### Introduction

[React](https://reactjs.org/) est un framework JavaScript populaire pour créer des applications frontales. Créé à l'origine par Facebook, il a gagné en popularité en permettant aux développeurs de créer des applications rapides à l'aide d'un paradigme de programmation intuitif qui lie JavaScript à une syntaxe de type HTML connue sous le nom de [JSX](https://reactjs.org/docs/introducing-jsx.html) .

Le démarrage d'un nouveau projet React était auparavant un processus complexe en plusieurs étapes qui impliquait la mise en place d'un système de construction, un transpileur de code pour convertir la syntaxe moderne en code lisible par tous les navigateurs et une structure de répertoire de base. Mais maintenant, [Create React App](https://github.com/facebook/create-react-app) inclut tous les packages JavaScript dont vous avez besoin pour exécuter un projet React, y compris la transpilation de code, le lintage de base, les tests et les systèmes de construction. Il comprend également un serveur avec rechargement à chaud qui actualisera votre page au fur et à mesure que vous apporterez des modifications au code. Enfin, il créera une structure pour vos répertoires et composants afin que vous puissiez vous lancer et commencer à coder en quelques minutes seulement.

En d'autres termes, vous n'avez pas à vous soucier de la configuration d'un système de construction comme [Webpack](https://www.digitalocean.com/community/tutorials/whats-new-in-webpack-4) . Vous n'avez pas besoin de configurer [Babel](https://babeljs.io/) pour transpiler votre code afin qu'il soit utilisable par tous les navigateurs. Vous n'avez pas à vous soucier de la plupart des systèmes complexes du développement frontal moderne. Vous pouvez commencer à écrire du code React avec un minimum de préparation.

À la fin de ce didacticiel, vous disposerez d'une application React en cours d'exécution que vous pourrez utiliser comme base pour toutes les applications futures. Vous apporterez vos premières modifications au code React, mettrez à jour les styles et exécuterez une version pour créer une version entièrement minifiée de votre application. Vous utiliserez également un serveur avec rechargement à chaud pour vous donner un retour instantané et explorerez en profondeur les parties d'un projet React. Enfin, vous commencerez à écrire des composants personnalisés et à créer une structure qui peut évoluer et s'adapter à votre projet.

## Conditions préalables

Pour suivre ce tutoriel, vous aurez besoin des éléments suivants :

* [Node.js](https://nodejs.org/) version 10.16.0 installé sur votre ordinateur. Pour l'installer sur macOS ou Ubuntu 18.04, suivez les étapes de [Comment installer Node.js et créer un environnement de développement local sur macOS](https://www.digitalocean.com/community/tutorials/how-to-install-node-js-and-create-a-local-development-environment-on-macos) ou la section **Installation à l'aide d'un PPA** de [Comment installer Node.js sur Ubuntu 18.04](https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-18-04) .
* Cela vous aidera également à avoir une compréhension de base de JavaScript, que vous pouvez trouver dans la [série Comment coder en JavaScript](https://www.digitalocean.com/community/tutorial_series/how-to-code-in-javascript) , ainsi qu'une connaissance de base de HTML et CSS.

## Étape 1 - Création d'un nouveau projet avec Create React App

Dans cette étape, vous allez créer une nouvelle application à l'aide du gestionnaire de packages [npm](https://www.npmjs.com/) pour exécuter un script distant. Le script copiera les fichiers nécessaires dans un nouveau répertoire et installera toutes les dépendances.

Lorsque vous avez installé Node, vous avez également installé une application de gestion de packages appelée [npm](https://www.npmjs.com/). npminstallera les packages JavaScript dans votre projet et gardera également une trace des détails du projet. Si vous souhaitez en savoir plus sur npm, consultez notre [didacticiel Comment utiliser les modules Node.js avec npm et package.json](https://www.digitalocean.com/community/tutorials/how-to-use-node-js-modules-with-npm-and-package-json) .

npmcomprend également un outil appelé [npx](https://www.npmjs.com/package/npx), qui exécutera des packages exécutables. Cela signifie que vous exécuterez le code Create React App sans télécharger au préalable le projet.

Le package exécutable exécutera l'installation de create-react-appdans le répertoire que vous spécifiez. Il commencera par créer un nouveau projet dans un répertoire, qui dans ce tutoriel sera appelé digital-ocean-tutorial. Encore une fois, ce répertoire n'a pas besoin d'exister au préalable ; le package exécutable le créera pour vous. Le script s'exécutera également npm installdans le répertoire du projet, qui téléchargera toutes les dépendances supplémentaires.

Pour installer le projet de base, exécutez la commande suivante :

1. npx create-react-app digital-ocean-tutorial

Copie

Cette commande lancera un processus de construction qui téléchargera le code de base avec un certain nombre de dépendances.

Une fois le script terminé, vous verrez un message de réussite indiquant :

Output

...

Success! Created digital-ocean-tutorial at your\_file\_path/digital-ocean-tutorial

Inside that directory, you can run several commands:

npm start

Starts the development server.

npm run build

Bundles the app into static files for production.

npm test

Starts the test runner.

npm run eject

Removes this tool and copies build dependencies, configuration files

and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

cd digital-ocean-tutorial

npm start

Happy hacking!

your\_file\_pathsera votre chemin actuel. Si vous êtes un utilisateur macOS, ce sera quelque chose comme ; si vous êtes sur un serveur Ubuntu, il dira quelque chose comme ./Users/your\_username/home/your\_username

Vous verrez également une liste de npmcommandes qui vous permettront d'exécuter, de construire, de démarrer et de tester votre application. Vous les explorerez davantage dans la section suivante.

**Remarque :** Il existe un autre gestionnaire de packages pour JavaScript appelé [yarn](https://yarnpkg.com/). Il est pris en charge par Facebook et fait la plupart des mêmes choses que npm. À l'origine, yarnfournissait de nouvelles fonctionnalités telles que les fichiers de verrouillage, mais elles sont désormais npmégalement implémentées dans. yarninclut également quelques autres fonctionnalités telles que la mise en cache hors ligne. D' autres différences peuvent être trouvées dans la [yarndocumentation](https://yarnpkg.com/features/pnp) .

Si vous avez déjà installé yarnsur votre système, vous verrez une liste de yarncommandes telles que celles yarn startqui fonctionnent de la même manière que les npmcommandes. Vous pouvez exécuter des npmcommandes même si vous avez yarninstallé. Si vous préférez yarn, remplacez simplement npmpar yarndans toutes les commandes futures. Les résultats seront les mêmes.

