


Introduction à React.js

HETIC P2023

Hello 🧑

Grégoire Mielle

 Twitter: @greeeg

 GitHub: @greeeg

 Email: hello@greeeg.com

 Étudiant en H5

 Software Engineer @PayFit

 Freelance

Une expertise pour tout le monde

Une compétence pour vos futurs projets & stages

Si vous êtes perdu, dites-le

Challengez-moi !

Méthodologie



Must read



Must watch



Must learn

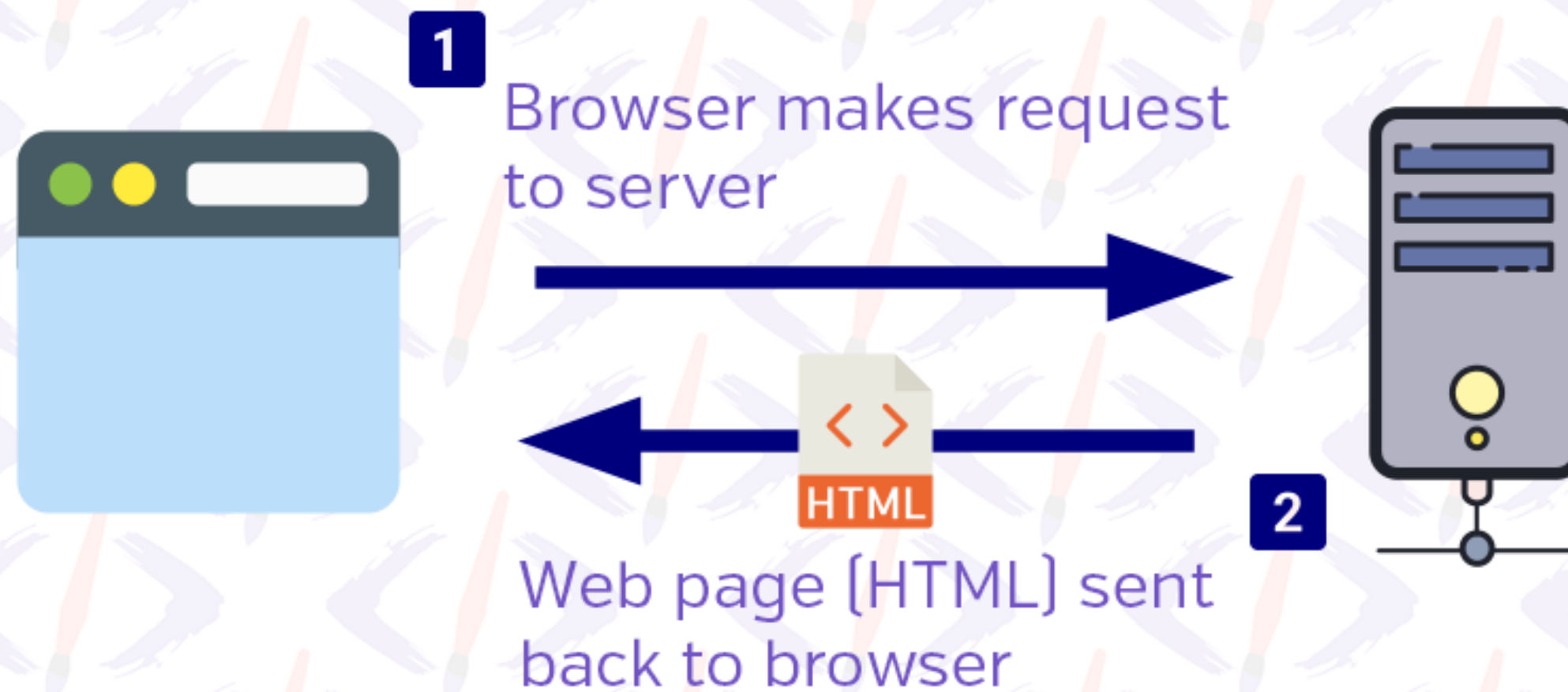


