

Introduction à React.js

HETIC P2023

Hello



Grégoire Mielle

📢 Twitter: [@greeeg](https://twitter.com/greeeg)

🔬 GitHub: [@greeeg](https://github.com/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 !

Un peu d'histoire...

Comment fonctionne l'affichage d'une page web ?

Server-Side Rendering

1

Browser makes request
to server



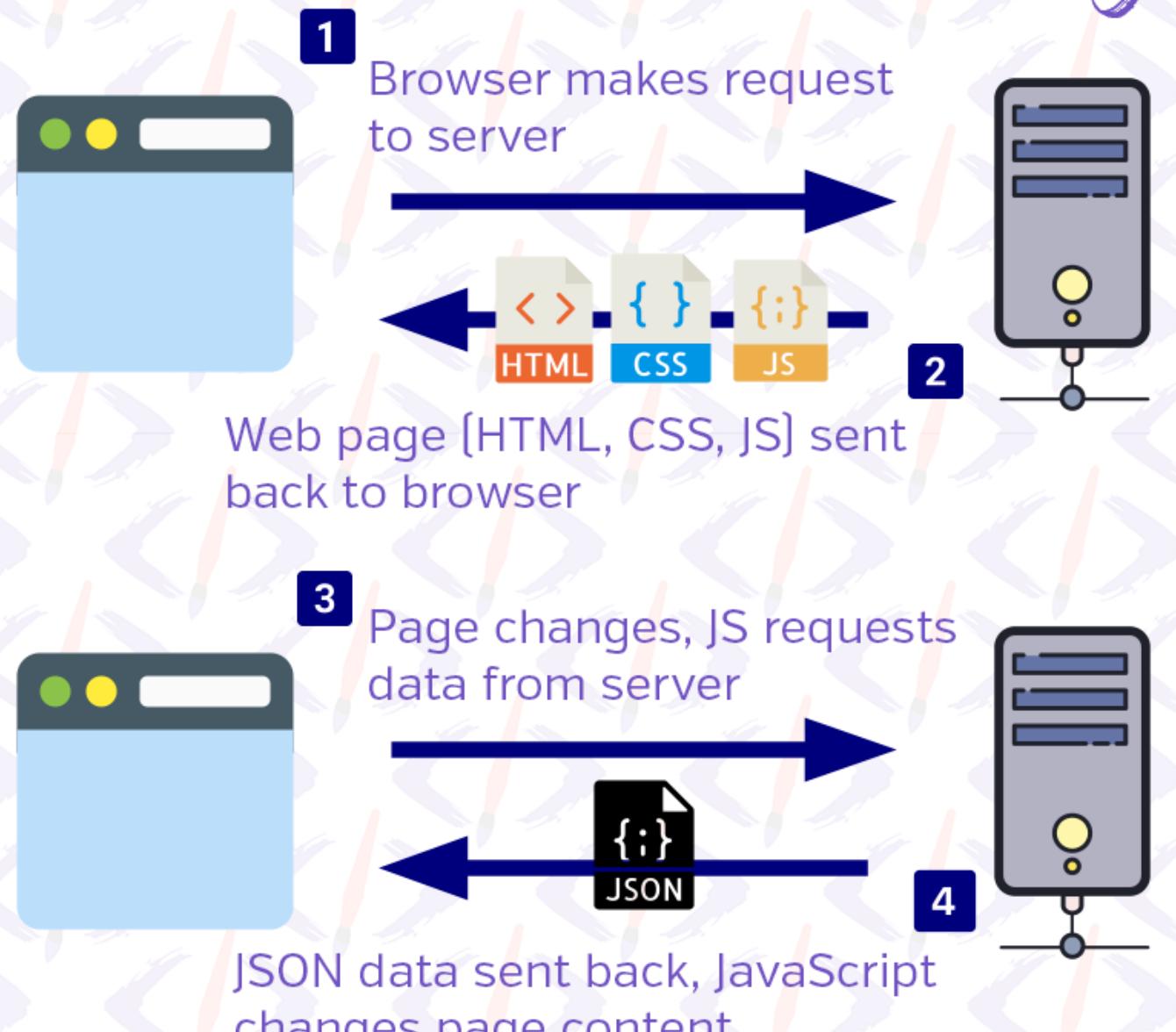
2

Web page [HTML] sent
back to browser



Comment le web fonctionne ?

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](#)

[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	housing	jobs
activities	apts / housing	accounting+finance
artists	housing swap	admin / office
childcare	housing wanted	arch / engineering
classes	office / commercial	art / media / design
events	parking / storage	biotech / science
general	real estate for sale	business / mgmt
groups	rooms / shared	customer service

personals	for sale	
strictly platonic	antiques	farm+garden
women seek women	appliances	free
women seeking men	arts+crafts	furniture
men seeking women	atv/utv/sno	garage sale
men seeking men	auto parts	general
misc romance	baby+kid	heavy equip
casual encounters	barter	household
missed connections	beauty+hlth	jewelry
rants and raves	bikes	materials

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

english ▼

nearby cl

- [bakersfield](#)
- [chico](#)
- [fresno](#)
- [gold country](#)
- [hanford](#)
- [humoldt](#)
- [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  Search Twitter  Tweet

What's happening?

See 2 new Tweets

Stephanie Hurlburt Retweeted

Dana Todd @danatodd · 3h
According to this article, online made-to-measure has doubled in sales over the last 5 years to become \$2B market. That's great news for @balodanafashion and a triple win for #espoke #fashion #fundraising !

Carolyn Said @CSaid
Custom clothes at scale - does the Internet make them feasible?
sfchronicle.com/business/article...

Sarah Drasner liked

April HalloWensel 🎃 @aprilwensel · 2h
Such truth! 🤝

Speech Moves @SpeechMoves
"The world is beautiful and inspirational and sad and complex - and everything in between...our greatest roles are how we show up as humans." -@katcoleatl

Who to follow · Refresh · View all

React Amsterdam @Rea... 

Krzysztof Magiera @kzzzf 

Followed by Ryan Florence and others

Amjad Masad @amasad 

Find people you know

© 2018 Twitter About Help Center Terms Privacy policy Cookies Ads info Brand Blog Status Apps Jobs Marketing Businesses Developers

Advertise with Twitter

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](#)



2000 000+ utilisateurs

React.js

A JavaScript library for building user interfaces

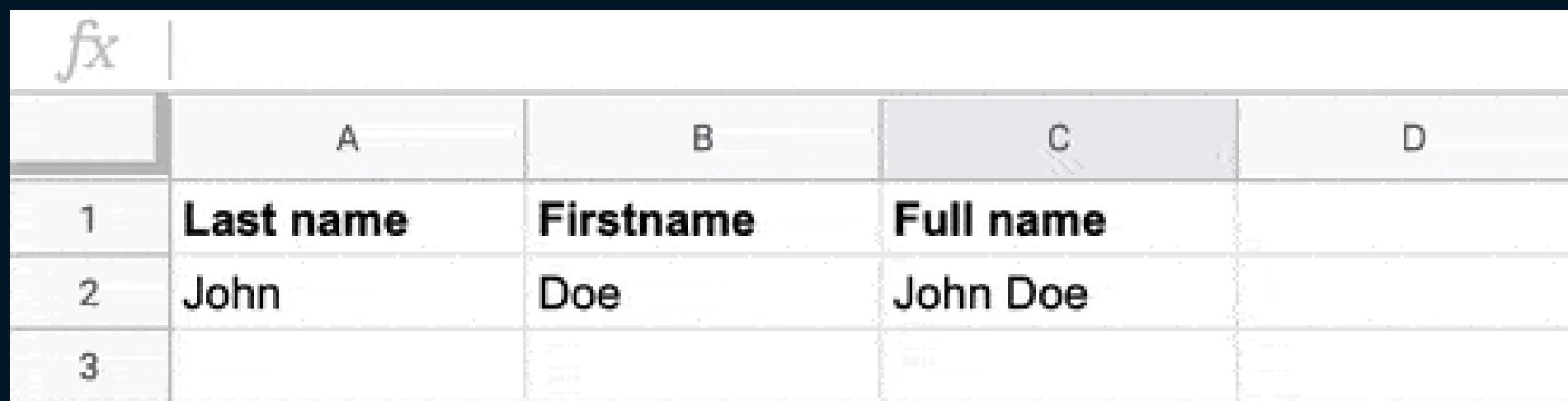
**Declarative
Component-Based
Learn Once, Write Anywhere**