Votre projet est maintenant configuré dans un nouveau répertoire. Changez dans le nouveau répertoire :

1. cd digital-ocean-tutorial

Copie

Vous êtes maintenant à la racine de votre projet. À ce stade, vous avez créé un nouveau projet et ajouté toutes les dépendances. Mais vous n'avez effectué aucune action pour exécuter le projet. Dans la section suivante, vous exécuterez des scripts personnalisés pour générer et tester le projet.

## Étape 2 - Utilisationreact-scripts

Dans cette étape, vous découvrirez les différents react-scriptsqui sont installés avec le dépôt. Vous exécuterez d'abord le testscript pour exécuter le code de test. Ensuite, vous exécuterez le buildscript pour créer une version minifiée. Enfin, vous verrez comment le ejectscript peut vous donner un contrôle total sur la personnalisation.

Maintenant que vous êtes dans le répertoire du projet, jetez un coup d'œil. Vous pouvez soit ouvrir tout le répertoire dans votre éditeur de texte, soit si vous êtes sur le terminal, vous pouvez lister les fichiers avec la commande suivante :

1. ls -a

Copie

L' -aindicateur garantit que la sortie inclut également les fichiers cachés.

Dans tous les cas, vous verrez une structure comme celle-ci :

Output

node\_modules/

public/

src/

.gitignore

README.md

package-lock.json

package.json

Expliquons-les un par un :

* node\_modules/contient toutes les bibliothèques JavaScript externes utilisées par l'application. Vous aurez rarement besoin de l'ouvrir.
* Le public/répertoire contient des fichiers HTML, [JSON](https://www.digitalocean.com/community/tutorials/an-introduction-to-json) et image de base. Ce sont les racines de votre projet. Vous aurez l'occasion de les explorer davantage à l' [étape 4](https://www.digitalocean.com/community/tutorials/how-to-set-up-a-react-project-with-create-react-app#step-4-%E2%80%93-modifying-the-homepage) .
* Le src/répertoire contient le code JavaScript React pour votre projet. La plupart du travail que vous ferez sera dans ce répertoire. Vous explorerez ce répertoire en détail à l' [étape 5](https://www.digitalocean.com/community/tutorials/how-to-set-up-a-react-project-with-create-react-app#step-5-%E2%80%93-modiifying-the-heading-tag-and-styling) .
* Le .gitignorefichier contient des répertoires et des fichiers par défaut que [git](https://www.digitalocean.com/community/tutorial_series/introduction-to-git-installation-usage-and-branches) — votre contrôle de code source — ignorera, comme le node\_modulesrépertoire. Les éléments ignorés ont tendance à être des répertoires ou des fichiers journaux plus volumineux dont vous n'auriez pas besoin dans le contrôle de code source. Il inclura également certains répertoires que vous créerez avec certains des scripts React.
* README.mdest un fichier Markdown qui contient de nombreuses informations utiles sur Create React App, telles qu'un résumé des commandes et des liens vers la configuration avancée. Pour l'instant, il est préférable de laisser le README.mdfichier tel que vous le voyez. Au fur et à mesure de l'avancement de votre projet, vous remplacerez les informations par défaut par des informations plus détaillées sur votre projet.

Les deux derniers fichiers sont utilisés par votre gestionnaire de packages. Lorsque vous avez exécuté la commande initiale npx, vous avez créé le projet de base, mais vous avez également installé les dépendances supplémentaires. Lorsque vous avez installé les dépendances, vous avez créé un package-lock.jsonfichier. Ce fichier est utilisé par npmpour s'assurer que les packages correspondent aux versions exactes. De cette façon, si quelqu'un d'autre installe votre projet, vous pouvez vous assurer qu'il a des dépendances identiques. Étant donné que ce fichier est créé automatiquement, vous modifierez rarement ce fichier directement.

Le dernier fichier est un [package.json](https://www.digitalocean.com/community/tutorials/how-to-use-node-js-modules-with-npm-and-package-json" \l "step-1-%E2%80%94-creating-a-packagejson-file). Celui-ci contient des métadonnées sur votre projet, telles que le titre, le numéro de version et les dépendances. Il contient également des scripts que vous pouvez utiliser pour exécuter votre projet.

Ouvrez le package.jsonfichier dans votre éditeur de texte préféré :

1. nano package.json

Copie

Lorsque vous ouvrez le fichier, vous verrez un objet JSON contenant toutes les métadonnées. Si vous regardez l' scriptsobjet, vous trouverez quatre scripts différents : start, build, testet eject.

Ces scripts sont classés par ordre d'importance. Le premier script démarre l'environnement de développement local ; vous y arriverez à l'étape suivante. Le deuxième script construira votre projet. Vous explorerez cela en détail à l' [étape 4](https://www.digitalocean.com/community/tutorials/how-to-set-up-a-react-project-with-create-react-app#step-4-%E2%80%93-modifying-the-homepage) , mais cela vaut la peine de courir maintenant pour voir ce qui se passe.

### Le buildScénario

Pour exécuter n'importe quel script npm, il vous suffit de taper dans votre terminal. Il existe quelques scripts spéciaux dans lesquels vous pouvez omettre une partie de la commande, mais il est toujours possible d'exécuter la commande complète. Pour exécuter le script, saisissez ce qui suit dans votre terminal :npm run script\_namerunbuild

1. npm run build

Copienpm

Vous verrez immédiatement le message suivant :

Output

> digital-ocean-tutorial@0.1.0 build your\_file\_path/digital-ocean-tutorial

> react-scripts build

Creating an optimized production build...

...

reactCela vous indique que Create React App compile votre code dans un bundle utilisable.

Une fois terminé, vous verrez le résultat suivant :

Output

...

Compiled successfully.

File sizes after gzip:

39.85 KB build/static/js/9999.chunk.js

780 B build/static/js/runtime-main.99999.js

616 B build/static/js/main.9999.chunk.js

556 B build/static/css/main.9999.chunk.css

The project was built assuming it is hosted at the server root.

You can control this with the homepage field in your package.json.

For example, add this to build it for GitHub Pages:

"homepage" : "http://myname.github.io/myapp",

The build folder is ready to be deployed.

You may serve it with a static server:

serve -s build

Find out more about deployment here:

bit.ly/CRA-deploy

Listez le contenu du projet et vous verrez de nouveaux répertoires :

1. ls -a

Copie

Output

build/

node\_modules/

public/

src/

.gitignore

README.md

package-lock.json

package.json

Vous avez maintenant un buildrépertoire. Si vous avez ouvert le .gitignorefichier, vous avez peut-être remarqué que le buildrépertoire est ignoré par git. C'est parce que le buildrépertoire n'est qu'une version minifiée et optimisée des autres fichiers. Il n'est pas nécessaire d'utiliser le contrôle de version puisque vous pouvez toujours exécuter la buildcommande. Vous explorerez la sortie plus tard ; pour l'instant, il est temps de passer au testscript.