Action time

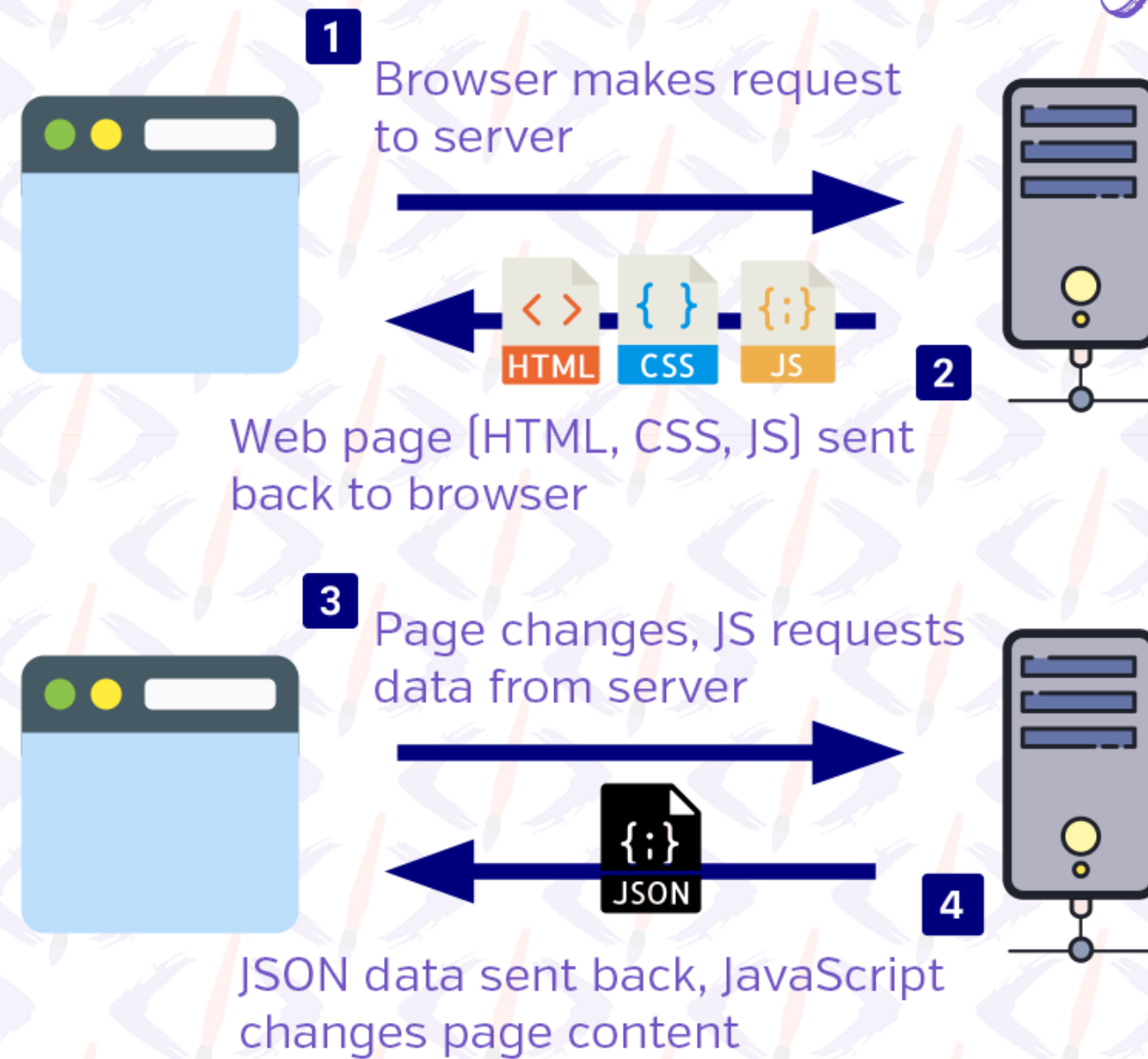
Un peu d'histoire...

Comment fonctionne l'affichage d'une page web ?

Server-Side Rendering



Client-Side Rendering





Quel est le lien entre toutes ces technologies ?

Technologies client-side

Interfaces plus complexes

Beaucoup plus d'interactions

Les besoins côté utilisateur ont évolué

Expérience mobile (3G, offline, etc.)