Declarative vs Imperative

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



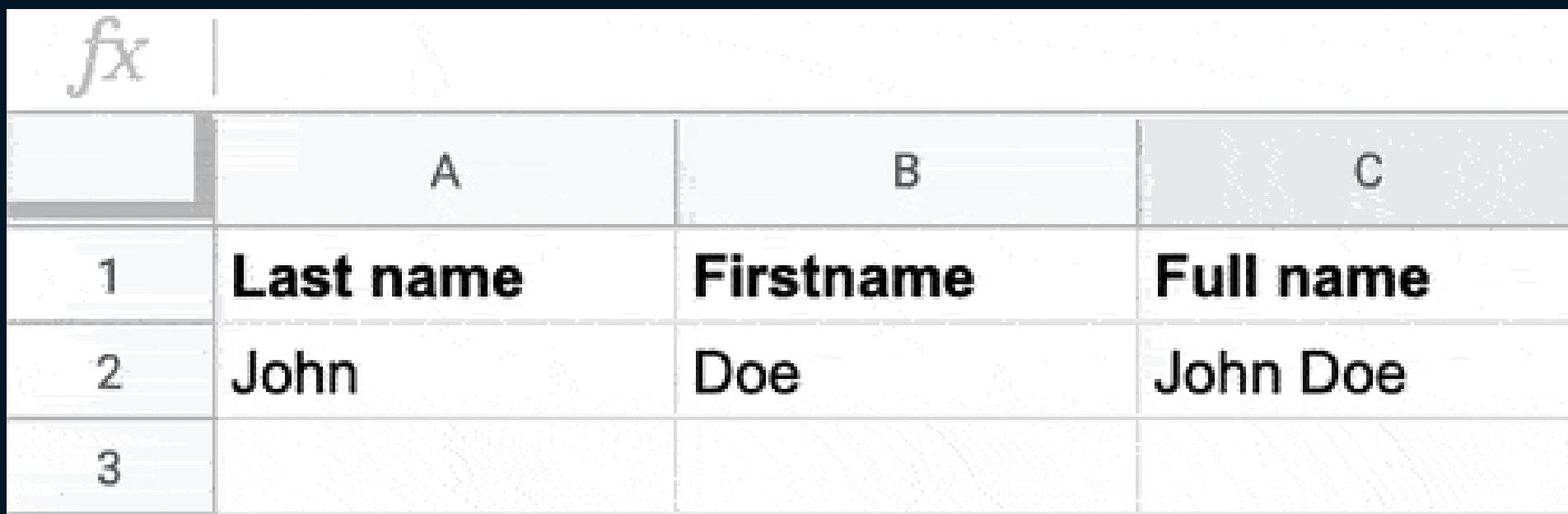
Imperative



	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"

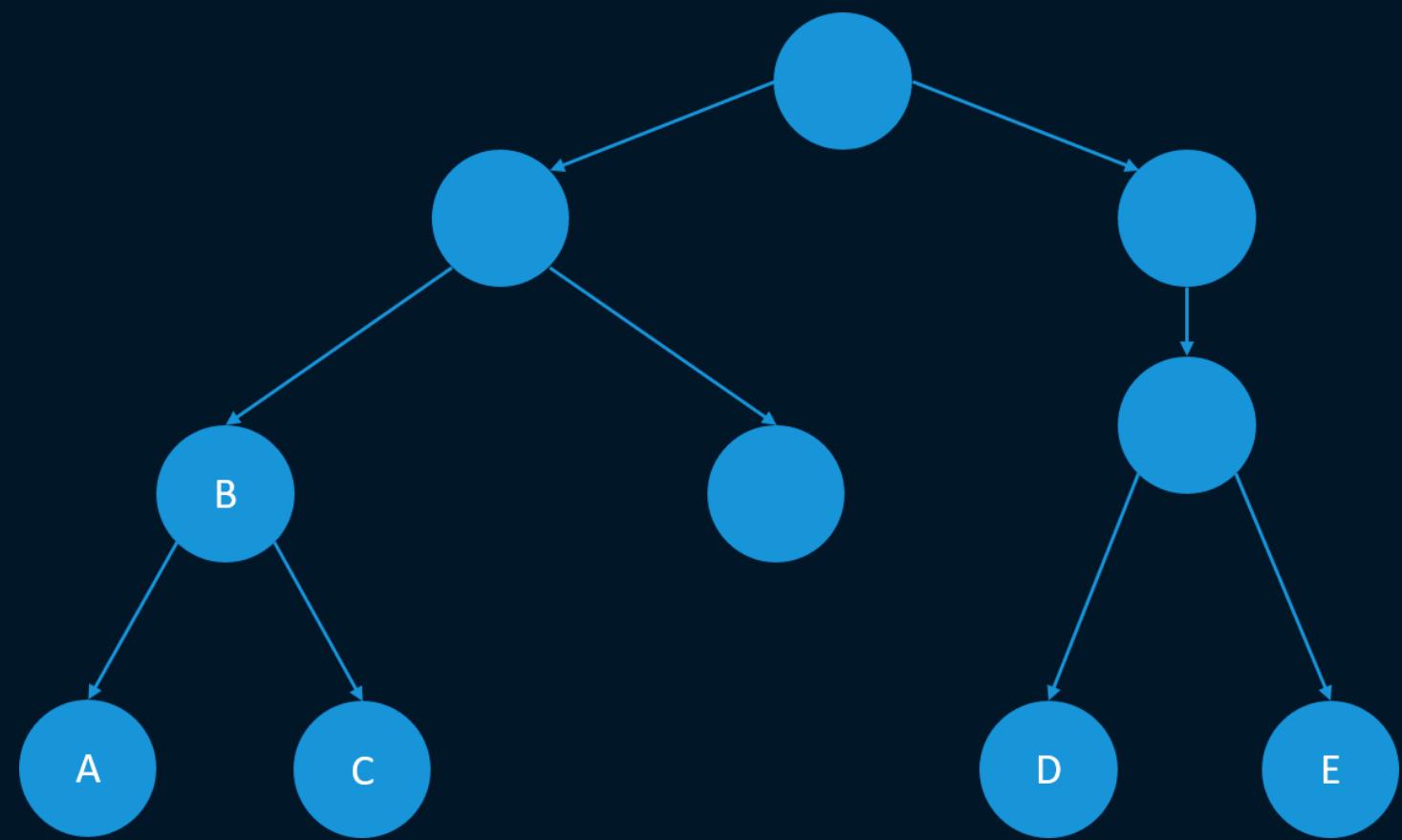
Declarative



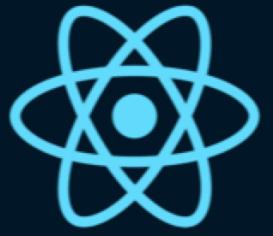
	A	B	C
1	Last name	Firstname	Full name
2	John	Doe	John Doe
3			

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

Component-based



Learn Once, Write Anywhere



React Native



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

Les bases de React.js

Le JSX

Les composants

Les props

Le style

Gestion des évènements

Le state

Le cycle de vie

Le rendu conditionnel

Les listes

Les formulaires

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

Transpile par **Babel** en Javascript



WTF is JSX?