### Le testScénario

Le testscript est l'un de ces scripts spéciaux qui ne nécessite pas le mot- runclé, mais fonctionne même si vous l'incluez. Ce script lancera un lanceur de test appelé [Jest](https://jestjs.io/) . L'exécuteur de test recherche dans votre projet tous les fichiers avec une extension .spec.jsou .test.js, puis exécute ces fichiers.

Pour exécuter le testscript, saisissez la commande suivante :

1. npm test

Copie

Après avoir exécuté ce script, votre terminal aura la sortie de la suite de tests et l'invite du terminal disparaîtra. Cela ressemblera à ceci :

Output

PASS src/App.test.js

✓ renders learn react link (67ms)

Test Suites: 1 passed, 1 total

Tests: 1 passed, 1 total

Snapshots: 0 total

Time: 4.204s

Ran all test suites.

Watch Usage

› Press f to run only failed tests.

› Press o to only run tests related to changed files.

› Press q to quit watch mode.

› Press p to filter by a filename regex pattern.

› Press t to filter by a test name regex pattern.

› Press Enter to trigger a test run.

Il y a quelques choses à remarquer ici. Tout d'abord, comme indiqué précédemment, il détecte automatiquement tous les fichiers avec des extensions de test, y compris .test.jset .spec.js. Dans ce cas, il n'y a qu'une seule suite de tests — c'est-à-dire un seul fichier avec une .test.jsextension — et cette suite de tests ne contient qu'un seul test. Jest peut détecter des tests dans votre hiérarchie de code, vous pouvez donc imbriquer des tests dans un répertoire et Jest les trouvera.

Deuxièmement, Jest n'exécute pas votre suite de tests une seule fois, puis quitte. Au contraire, il continue de fonctionner dans le terminal. Si vous apportez des modifications au code source, il relancera les tests à nouveau.

Vous pouvez également limiter les tests que vous exécutez en utilisant l'une des options du clavier. Si vous tapez o, par exemple, vous n'exécuterez les tests que sur les fichiers qui ont changé. Cela peut vous faire gagner beaucoup de temps à mesure que vos suites de tests se développent.

Enfin, vous pouvez quitter le testeur en tapant q. Faites-le maintenant pour retrouver votre invite de commande.

### Le ejectScénario

Le script final est npm eject. Ce script copie vos dépendances et vos fichiers de configuration dans votre projet, vous donnant un contrôle total sur votre code mais éjectant le projet de la chaîne d'outils intégrée Create React App. Vous ne l'exécuterez pas maintenant car, une fois que vous aurez exécuté ce script, vous ne pourrez plus annuler cette action et vous perdrez toutes les futures mises à jour de l'application Create React.

La valeur dans Create React App est que vous n'avez pas à vous soucier d'une quantité importante de configuration. La création d'applications JavaScript modernes nécessite de nombreux outils, des systèmes de construction, tels que [Webpack](https://www.digitalocean.com/community/tutorials/whats-new-in-webpack-4) , aux outils de compilation, tels que [Babel](https://babeljs.io/) . Create React App gère toute la configuration pour vous, donc éjecter signifie gérer vous-même cette complexité.

L'inconvénient de Create React App est que vous ne pourrez pas entièrement personnaliser le projet. Pour la plupart des projets, ce n'est pas un problème, mais si jamais vous voulez prendre le contrôle de tous les aspects du processus de construction, vous devrez éjecter le code. Cependant, comme mentionné précédemment, une fois que vous avez éjecté le code, vous ne pourrez pas mettre à jour les nouvelles versions de Create React App, et vous devrez ajouter manuellement toutes les améliorations par vous-même.

À ce stade, vous avez exécuté des scripts pour créer et tester votre code. À l'étape suivante, vous démarrerez le projet sur un serveur en direct.

## Étape 3 - Démarrage du serveur

Dans cette étape, vous allez initialiser un serveur local et exécuter le projet dans votre navigateur.

Vous démarrez votre projet avec un autre npmscript. Comme npm test, ce script n'a pas besoin de la runcommande. Lorsque vous exécutez le script, vous démarrez un serveur local, exécutez le code du projet, démarrez un observateur qui écoute les modifications de code et ouvrez le projet dans un navigateur Web.

Démarrez le projet en tapant la commande suivante à la racine de votre projet. Pour ce tutoriel, la racine de votre projet est le digital-ocean-tutorialrépertoire. Assurez-vous de l'ouvrir dans un terminal ou un onglet séparé, car ce script continuera à s'exécuter tant que vous l'autoriserez :

1. npm start

Copie

Vous verrez un texte d'espace réservé pendant un bref instant avant le démarrage du serveur, donnant cette sortie :

Output

Compiled successfully!

You can now view digital-ocean-tutorial in the browser.

http://localhost:3000

Note that the development build is not optimized.