Sources de données nombreuses



craigslist

post to classifieds

my account

search craigslist

search

event calendar

M	T	W	T	F	S	S
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17

help, faq, abuse, legal

avoid scams & fraud

personal safety tips

terms of use

privacy policy

system status

about craigslist

craigslist is hiring in sf

craigslist open source

craigslist blog

best-of-craigslist

craigslist TV

SF bay area ^w

sfc

sby

eby

pen

nby

scz

community

activities local news
artists lost+found
childcare musicians
classes pets
events politics
general rideshare
groups volunteers

personals

strictly platonic
women seek women
women seeking men
men seeking women
men seeking men
misc romance
casual encounters
missed connections
rants and raves

discussion forums

apple help photo
arts history p.o.c.
atheist housing politics
autos jobs psych
beauty jokes queer
bikes kink recover
celebs legal religion
comp linux romance
crafts m4m science
diet manners spirit
divorce marriage sports

housing

apts / housing
housing swap
housing wanted
office / commercial
parking / storage
real estate for sale
rooms / shared
rooms wanted
sublets / temporary
vacation rentals

for sale

antiques farm+garden
appliances free
arts+crafts furniture
atv/utv/sno garage sale
auto parts general
baby+kid heavy equip
barter household
beauty+hlth jewelry
bikes materials
boats motorcycles
books music instr
business photo+video
cars+trucks rvs+camp
cds/dvd/vhs sporting
cell phones tickets
clothes+acc tools
collectibles toys+games
computers video gaming

jobs

accounting+finance
admin / office
arch / engineering
art / media / design
biotech / science
business / mgmt
customer service
education
food / bev / hosp
general labor
government
human resources
internet engineers
legal / paralegal
manufacturing
marketing / pr / ad
medical / health
nonprofit sector
real estate
retail / wholesale
sales / biz dev
salon / spa / fitness
security
skilled trade / craft
software / qa / dba
systems / network
technical support
transport
tv / film / video
web / info design

english

nearby cl

bakersfield
chico
fresno
gold country
hanford
humboldt
inland empire
klamath falls
las vegas
los angeles
medford
mendocino co
merced
modesto
monterey
orange co
palm springs
redding
reno
roseburg
sacramento
san luis obispo
santa barbara
santa maria
siskiyou co
stockton
susanville
ventura
visalia-tulare
yuba-sutter

us cities

us states

Home

Moments

Notifications

Messages

Twitter

Search Twitter

Tweet

</

De nouvelles solutions ont émergé

Jquery (2006)

Angular.js (2009)

Backbone.js (2010)

Ember.js (2011)

React.js (2013)

Vue.js (2014)



Comment bien les choisir ?

Quelles différences/ressemblances ?



State of JS - 2019 Edition

Install

```
> npm i react
```

↓ Weekly Downloads

7,380,054



Version

16.13.0

License

MIT

 facebook / react

 Used by ▼ 3.2m

 Watch ▼ 6.6k

★ Unstar 145k

 Fork 27.8k

↔ Code

! Issues 466

 Pull requests 88

▶ Actions

 Projects 0

 Wiki

 Security

 Insights

A declarative, efficient, and flexible JavaScript library for building user interfaces. <https://reactjs.org>

javascript

react

frontend

declarative

ui

library



React Developer Tools

Proposé par : Facebook



1247

[Outils de développement](#)



2 000 000+ utilisateurs

React.js

A JavaScript library for building user interfaces

Declarative

Component-Based

Learn Once, Write Anywhere

Declarative vs Imperative

Dessine un chat. vs Fais un trait noir, puis un arrondi vers le bas, etc.