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 déstructurant 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)





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.)

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



Nous allons réaliser une interface de Threads à partir d'une intégration front

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

Le rendu conditionnel

Il permet de rendre un composant que si certaines conditions sont respectées

```
const FormInput = () => {
  const containsError = true;
  return (
    <div>
      <input type="text" placeholder="Name" />
      {containsError && <p>Please fill your name</p>}
    </div>
  );
};
```

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

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

const TextInput = () => {
  const [value, setValue] = useState('Greg');

  return (
    <input
      type="text"
      value={value}
      onChange={(e) => setValue(e.target.value)}
    />
  );
}
```

Avantages/Inconvénients de React.js

Avantages

Approche orientée composant
Écosystème/Communauté

Inconvénients

Slow startup time

Bundle Size

SEO

Alternatives

Vue.js

Preact

Angular

Svelte.js

Aller plus loin avec React.js

create-react-app

React Dev Tools

Déployer une app React.js

L'AJAX avec React.js

Le routing

Le CSS-in-JS

Les performances

create-react-app

`create-react-app` est un outil permettant de créer et maintenir des applications React.js facilement. Il prend la forme d'un CLI (Command line interface) qui génère une structure de dossiers d'application React.js.

Avantages

Pas de configuration initiale

Plein d'éléments intégrés par défaut (Webpack, Dev server, Babel)

Maintenu par la communauté React.js

Inconvénients

Difficile à personnaliser pour des cas particuliers

Une fois qu'on **eject**, impossible de revenir en arrière

Prérequis

Node.js (>8.10)

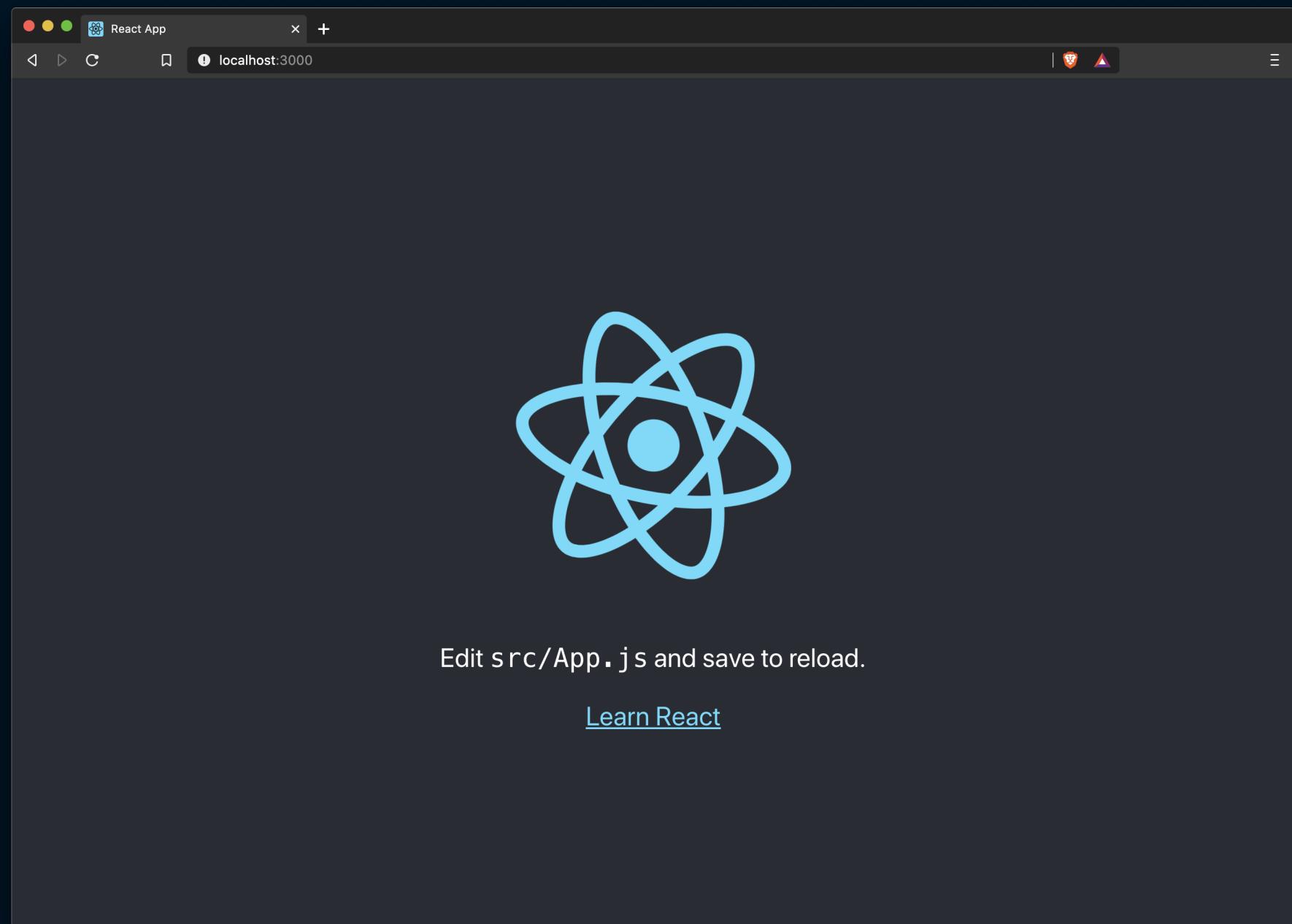
NPM (>5.2) ou Yarn (>0.25)


```
{  
  "scripts": {  
    "start": "react-scripts start",  
    "build": "react-scripts build",  
    "test": "react-scripts test",  
    "eject": "react-scripts eject"  
  }  
}
```

Le package **react-scripts** contient toute la logique que nous aurions du mettre en place manuellement avec Webpack.