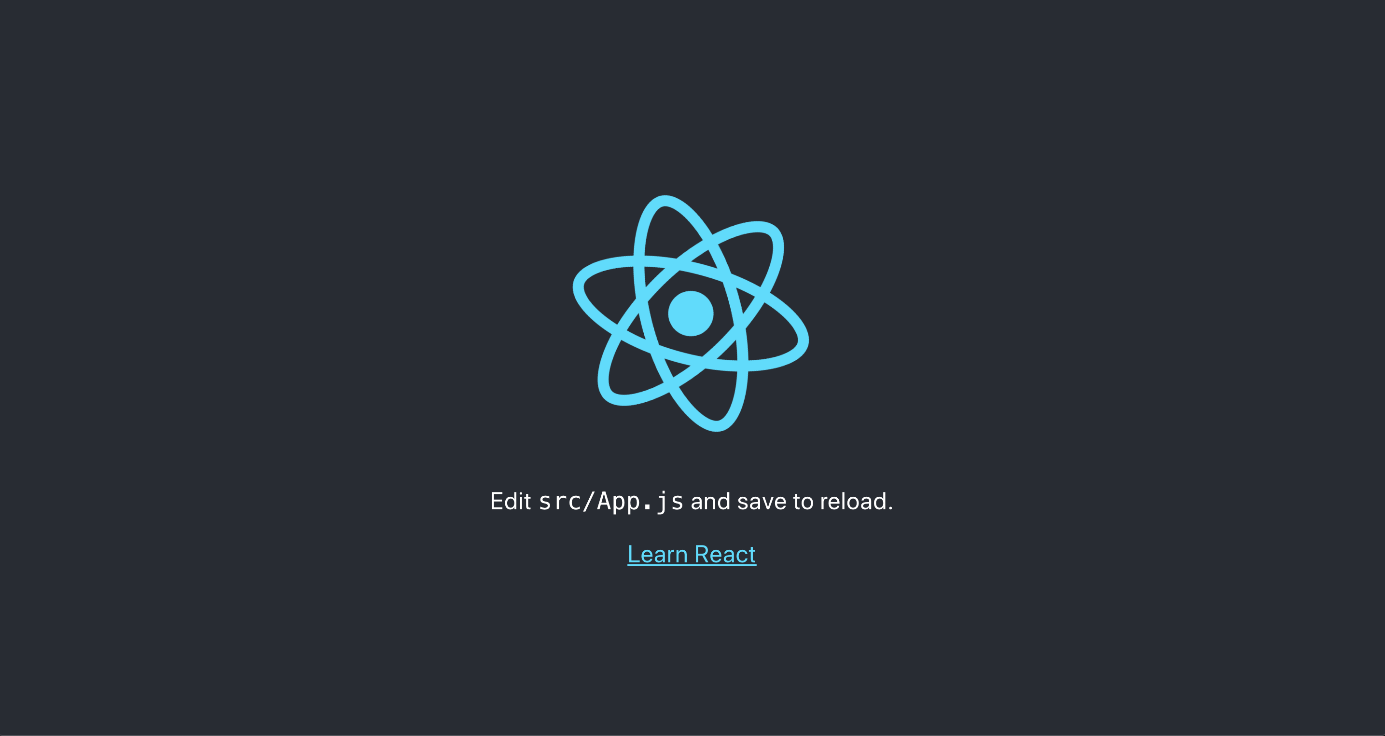
To create a production build, use npm run build.

Si vous exécutez le script localement, il ouvrira le projet dans la fenêtre de votre navigateur et déplacera le focus du terminal vers le navigateur.

Si cela ne se produit pas, vous pouvez visiter <http://localhost:3000/>pour voir le site en action. Si vous avez déjà un autre serveur en cours d'exécution sur le port 3000, c'est bien. Create React App détectera le prochain port disponible et exécutera le serveur avec celui-ci. En d'autres termes, si vous avez déjà un projet en cours d'exécution sur le port 3000, ce nouveau projet démarrera sur le port 3001.

Si vous l'exécutez à partir d'un serveur distant, vous pouvez toujours voir votre site sans aucune configuration supplémentaire. L'adresse sera . Si vous avez configuré un pare-feu, vous devrez [ouvrir le port](https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands) sur votre serveur distant.http://your\_server\_ip:3000

Dans le navigateur, vous verrez le projet de modèle React suivant :



Tant que le script est en cours d'exécution, vous aurez un serveur local actif. Pour arrêter le script, fermez la fenêtre ou l'onglet du terminal ou tapez CTRL+Cou ⌘-+cdans la fenêtre ou l'onglet du terminal qui exécute votre script.

À ce stade, vous avez démarré le serveur et exécutez votre premier code React. Mais avant d'apporter des modifications au code JavaScript React, vous verrez comment React s'affiche sur la page en premier lieu.

## Étape 4 - Modification de la page d'accueil

Dans cette étape, vous allez modifier le code dans le public/répertoire. Le publicrépertoire contient votre page HTML de base. C'est la page qui servira de racine à votre projet. Vous modifierez rarement ce répertoire à l'avenir, mais c'est la base à partir de laquelle le projet démarre et une partie cruciale d'un projet React.

Si vous avez annulé votre serveur, continuez et redémarrez-le avec npm start, puis ouvrez public/-le dans votre éditeur de texte préféré dans une nouvelle fenêtre de terminal :

1. nano public/

Copie

Alternativement, vous pouvez lister les fichiers avec la lscommande :

1. ls public/

Copie

Vous verrez une liste de fichiers comme celle-ci :

Output

favicon.ico

logo192.png

manifest.json

index.html

logo512.png

robots.txt

favicon.ico, logo192.png, et logo512.pngsont des icônes qu'un utilisateur verra soit dans l'onglet de son navigateur, soit sur son téléphone. Le navigateur sélectionnera les icônes de taille appropriée. Finalement, vous voudrez les remplacer par des icônes plus adaptées à votre projet. Pour l'instant, vous pouvez les laisser tranquilles.

Il manifest.jsons'agit d'un ensemble structuré de [métadonnées](https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json) décrivant votre projet. Entre autres choses, il répertorie quelle icône sera utilisée pour différentes options de taille.

Le robots.txtfichier contient des informations pour [les robots d'exploration Web](https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt) . Il indique aux crawlers quelles pages ils sont ou ne sont pas autorisés à indexer. Vous n'aurez pas besoin de modifier l'un ou l'autre des fichiers à moins qu'il y ait une raison impérieuse de le faire. Par exemple, si vous souhaitez donner à certains utilisateurs une URL vers un contenu spécial que vous ne voulez pas facilement accessible, vous pouvez l'ajouter robots.txtet il sera toujours accessible au public, mais non indexé par les moteurs de recherche.

Le index.htmlfichier est la racine de votre application. C'est le fichier que le serveur lit, et c'est le fichier que votre navigateur affichera. Ouvrez-le dans votre éditeur de texte et jetez un œil.

Si vous travaillez depuis la ligne de commande, vous pouvez l'ouvrir avec la commande suivante :

1. nano public/index.html

Copie

Voici ce que vous verrez :

didacticiel-digital-ocean/public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta

name="description"

content="Web site created using create-react-app"

/>

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<!--

manifest.json provides metadata used when your web app is installed on a

user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

-->

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<!--

Notice the use of %PUBLIC\_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

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

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

Copie

Le dossier est assez court. Il n'y a pas d'images ou de mots dans le fichier <body>. C'est parce que React construit lui-même toute la structure HTML et l'injecte avec JavaScript. Mais React a besoin de savoir où injecter le code, et c'est le rôle de index.html.

Dans votre éditeur de texte, changez la <title>balise de React Appà Sandbox:

didacticiel-digital-ocean/public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

...

<title>Sandbox</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

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

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

Copie

Enregistrez et quittez votre éditeur de texte. Vérifiez votre navigateur. Le titre est le nom situé sur l'onglet du navigateur. Il se mettra à jour automatiquement. Si ce n'est pas le cas, actualisez la page et notez le changement.

Retournez maintenant à votre éditeur de texte. Chaque projet React commence à partir d'un élément racine. Il peut y avoir plusieurs éléments racine sur une page, mais il doit y en avoir au moins un. C'est ainsi que React sait où placer le code HTML généré. Trouvez l'élément <div id="root">. C'est ce divque React utilisera pour toutes les futures mises à jour. Changez le idde rootà base:

didacticiel-digital-ocean/public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

...