Imperative

<i>fx</i>				
	A	B	C	D
1	Last name	Firstname	Full name	
2	John	Doe	John Doe	
3				

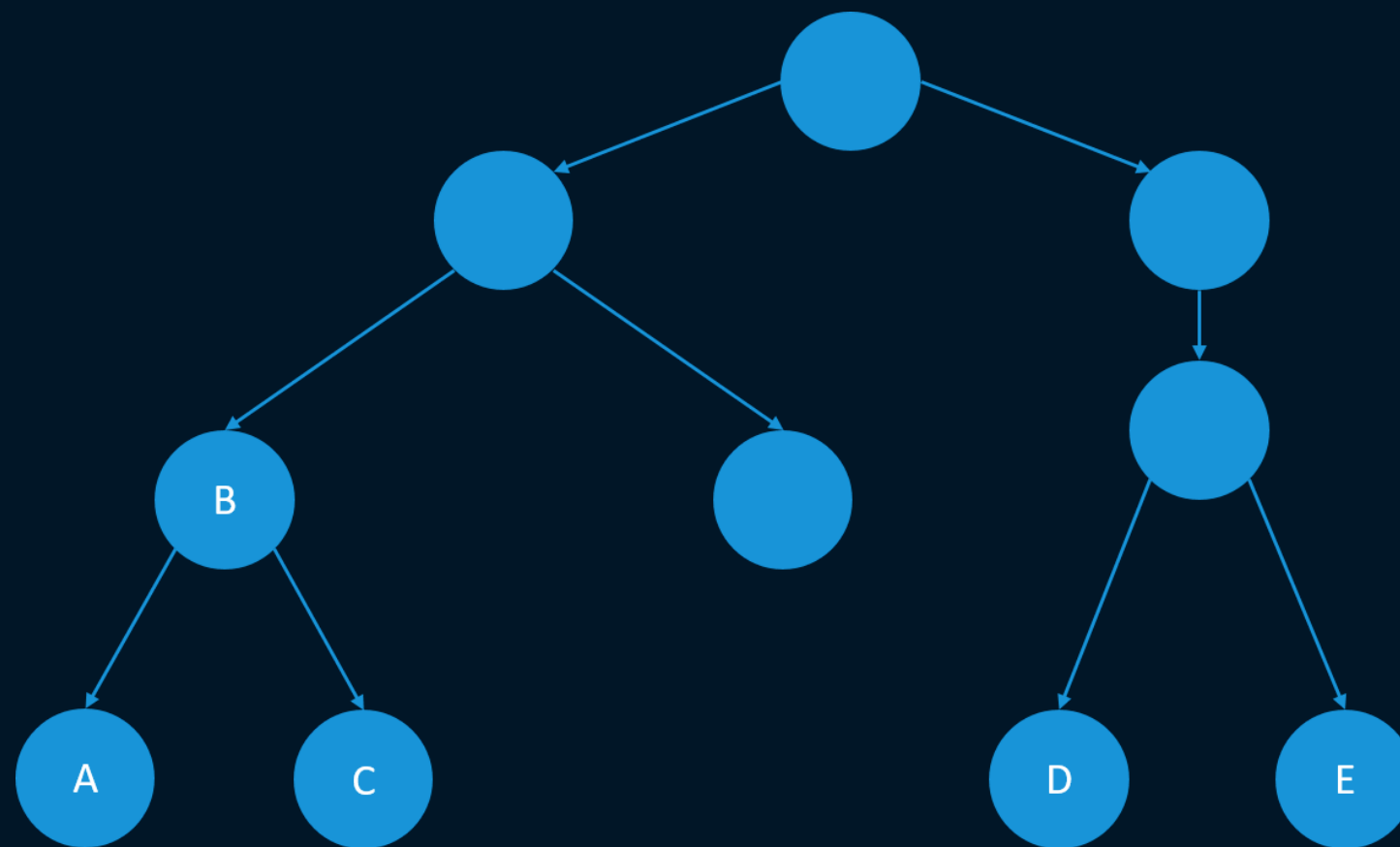
On doit effectuer un calcul à chaque modification d'une des 2 variables qui définissent "fullname"

Declarative

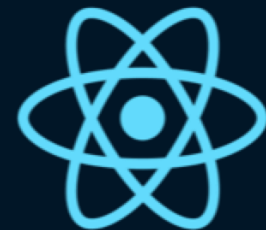
<i>fx</i>			
	A	B	C
1	Last name	Firstname	Full name
2	John	Doe	John Doe
3			

On déclare comment "fullname" est défini sans jamais le modifier
manuellement

Component-based



Learn Once, Write Anywhere



React Native



oculus



React Podcast: Evan Bacon on Expo and the Future of
"Build Once; Run Anywhere"

Qui l'utilise ?