```
app-name
├── README.md
├── node_modules
├── package.json
└── .gitignore
public
├── favicon.ico
├── index.html
├── logo192.png
├── logo512.png
└── manifest.json
    └── robots.txt
src
├── App.css
├── App.js
├── App.test.js
├── index.css
├── index.js
└── logo.svg
    └── serviceWorker.js
```

Le point d'entrée de l'application est le fichier
src/index.js.



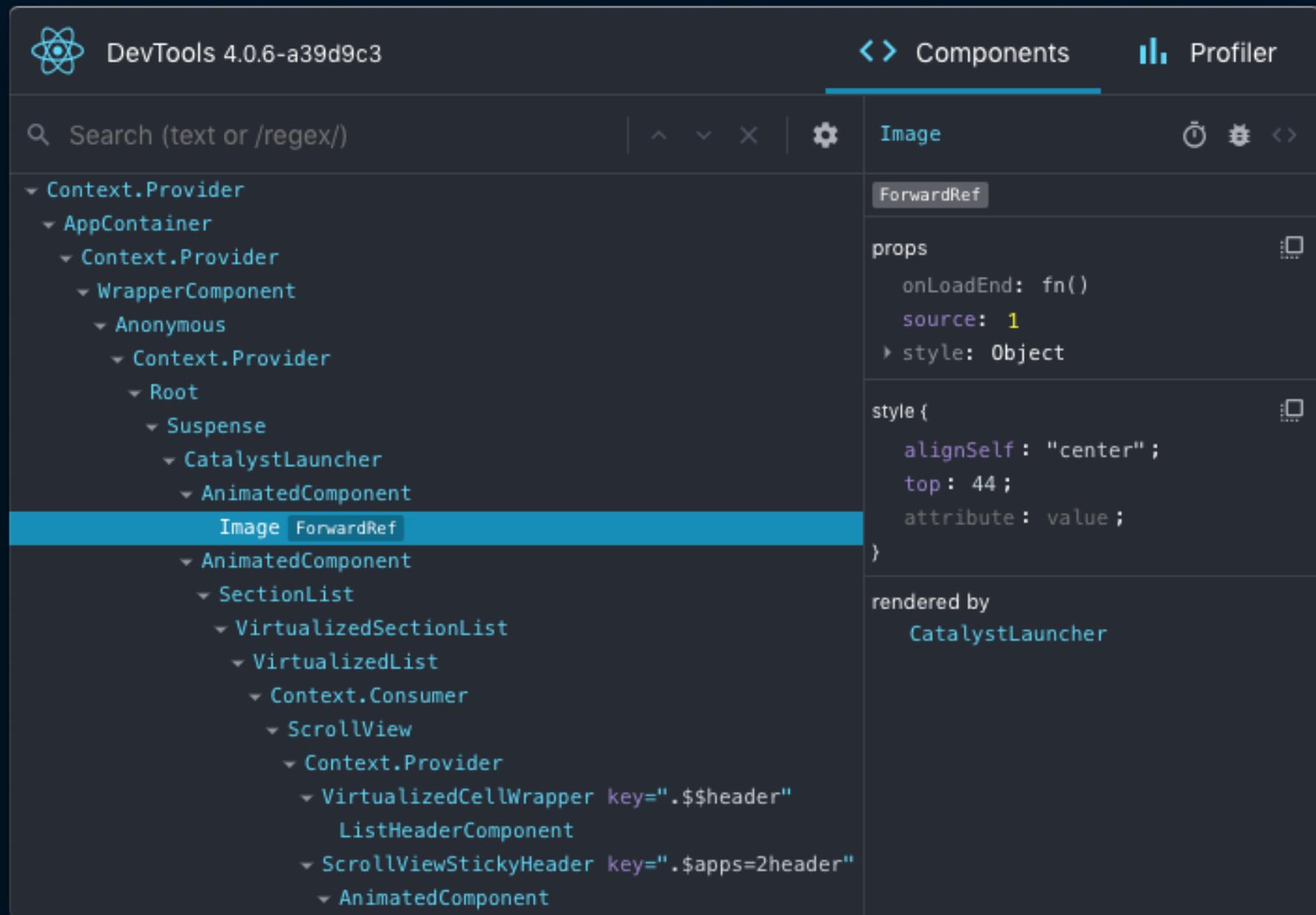
En utilisant la commande **yarn start**, cette page devrait s'afficher dans votre navigateur.