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="base"></div>

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

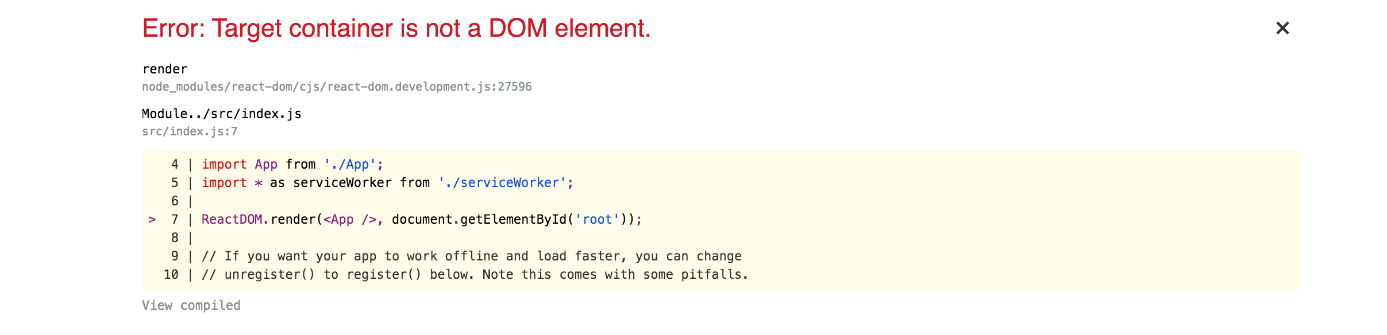
</body>

</html>

Copie

Enregistrez les modifications.

Vous verrez une erreur dans votre navigateur :



React recherchait un élément avec un idof root. Maintenant qu'il est parti, React ne peut pas démarrer le projet.

Changez le nom de baseà root:

didacticiel-digital-ocean/public/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

...

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

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

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

Copie

Enregistrez et quittez index.html.

À ce stade, vous avez démarré le serveur et apporté une petite modification à la page HTML racine. Vous n'avez pas encore modifié de code JavaScript. Dans la section suivante, vous mettrez à jour le code JavaScript de React.

## Étape 5 - Modification de la balise de titre et du style

Dans cette étape, vous apporterez votre première modification à un composant React dans le src/répertoire. Vous apporterez une petite modification au CSS et au code JavaScript qui seront automatiquement mis à jour dans votre navigateur à l'aide du rechargement à chaud intégré.

Si vous avez arrêté le serveur, assurez-vous de le redémarrer avec npm start. Maintenant, prenez le temps de voir les parties du src/répertoire. Vous pouvez soit ouvrir le répertoire complet dans votre éditeur de texte préféré, soit lister le projet dans un terminal avec la commande suivante :

1. ls src/

Copie

Vous verrez les fichiers suivants dans votre terminal ou votre éditeur de texte.

Output

App.css

App.js

App.test.js

index.css

index.js

logo.svg

serviceWorker.js

setupTests.js

Passons en revue ces fichiers un par un.

Vous ne passerez pas beaucoup de temps avec le serviceWorker.jsfichier au début, mais cela peut être important lorsque vous commencez à créer [des applications Web progressives](https://developers.google.com/web/progressive-web-apps) . Le technicien de service peut faire beaucoup de choses, y compris les notifications push et la mise en cache hors ligne, mais pour l'instant, il est préférable de le laisser tranquille.

Les prochains fichiers à examiner sont setupTests.jset App.test.js. Ceux-ci sont utilisés pour les fichiers de test. En fait, lorsque vous avez exécuté npm testl'étape 2, le script a exécuté ces fichiers. Le setupTests.jsdossier est court ; tout ce qu'il comprend est quelques expectméthodes personnalisées. Vous en apprendrez plus à ce sujet dans les prochains didacticiels de cette série.

Ouvert App.test.js:

1. nano src/App.test.js

Copie

Lorsque vous l'ouvrirez, vous verrez un test de base :

digital-ocean-tutorial/src/App.test.js

import React from 'react';

import { render } from '@testing-library/react';

import App from './App';

test('renders learn react link', () => {

const { getByText } = render(<App />);

const linkElement = getByText(/learn react/i);

expect(linkElement).toBeInTheDocument();

});

Copie

Le test recherche la phrase learn reactà inclure dans le document. Si vous revenez au navigateur exécutant votre projet, vous verrez la phrase sur la page. Les tests React sont différents de la plupart [des tests unitaires](https://en.wikipedia.org/wiki/Unit_testing) . Étant donné que les composants peuvent inclure des informations visuelles, telles que le balisage, ainsi qu'une logique de manipulation des données, les tests unitaires traditionnels ne fonctionnent pas aussi facilement. Les tests React sont plus proches d'une forme de [tests fonctionnels ou d'intégration](https://en.wikipedia.org/wiki/Integration_testing) .

Ensuite, vous verrez quelques fichiers de style : App.css, index.csset logo.svg. Il existe plusieurs façons de travailler avec le style dans React, mais la plus simple consiste à écrire du CSS simple car cela ne nécessite aucune configuration supplémentaire.

Il existe plusieurs fichiers CSS car vous pouvez importer les styles dans un composant comme s'il s'agissait d'un autre fichier JavaScript. Puisque vous avez le pouvoir d'importer du CSS directement dans un composant, vous pouvez aussi diviser le CSS pour ne l'appliquer qu'à un composant individuel. Ce que vous faites, c'est séparer les préoccupations. Vous ne gardez pas tout le CSS séparé du JavaScript. Au lieu de cela, vous conservez tous les CSS, JavaScript, balisage et images associés regroupés.

Ouvrez App.cssdans votre éditeur de texte. Si vous travaillez depuis la ligne de commande, vous pouvez l'ouvrir avec la commande suivante :

1. nano src/App.css

Copie

Voici le code que vous verrez :

digital-ocean-tutoriel/src/App.css

.App {

text-align: center;

}

.App-logo {

height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.App-logo {

animation: App-logo-spin infinite 20s linear;

}

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.App-link {

color: #61dafb;

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

Copie

Il s'agit d'un fichier CSS standard sans préprocesseurs CSS spéciaux. Vous pouvez les ajouter plus tard si vous le souhaitez, mais au début, vous n'avez que du CSS simple. Create React App essaie de ne pas avoir d'opinion tout en offrant un environnement prêt à l'emploi.

De retour à App.css, l'un des avantages de l'utilisation de Create React App est qu'elle surveille tous les fichiers, donc si vous apportez une modification, vous la verrez dans votre navigateur sans recharger.