Facebook (FB.com, WhatsApp, Instagram)

American Express

Apple

Baidu

Netflix

Uber

Qui l'utilise ?

Skype

Tesla

Slack

Discord

Walmart

Pinterest

Qui l'utilise ?

Twitter

PayPal

Tinder

AirBnB

Dropbox

PayFit

Hello World 🚀

Comment crée-t-on un élément du DOM en
Javascript ?


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

  <script type="text/javascript">
    function addHelloWorld() {
      const root = document.querySelector('#root')
      const element = document.createElement('div')
      element.textContent = 'Hello World'
      element.className = 'container'
      root.appendChild(element)
    }
  </script>
</body>
```

On l'ajoute dans le parent

Avec React.js

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

  <script src="https://unpkg.com/react@16.12.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.12.0/umd/react-dom.development.js"></script>
  <script type="text/javascript">
    function addHelloWorld() {
      const root = document.getElementById('root')
      const element = React.createElement('div', {
        className: 'container',
        children: 'Hello World',
      })
      ReactDOM.render(element, root)
    }
  </script>
</body>
```

On l'ajoute dans le parent

Pour utiliser React.js on a donc besoin de 2 packages

React: Les fonctionnalités permettant de définir des composants React

React DOM: Un moteur de rendu pour le web

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

  <script src="https://unpkg.com/react@16.12.0/umd/react.development.js"></script>
  <script src="https://unpkg.com/react-dom@16.12.0/umd/react-dom.development.js"></script>
  <script src="https://unpkg.com/@babel/standalone@7.8.3/babel.js"></script>
  <script type="text/babel">
    function addHelloWorld() {
      const root = document.getElementById('root')
      const element = <div className="container">Hello World</div>
      ReactDOM.render(element, root)
    }
  </script>
</body>
```

Du HTML dans du Javascript 🤔

Le JSX

Une extension syntaxique au Javascript

Transpilé par **Babel** en Javascript

Pourquoi le JSX ?

Permet de définir des arbres de composants simplement

S'appuie sur une syntaxe familière (le HTML)

⚠ Pour utiliser du **JSX**, on doit avoir Babel configuré sur son projet


```
const message = (  
  <div>  
    <p>Hello World</p>  
  </div>  
)  
ReactDOM.render(message, document.getElementById('root'))
```

Si la variable contient des éléments imbriqués, on l'écrit comme ça

Embedded expressions

```
const price = 12.5
const tva = 19.6
const total = price + (price * (tva / 100))

const message = <p>The price with included taxes is {parseInt(total)} euros.</p>
ReactDOM.render(message, document.getElementById('root'))
```

Et aussi appeler des fonctions dans le JSX

Les composants React.js

On définit un composant comme étant un
élément/objet qui entre dans la composition de
quelque chose

```
const julia = 'Julia Robert'  
const thomas = 'Thomas René'  
  
function wishHappyBirthday(name) {  
  return `HAPPY BIRTHDAY ${name.toUpperCase()}!`  
}  
  
const wishMessageJulia = wishHappyBirthday(julia)  
const wishMessageThomas = wishHappyBirthday(thomas)
```

Pour simplifier la tâche, on crée une fonction qui génère les messages

```
const julia = 'Julia Robert'  
const thomas = 'Thomas René'  
  
function WishHappyBirthday() {  
  return (  
    <p>HAPPY BIRTHDAY!</p>  
  )  
}
```

On peut, comme en Javascript, créer une fonction qui génère ces messages

Les composants React.js

Des fonctions qui retournent des éléments React.js

Ou d'autres composants (fonctions) qu'on a créé

Des briques d'interface réutilisables


```
const WishHappyBirthday = () => (  
  <p>HAPPY BIRTHDAY!</p>  
)  
  
const app = (  
  <div>  
    <WishHappyBirthday />  
    <a href="/index.html">Link to home</a>  
    <WishHappyBirthday />  
  </div>  
)
```

On encore comme ceci

Les props

Les props sont des paramètres/arguments passés
aux composants React.js pour :

- afficher quelque chose de dynamique
- pour faire du rendu conditionnel
- pour passer ce même paramètre à un composant enfant

```
function WishHappyBirthday({ name }) {  
  return (  
    <p>HAPPY BIRTHDAY {name.toUpperCase()}!</p>  
  )  
}  
  
const App = () => (  
  <WishHappyBirthday name="Julia" />  
)
```

On peut simplifier l'écriture en destructurant l'objet props

En résumé

React.js est une librairie pour construire des interfaces créées à partir de briques réutilisables (components) qui utilise une extension syntaxique appelée le JSX

En résumé

elle s'appuie sur 2 packages (`react` & `react-dom`)

mais peut aussi être utilisée ailleurs (native, VR, etc.)

