons';
import Clicker from '../Clicker';

jest.dontMock('../Clicker.js');

describe('Clicker', () => {
  it('changes count after click', () => {
    const TestUtils = React.addons.TestUtils;

    const clicker = TestUtils.renderIntoDocument(<Clicker />);
    const span = TestUtils.findRenderedDOMComponentWithTag(clicker, 'span');
    const button = TestUtils.findRenderedDOMComponentWithTag(clicker, 'button');

    expect(span.getDOMNode().textContent).toEqual('You have clicked 0 times');
    TestUtils.Simulate.click(button);
    expect(span.getDOMNode().textContent).toEqual('You have clicked 1 times');
  });
});
```

ES6

- La dernière version de React (0.13.x) est compatible avec les classes ES6
- De plus il est simple d'interfacer le build avec Babel qui supporte JSX

```
import React from 'react/addons';

export default class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = {count: 0};
  }
  handleClick() {
    this.setState({count: this.state.count + 1});
  }
  render() {
    return (
      <div style={{ display: 'flex' }}>
        <span>You have clicked {this.state.count} times</span>
        <button type="button" onClick={this.handleClick.bind(this)}>Click me</button>
      </div>
    );
  }
}
```

Isomorphic apps

- Comme React s'appuie sur un DOM virtuel, il n'est pas obligatoire se brancher dans le DOM du navigateur
  - React peut générer une chaine de caractère HTML
    - Ou tout autre chose …

```
React.renderToString(React.createElement(Clicker, null));
React.renderToStaticMarkup(React.createElement(Clicker, null));
```

- Possibilité de rendre ses vues React sur le serveur
  - Node, Nashorn, etc …
- pour ensuite laisse la version client prendre le relai dès que tout est chargé

Web components

- React support de rendu de WebComponents en tant qu'élément du DOM
- React ne supporte par l'exposition d'un composant en tant que web component
  - cependant c'est une feature prévue pour les prochaines version
  - l'équipe React travaille sur ce point, afin que l'intégration soit parfaite ;-)
  - En attendant, différentes librairies permettent de le faire

Breizh C@mp
Mix de technologies

Demo

Ecosystème

- React possède un écosystème très riche et en pleine expansion
  - Bibliothèques de composants graphiques
    - intégration bootstrap
    - intégration matérial design
    - etc …
  - Intégration avec d'autres librairies graphiques
    - Highcharts
    - etc …
  - Accélération des rendus Angular ;-)

React Native

- Version de React qui ne cible pas le DOM mais un moteur de rendu native sur device mobile
  - version Android (sortira dans 6 mois)
  - version iOS sortie en 0.2
    - encore un peu difficile a utiliser, mais déjà de bon résultats
- Learn Once, write everywhere
  - les composants sont différents pour chaque plateforme, il est donc toujours nécessaire d’écrire plusieurs versions de la même application

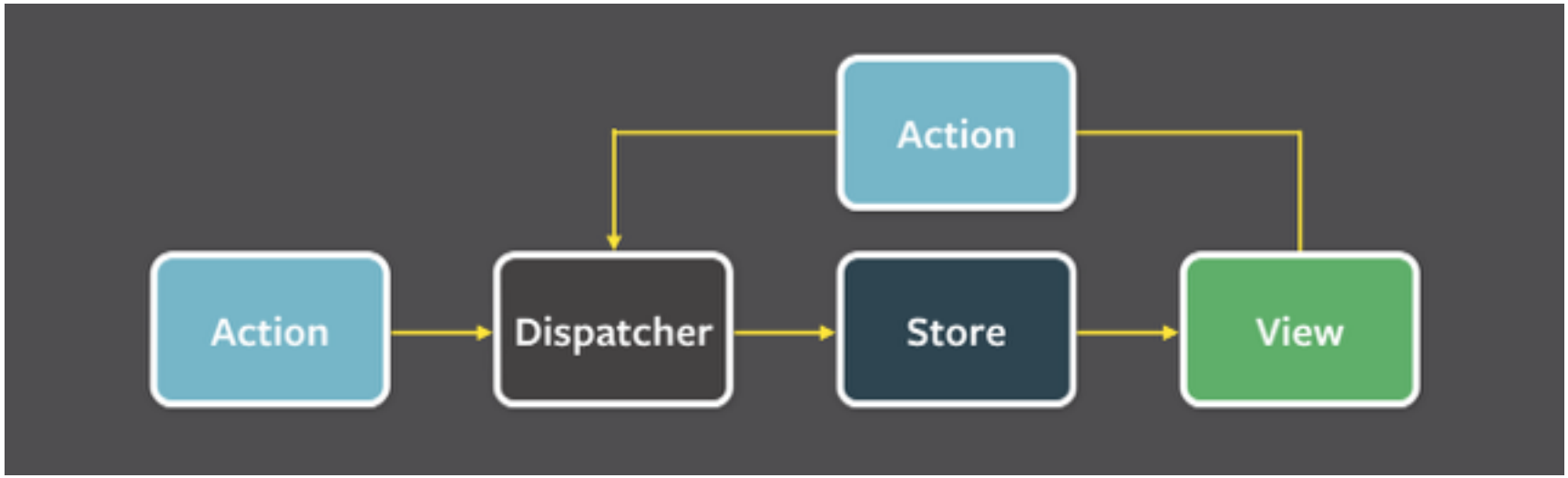
Breizh C@mp
Mix de technologies

Demo

Flux

- Plus un pattern qu'un framework, d'ou une explosion des implémentations :-)
- C'est la façon qu'a choisi Facebook pour construire ses applications en complément de React
  - basé sur le principe de flux de données unidirectionnel
  - plus simple à maintenir

Flux



Relay / GraphQL

- Framework fournissant du data fetching intelligent
- Chaque composant React déclare ses dépendances en terme de données
  - utilise un langage de description des données, GraphQL
- Relay compose toutes les dépendances
  - fetch, update optimisés
- Relay est une sorte d'unique store Flux permettant de mettre à jour tous les composants
- Devrait être open source sous peu

Relay / GraphQL

```
const FriendInfo = React.createClass({
  statics: {
    queries: {
      user() {
        return graphql`
          User {
            name,
            mutual_friends { count }
          }
        `;
      }
    }
  },
  render() {
    return (
      <div>
        <span>{this.props.user.name}</span>
        <span>{this.props.user.mutual_friends.count} mutual friends</span>
      </div>
    );
  }
});
```

Breizh C@mp

Mix de technologies

Q & A