Vous êtes prêts à travailler, happy coding 😊.

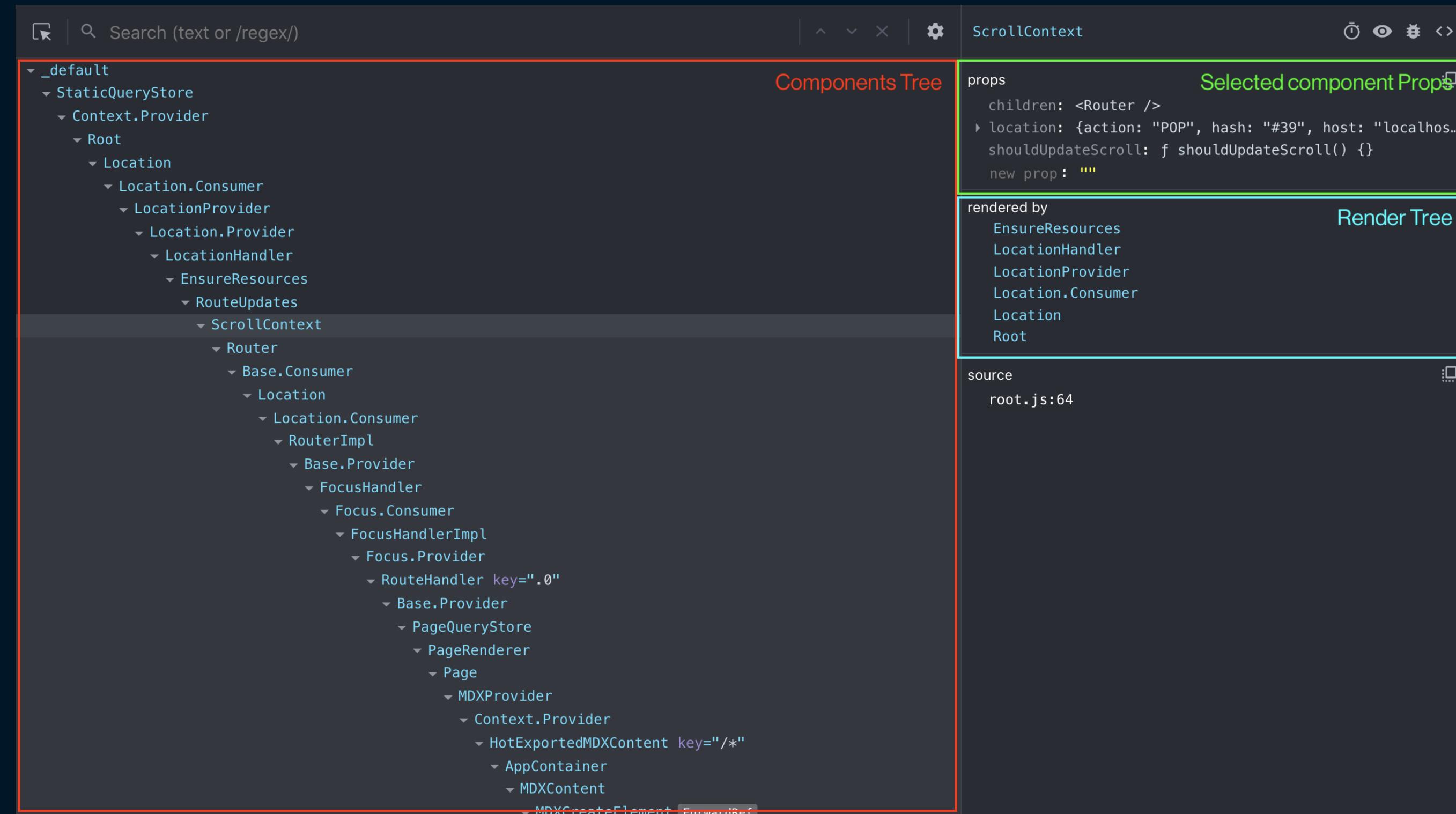
React Dev Tools

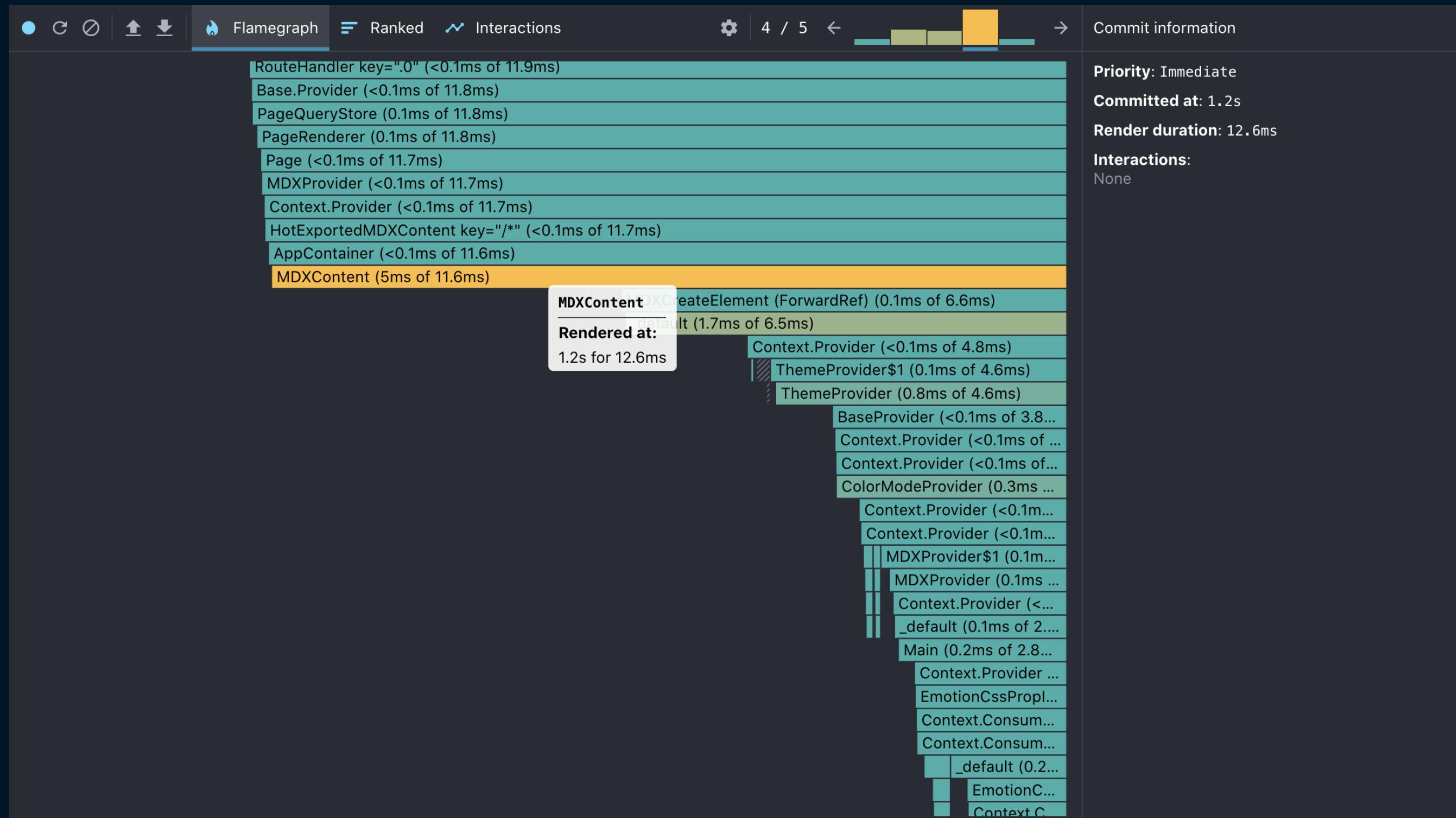
Une extension navigateur pour debugger des applications

React.js



Disponible sur **Chrome** et **Firefox**





Introducing the React Profiler

Déployer une application React.js

create-react-app est en mesure de générer pour vous
un export statique de votre application web avec la
commande `yarn build`.