Pour voir cela en action, apportez une petite modification au fichier background-colorin App.css. Changez-le de #282c34puis blueenregistrez le fichier. Le style final ressemblera à ceci :

digital-ocean-tutoriel/src/App.css

.App {

text-align: center;

}

...

.App-header {

background-color: blue

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

...

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

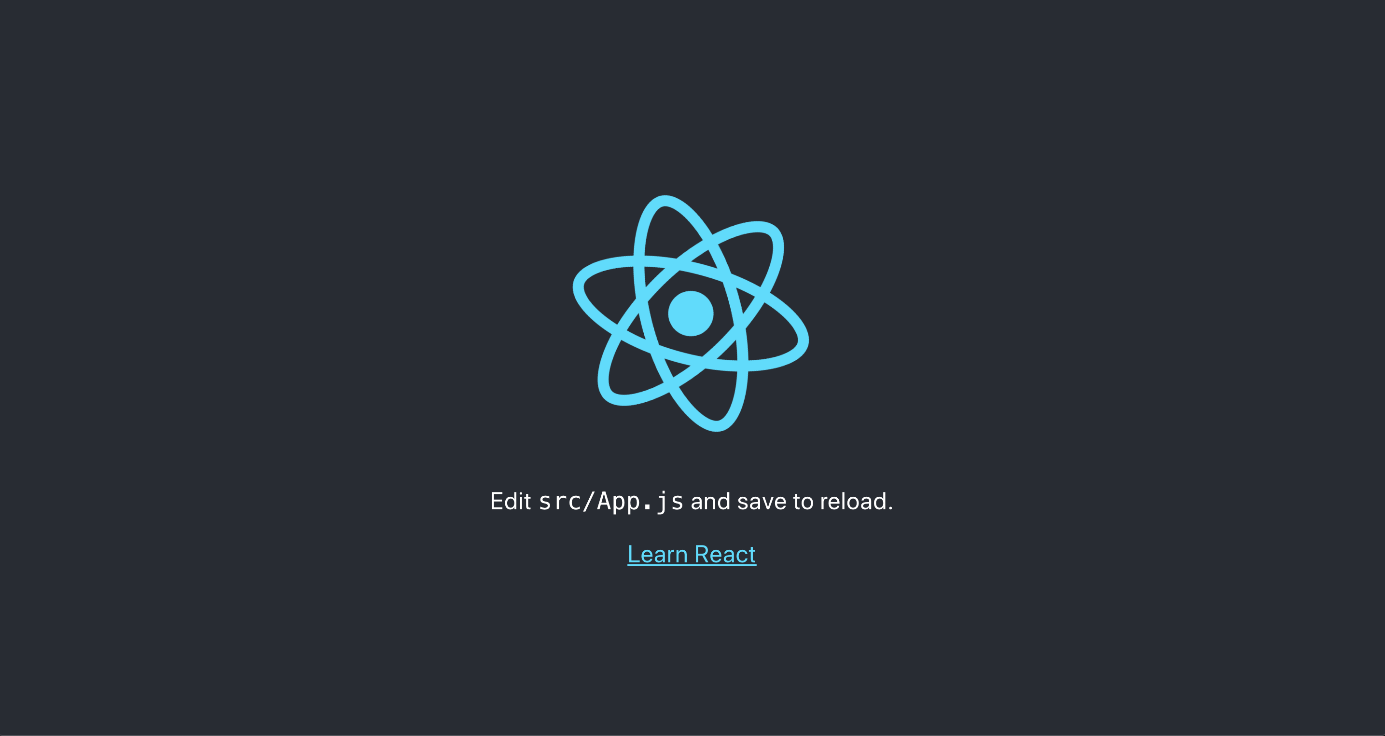
transform: rotate(360deg);

}

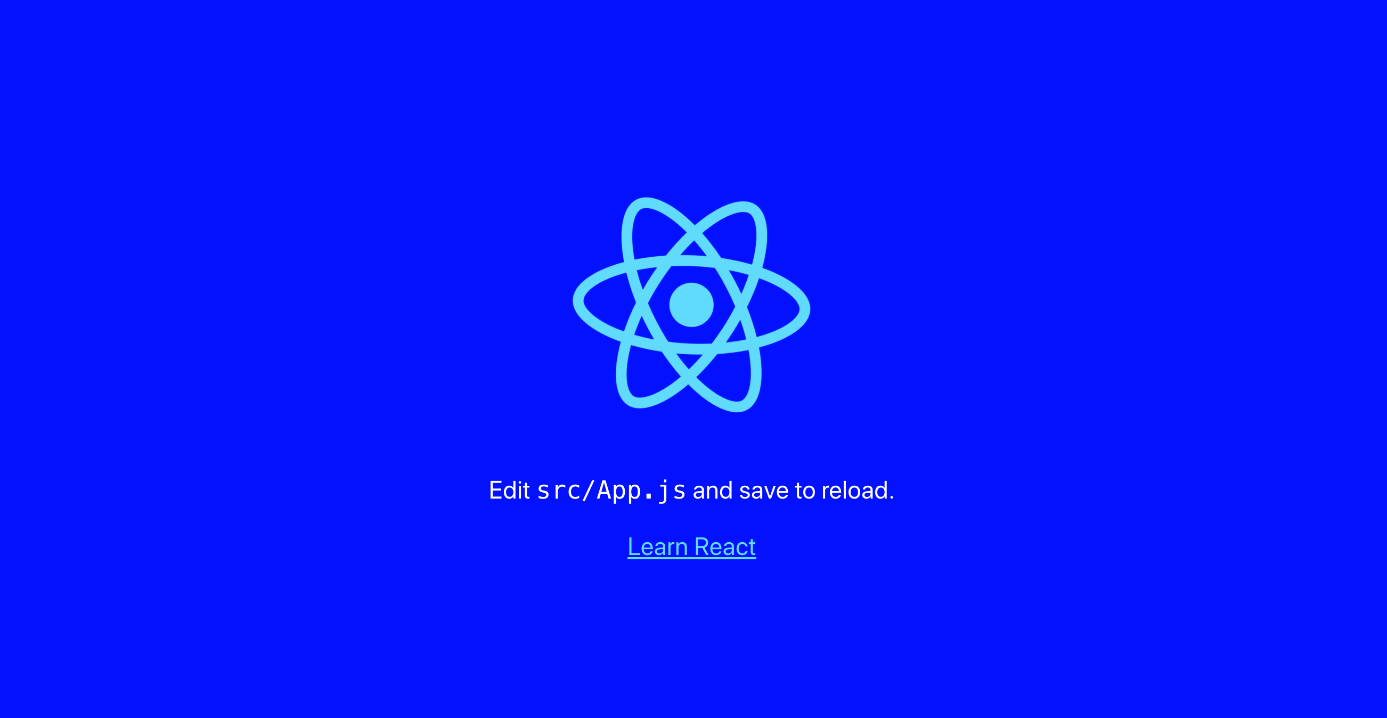
}

Copie

Vérifiez votre navigateur. Voici à quoi ça ressemblait avant :



Voici à quoi cela ressemblera après le changement :



Allez-y et background-colorrevenez à #282c34.

digital-ocean-tutoriel/src/App.css

.App {

text-align: center;

...