```
const element = <a href="https://google.com" title="Google">Search on Google</a>

function Component() {
  return (
    <a href="https://google.com" title="Google">Search on Google</a>
  )
}

const App = () => (
  <div>
    {element}
    <Component />
  </div>
)
```

Attention à ne pas confondre composants et éléments

Le style en React.js

**Il existe trois manières de styliser des applications
React**

En utilisant du CSS classique

En utilisant du `style inline`

En utilisant des librairies de CSS-in-JS

(Nous reviendrons dessus plus tard)



Construire sa première interface React.js

Nous allons réaliser un **clone de Slack** à partir d'**une intégration front**





Construire une interface à partir de composants réutilisables

Avoir de petites entités autonomes

Les faire évoluer de manière isolée

Travailler plus efficacement à plusieurs

Gestion des évènements

De manière générale, on utilise le Javascript pour dynamiser des pages web :

Pour créer des interactions (Click, Keydown, etc)

Pour créer des animations

Et bien plus (AJAX, WebGL, Web APIs, etc.)

```
const input = document.querySelector('input')

input.addEventListener('keyup', e => {
  const text = e.target.value

  if (text.length > 10) {
    alert('This is way too long')
  } else {
    document.title = `(${text}) Learning Javascript`
  }
})
```

```
const TextInput = () => {
  const onChange = e => {
    const text = e.target.value

    if (text.length > 10) {
      alert('This is way too long')
    } else {
      document.title = `(${text}) Learning Javascript`
    }
  }

  return (
    <input type="text" onKeyDown={onChange} />
  )
}

const App = () => (
  <TextInput />
)
```

Le state

Le state est similaire aux **props**. Il permet d'apporter une notion de dynamisme au composant.

Privé

Contrôlé par le composant



Pourquoi ne pas utiliser simplement une variable ?

Qu'est-ce que veut dire `useState` ?

useState est un mécanisme propre à React qui
permet de :

Affecter une nouvelle valeur à la variable

Prévenir React de ce changement

On appelle cela un **hook**

Les hooks ont été introduit dans React à partir de la version **16.8**. Ils permettent d'utiliser des fonctionnalités de React sans écrire de **Class**.

```
import React, { Component } from 'react'

class Counter extends Component {
  constructor(props) {
    super(props);

    this.state = {
      count: 0
    };
  }

  render() {
    return (
      <button onClick={() => this.setState({ count: this.state.count + 1 })}>
        Clicked: {count}
      </button>
    )
  }
}
```

Mais on peut aussi l'écrire de cette manière



Construire sa première interface interactive avec React.js

Nous allons réaliser l'exemple des Threads avec React.js à partir de cette source

Le cycle de vie des composants

Un composant React.js peut être mis à jour :

- lorsqu'une de ses props change
- lorsque son state change

On peut greffer un **side-effect** à ces changements en utilisant le hook **useEffect**

On peut préciser une dépendance au useEffect

De cette manière, la fonction qu'il contient ne sera appelée que lorsque l'une
de ces dépendances change

On peut également préciser une fonction dite de **cleanup** à appeler lorsque le composant est démonté

Quelques règles liées aux hooks

Only Call Hooks **at the Top Level**

Only Call Hooks **from React Functions**

Les listes

Les champs de formulaire

Controlled component

Certains éléments du DOM maintiennent leur état dans une de leur propriété

C'est le cas des `Input` dans la propriété `value`

On parle de `controlled component` lorsque leur valeur est maintenue dans le state



Construire une TODO app

Nous allons réaliser une **TODO app** comme celle-ci à **partir d'une intégration front**