```
app-name
└── build
    ├── asset-manifest.json
    ├── favicon.ico
    ├── index.html
    ├── logo192.png
    ├── logo512.png
    ├── manifest.json
    ├── precache-manifest.a3de7dc90081be00fbc342ae7cc26178.js
    ├── robots.txt
    ├── service-worker.js
    └── static
```

L'export statique se trouve dans le dossier **build** de votre projet.

Il existe 2 grands moyens de déployer une application web pour la rendre accessible :

Utiliser un serveur web avec **FTP** par exemple

Utiliser un CDN/PaaS

Nous allons utiliser **Netlify**

Netlify est une plateforme cloud permettant le déploiement "simple" d'applications web statiques.

Plan gratuit très généreux (100GB/mois)

Intégrations GitHub/BitBucket/Gitlab

Continuous deployment

Autres fonctionnalités intéressantes (Lambda functions, Forms, Auth, etc.)

Netlify: All-in-one platform for a → +

www.netlify.com

netlify Platform Pricing Enterprise Jamstack Docs Blog Contact sales Log in Sign up →

The fastest way to build the fastest sites

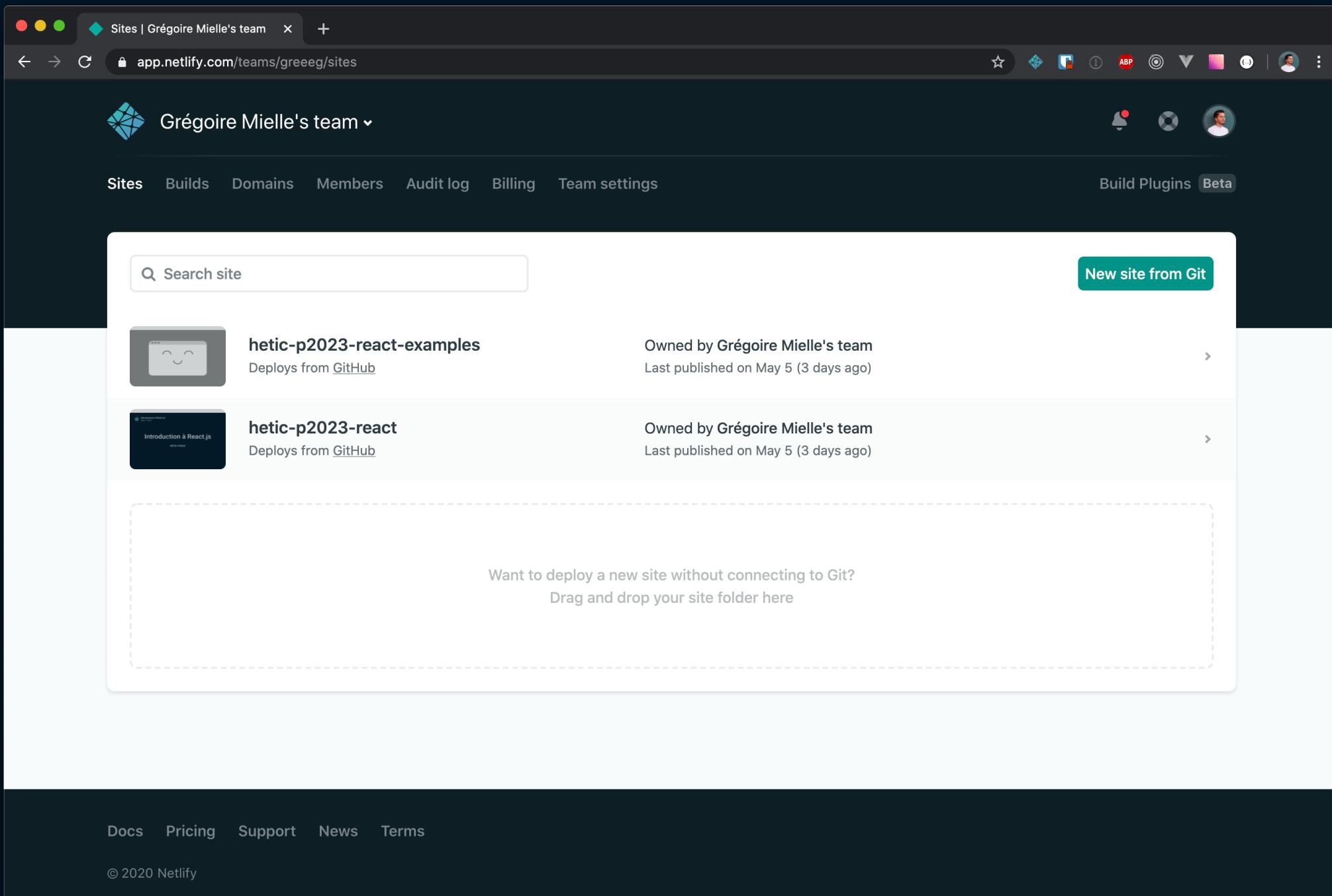
Build, test, and deploy globally with Netlify's all-in-one platform for modern web projects

Get started for free

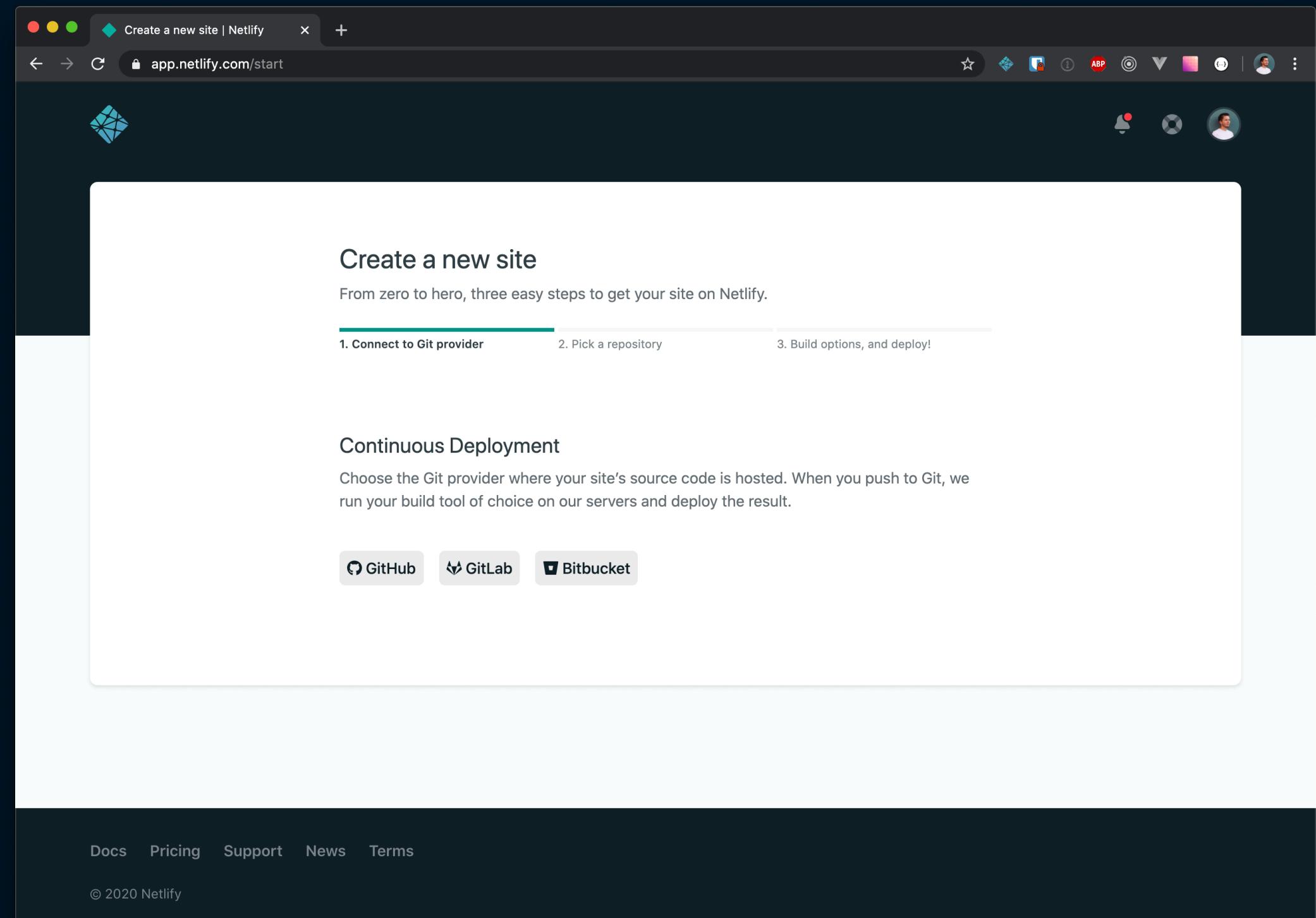
Questions? Talk to an expert →

800,000 developers and businesses trust Netlify

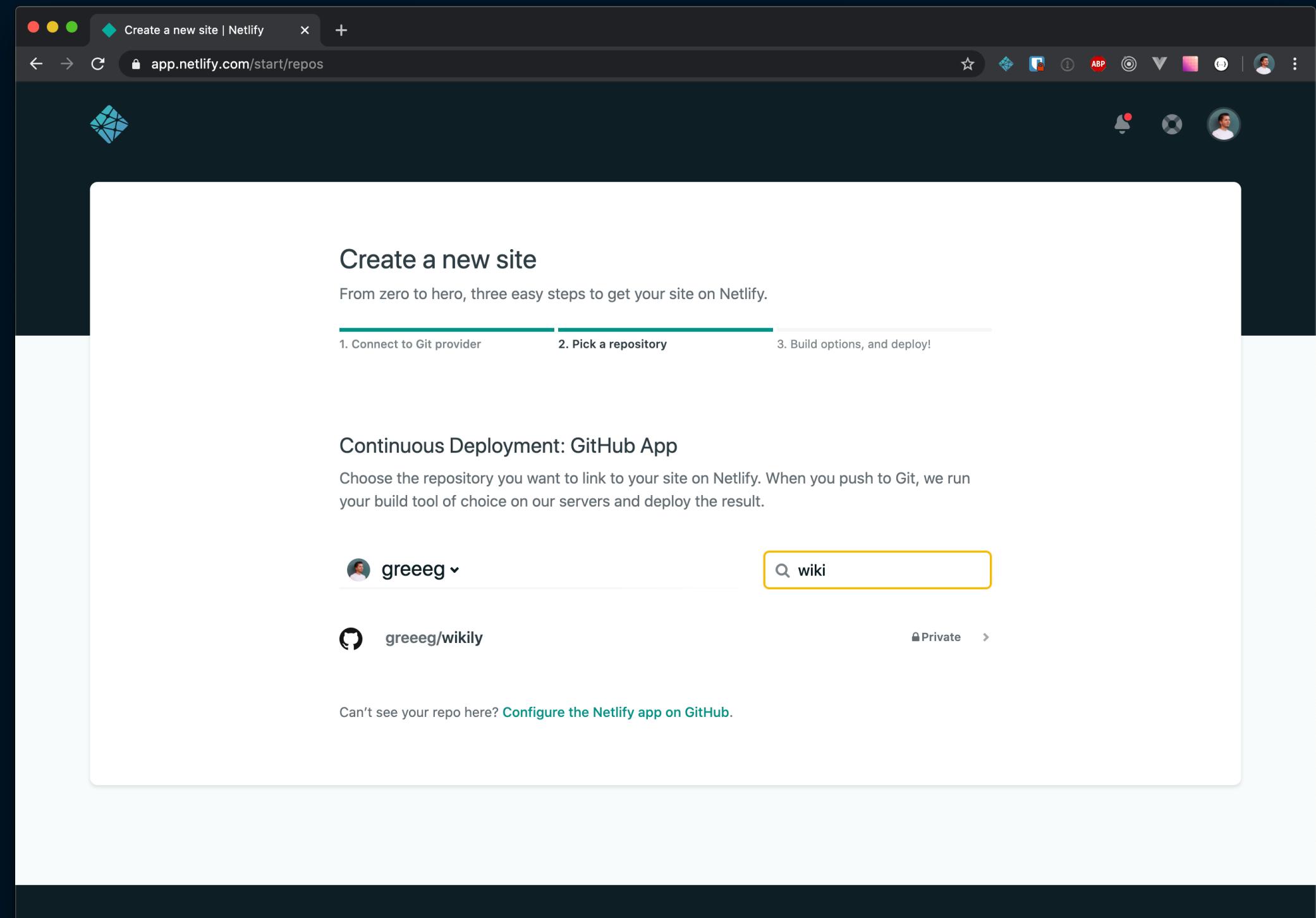
ITRIX cornerstone flexport. verizon[✓] TriNet[↗] PELOTON kubernetes



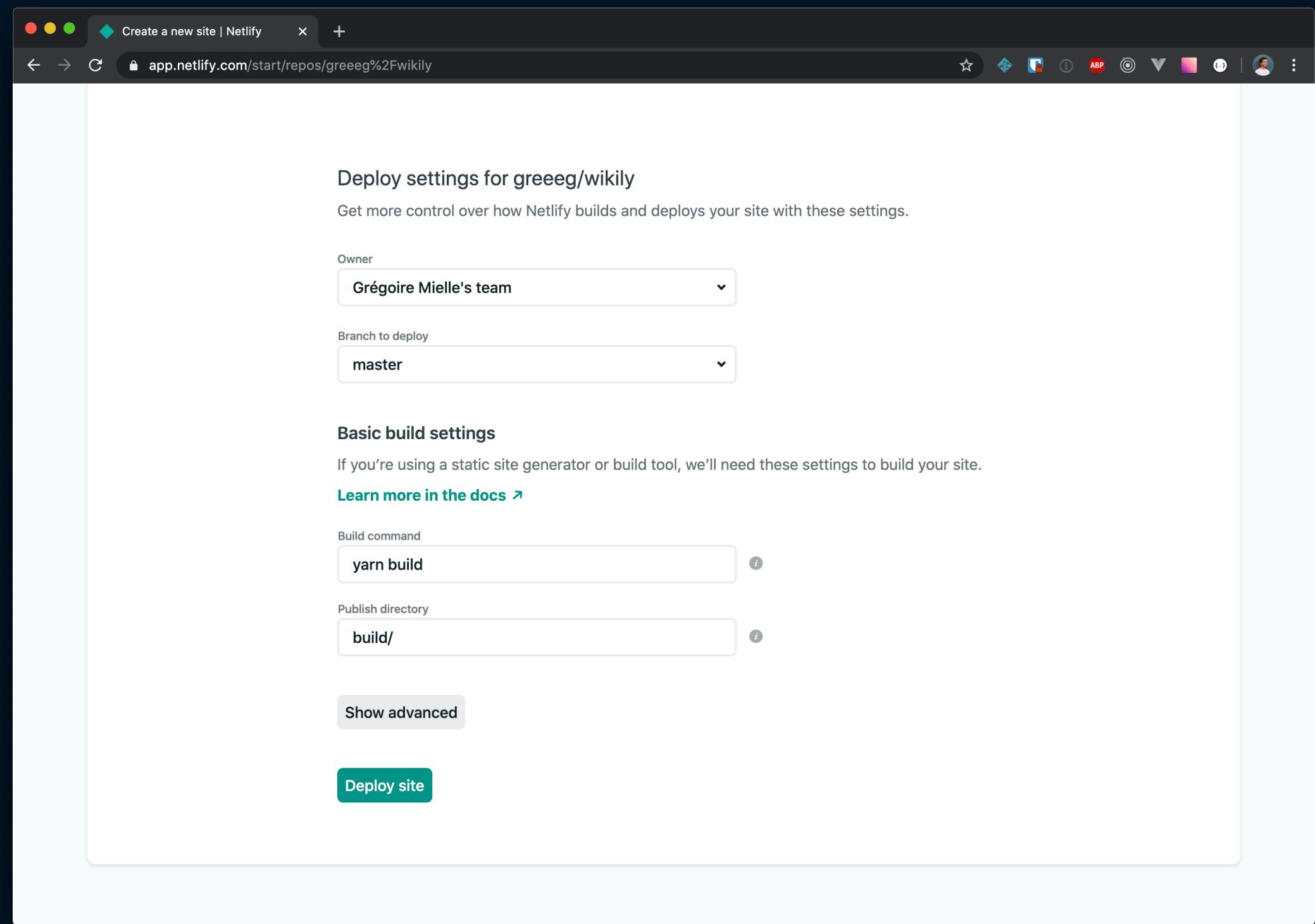
Étape 1 : Créer un site à partir d'un dossier statique ou un repository Git



Étape 2 : Avec Git, il faut effectuer une connexion OAuth



Étape 3 : Sélectionner son repository



Étape 4 : Définir le contexte de **build**