.App-header {

background-color: #282c34

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

...

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

Copie

Enregistrez et quittez le fichier.

Vous avez apporté une petite modification CSS. Il est maintenant temps d'apporter des modifications au code JavaScript React. Commencez par ouvrir index.js.

1. nano src/index.js

Copie

Voici ce que vous verrez :

digital-ocean-tutoriel/src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import \* as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

Copie

En haut, vous importez React, ReactDOM, index.css, Appet serviceWorker. En important React, vous tirez en fait du code pour convertir JSX en JavaScript. JSX sont les éléments de type HTML. Par exemple, remarquez que lorsque vous utilisez App, vous le traitez comme un élément HTML <App />. Vous explorerez cela plus en détail dans les futurs didacticiels de cette série.

ReactDOMest le code qui relie votre code React aux éléments de base, comme la index.htmlpage que vous avez vue dans public/. Regardez la ligne en surbrillance suivante :

digital-ocean-tutoriel/src/index.js

...

import \* as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

...

serviceWorker.unregister();

Copie

Ce code demande à React de trouver un élément avec un idof rootet d'y injecter le code React. <App/>est votre élément racine, et tout se ramifiera à partir de là. C'est le point de départ de tout futur code React.

En haut du fichier, vous verrez quelques importations. Vous importez index.css, mais ne faites rien avec. En l'important, vous dites à Webpack via les scripts React d'inclure ce code CSS dans le bundle final compilé. Si vous ne l'importez pas, il n'apparaîtra pas.

Sortir de src/index.js.

À ce stade, vous n'avez toujours rien vu de ce que vous visualisez dans votre navigateur. Pour le voir, ouvrez App.js:

1. nano src/App.js

Copie

Le code de ce fichier ressemblera à une série d'éléments HTML normaux. Voici ce que vous verrez :

digital-ocean-tutoriel/src/App.js

import React from 'react';

import logo from './logo.svg';

import './App.css';

function App() {

return (

<div className="App">

<header className="App-header">

<img src={logo} className="App-logo" alt="logo" />

<p>

Edit <code>src/App.js</code> and save to reload.

</p>

<a

className="App-link"

href="https://reactjs.org"

target="\_blank"

rel="noopener noreferrer"

>

Learn React

</a>

</header>

</div>

);

}

export default App;

Copie

Modifiez le contenu de la <p>balise de Edit <code>src/App.js</code> and save to reload.à Hello, worldet enregistrez vos modifications.

digital-ocean-tutoriel/src/App.js

...

function App() {

return (

<div className="App">

<header className="App-header">

<img src={logo} className="App-logo" alt="logo" />

<p>

Hello, world

</p>

<a

className="App-link"

href="https://reactjs.org"

target="\_blank"

rel="noopener noreferrer"

>

Learn React

</a>

</header>

</div>

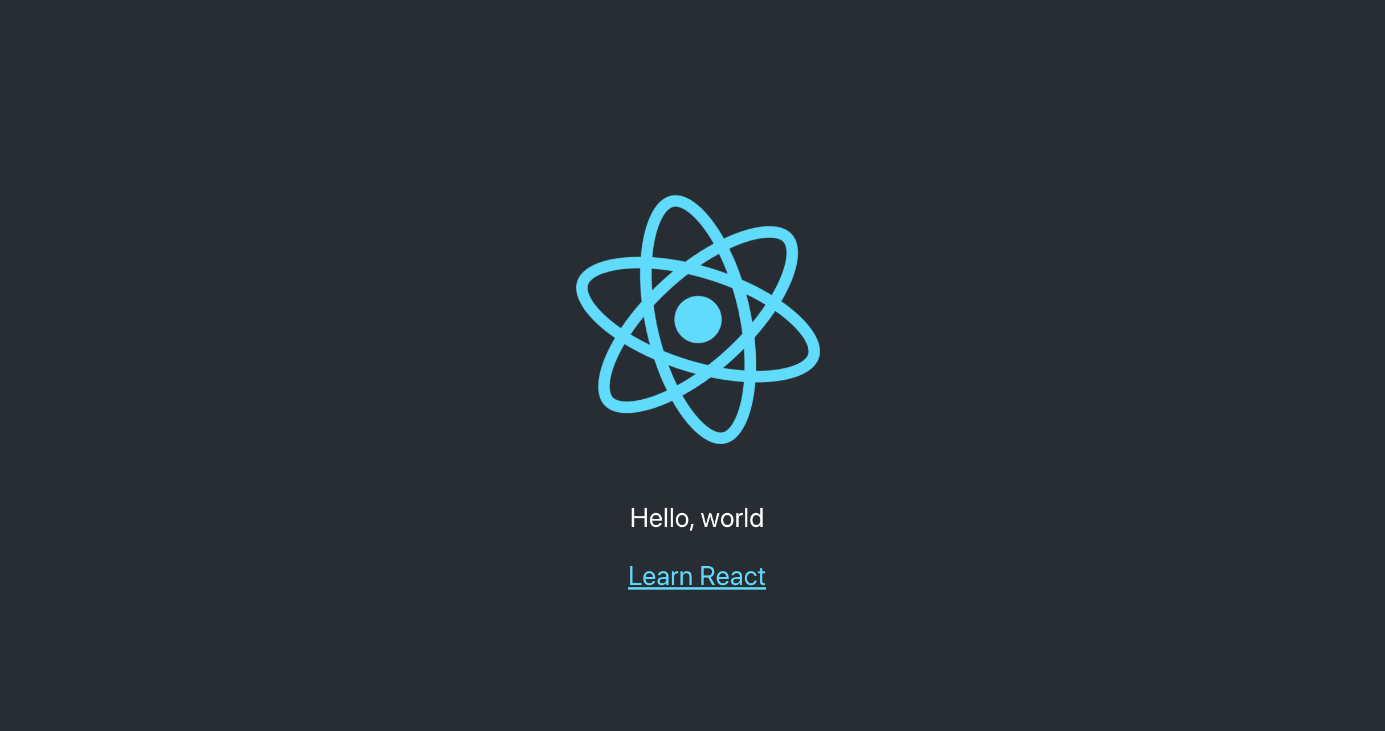
);

}

...