The screenshot shows the Netlify Overview page for the 'hetic-p2023-react' site. At the top, there's a navigation bar with tabs for Overview, Deployments, Functions, Identity, Forms, Large Media, Split Testing, Analytics, Settings, and Build Plugins (Beta). Below the navigation, the site title 'hetic-p2023-react' is displayed along with its URL (<https://hetic-p2023-react.netlify.app>). A preview card for the site is shown, featuring a dark blue background with white text: 'Introduction à React.js' and 'HETIC P2023'. Below the preview card, it says 'Deploys from GitHub. Last published on May 5.' and provides links for 'Site settings' and 'Domain settings'. A large 'Getting started' section follows, divided into three numbered steps: 1. 'Your site is deployed ✓' (with a note about trying a test build), 2. 'Set up a custom domain →' (with a note about buying a new domain or using an existing one), and 3. 'Secure your site with HTTPS' (with a note about automatic Let's Encrypt certificate). At the bottom of the 'Getting started' section, there are two collapsed sections: 'Production deploys' (showing a single deployment log entry) and 'Deploy Previews'.

Étape 5 : Votre site est disponible à l'URL indiquée

L'AJAX avec React.js

Asynchronous JavaScript and XML

L'AJAX permet à des interfaces web d'effectuer des actions asynchrones sans pour autant recharger la page du navigateur.

Les applications React.js mettent à jour l'interface utilisateur sans jamais recharger la page du navigateur. AJAX est donc la seule manière de mettre à jour notre interface pour :

Charger des données depuis une API (Profil utilisateur, etc.)

Envoyer des données à une API (Formulaire, Analytics, etc.)

```
const API_ENDPOINT = 'https://jsonplaceholder.typicode.com/posts';

fetch(API_ENDPOINT)
  .then((response) => {
    return response.json();
  })
  .then((json) => {
    console.log(json);
  });
}
```

On peut chaîner plusieurs 'then', notamment pour utiliser la réponse au format JSON

Et avec React.js ?

```
.then((response) => {
  return response.json();
})
.then((json) => {
  setPosts(json);
})
, []);
```



```
return (
  <div>
    <h1>The New Yorker</h1>

    <ul>
      {posts.map((post) => (
        <li key={post.id}>{post.title}</li>
      )));
    </ul>
  </div>
);
```

Et ensuite boucler à travers les posts dans le JSX pour les afficher



Nous allons réaliser une application permettant de découvrir des articles Wikipedia

Le routing

De manière générale, le routing est le processus qui fait correspondre un chemin à une ressource.

En déployant un site web composé de pages HTML statiques, vous laissez le serveur web s'occuper de faire correspondre une URL au fichier correspondant.

L'URL / affiche le fichier `index.html`

Avec React.js, peu importe le chemin demandé, on charge toujours le fichier `index.html`. C'est un package Javascript qui se chargera d'afficher les composants associés à la page demandée.

Il existe plusieurs solutions de routing avec React.js

React router

Reach router

Router 5

Nous allons utiliser React router

Le CSS-in-JS

**Le CSS-in-JS est une méthodologie visant à écrire
son style au sein de fichiers Javascript.**

Elle a été démocratisée avec le développement
d'interfaces utilisateurs orientées composants.



Max Stoiber - The Component Age & Styling React
Applications

```
<Button></Button>  
<Button primary></Button>
```

À l'échelle d'un composant, on préfère encapsuler tout son fonctionnement en un fichier

If you're using React, you have access to a more powerful styling construct than CSS class names.
You have components.

-Michael Chan

Il existe des dizaines de solutions de CSS-in-JS avec React.js

Styled-components

Stylized JSX

JSS

Glamor

Emotion

Aphrodite

Nous allons utiliser **Styled-components**

Les performances

React is slow



Toute application Javascript mal conçue peut avoir
tendance à perdre en performances.

Il s'agit alors de connaître les limites de son fonctionnement et les actions possibles pour en améliorer les performances.

Utiliser des dépendances sans en comprendre l'impact

Afficher des listes importantes de données

Build son application en un seul bundle Javascript

Re-calculer des choses inutilement

Utiliser des dépendances sans en comprendre
l'impact

Il est difficile de comprendre l'impact d'une librairie sur les performances d'une application. Sans aucun contrôle ni mesure, on peut rapidement dégrader l'expérience utilisateur de son site web.

Un bon outil pour ça : **Bundlephobia**

Afficher des listes importantes de données

Il peut être intéressant de virtualiser des listes importantes de données pour en améliorer les performances. On utilise des packages tels que `react-window` ou `react-virtualized`.

Build son application en un seul bundle Javascript

Par défaut, avec `create-react-app`, si vous utilisez la commande `yarn build`, un seul fichier JS main est généré. Il contient tous vos composants.

Si certains de vos composants utilisent de grosses dépendances, il peut être pertinent d'en générer un fichier séparé qui ne sera chargé qu'au moment voulu.

```
import React, { Suspense } from 'react';
const ThreeScene = React.lazy(() => import('./ThreeScene'));
const App = () => {
  return (
    <div>
      <header>App Name</header>
      <Suspense fallback={<div>Loading 3D Scene...</div>}>
        <ThreeScene />
      </Suspense>
    </div>
  );
}
```

Avec `React.lazy`, on indique à React que le chargement de ce composant n'est pas essentiel et peut être retardé. Un fichier JS à part sera donc généré et chargé en temps voulu.

Éviter de re-calculer des choses inutilement

Le hook `useMemo` permet de memoiser une valeur afin de ne la recalculer que si une de ses entrées change.

L'écosystème React

The React community

React.js developer roadmap

The React community

En plus de la **Core Team** qui travaille à plein temps sur React.js, il existe des **dizaines de personnes actives** dans la communauté à suivre.

En particulier

Dan Abramov

Kent C Dodds

Max Stoiber

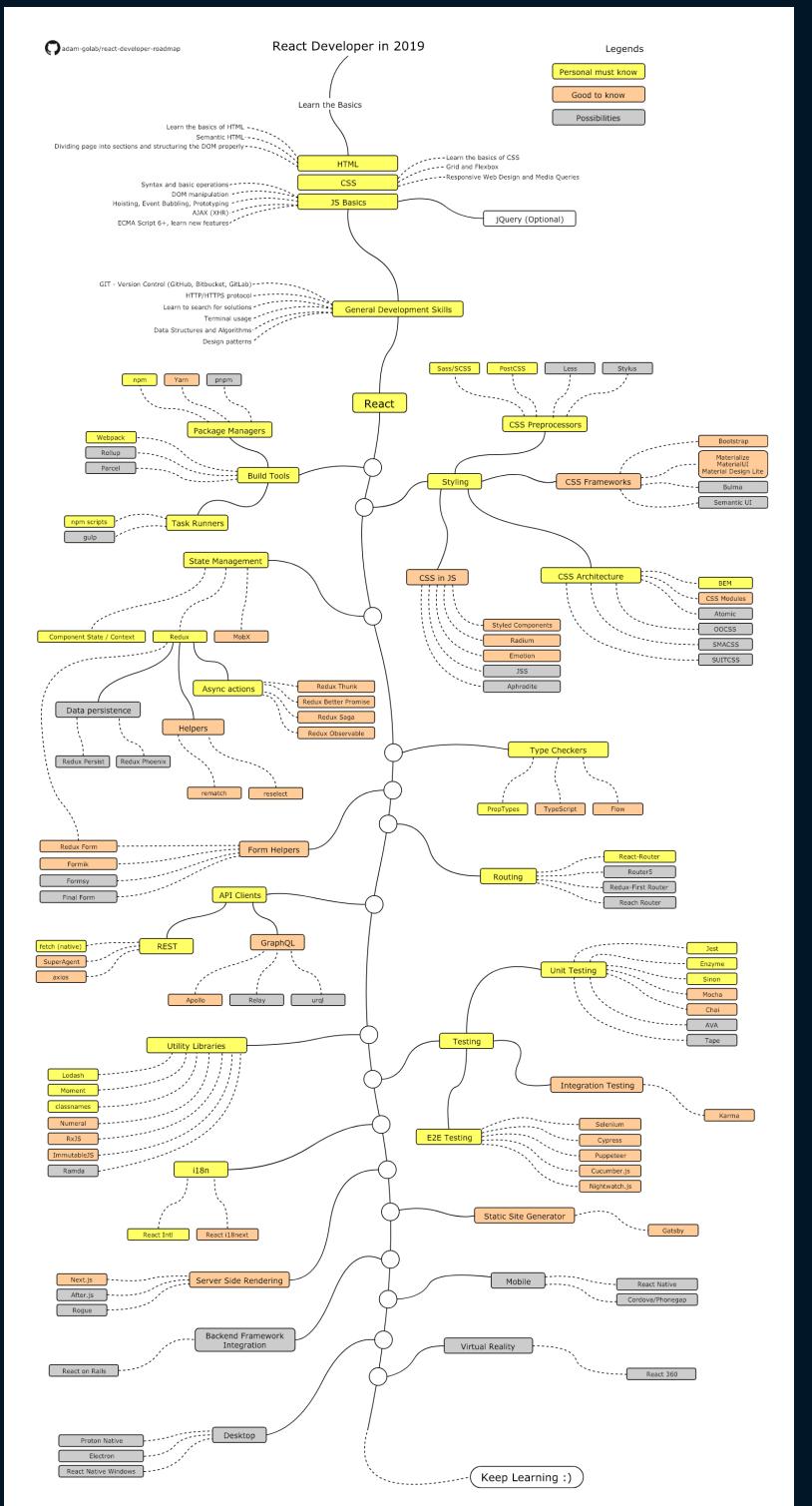
Peggy Rayzis

Wes Bos

Sara Vieira

Guillermo Rauch

React.js developer roadmap



Bonus 😊

React Native

Build native mobile apps using JavaScript and React

Native vs Hybrid vs Web App

Native apps

Elles représentent la grande majorité des applications sur vos téléphones

Avantages

Performances

Fonctionnalités natives

Inconvénients

Plusieurs codebases

Nécessite plus de développeurs

Hybrid apps

Une application web encapsulée dans une application native

Avantages

Temps de développement
HTML, CSS, JS

Inconvénients

Performances

Expérience utilisateur

Web apps (PWA)

Une application web avec des capacités mobiles

Avantages

Rien à installer

Fonctionnalités natives

Performances

Temps de développement

Inconvénients

Absent des stores

Compatibilité limitée

Et React Native au milieu de tout ça ?

Faire du natif lorsque l'on vient du web

1 codebase : 2 plateformes

Aller vite

On essaye ?

reactnative.dev

🔊 Twitter: @greeeg

🔬 GitHub: @greeeg

✉️ Email: hello@greeeg.com