Copie

Rendez-vous sur votre navigateur et vous verrez le changement :



Vous avez maintenant effectué votre première mise à jour d'un composant React.

Avant de partir, remarquez quelques autres choses. Dans ce composant, vous importez le logo.svgfichier et l'affectez à une variable. Ensuite, dans l' <img>élément, vous ajoutez ce code en tant que fichier src.

Il se passe quelques choses ici. Regardez l' imgélément :

digital-ocean-tutoriel/src/App.js

...

function App() {

return (

<div className="App">

<header className="App-header">

<img src={logo} className="App-logo" alt="logo" />

<p>

Hello, world

</p>

...

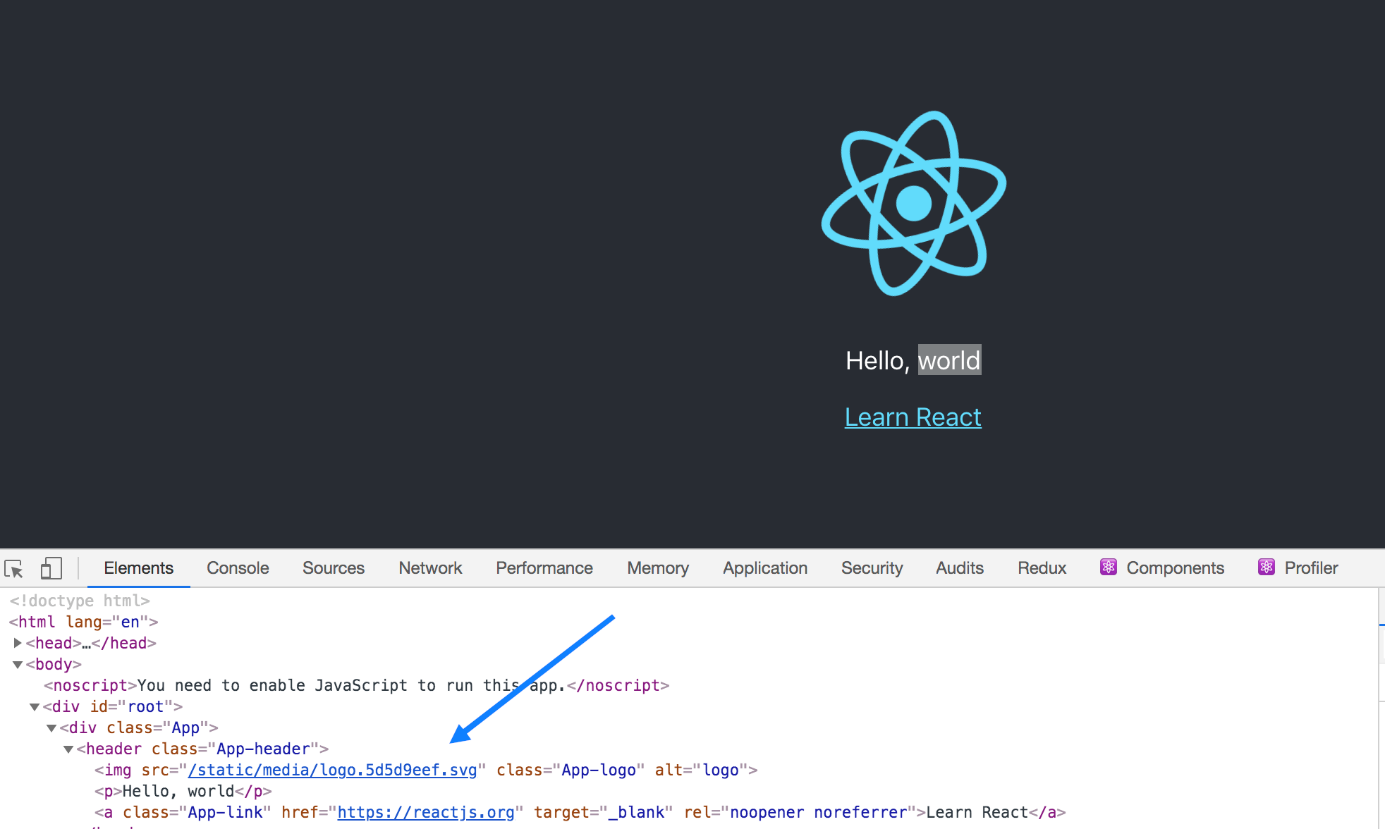
Copie

Remarquez comment vous passez les logoaccolades. Chaque fois que vous transmettez des attributs qui ne sont pas des chaînes ou des nombres, vous devez utiliser les accolades. React les traitera comme du JavaScript au lieu de chaînes. Dans ce cas, vous n'importez pas réellement l'image ; à la place, vous faites référence à l'image. Lorsque Webpack construit le projet, il gère l'image et définit la source à l'endroit approprié.

Quittez l'éditeur de texte.

Si vous regardez les éléments DOM dans votre navigateur, vous verrez qu'il ajoute un chemin. Si vous utilisez [Chrome](https://developers.google.com/web/tools/chrome-devtools) , vous pouvez inspecter l'élément en cliquant avec le bouton droit sur l'élément et en sélectionnant **Inspecter** .

Voici à quoi cela ressemblerait dans le navigateur :



Le DOM a cette ligne :

<img src="/static/media/logo.5d5d9eef.svg" class="App-logo" alt="logo">

Copie

Votre code sera légèrement différent puisque le logo aura un nom différent. Webpack veut s'assurer que le chemin de l'image est unique. Ainsi, même si vous importez des images portant le même nom, elles seront enregistrées avec des chemins différents.

À ce stade, vous avez apporté une petite modification au code JavaScript de React. À l'étape suivante, vous utiliserez la buildcommande pour réduire le code dans un petit fichier pouvant être déployé sur un serveur.

## Étape 6 - Construire le projet

Dans cette étape, vous allez créer le code dans un bundle qui peut être déployé sur des serveurs externes.

Retournez à votre terminal et construisez le projet. Vous avez déjà exécuté cette commande, mais pour rappel, cette commande exécutera le buildscript. Il créera un nouveau répertoire avec les fichiers combinés et minifiés. Pour exécuter la compilation, exécutez la commande suivante à partir de la racine de votre projet :

1. npm run build

Copie

Il y aura un délai au fur et à mesure que le code se compile et quand il sera terminé, vous aurez un nouveau répertoire appelé build/.

Ouvrir build/index.htmldans un éditeur de texte.

1. nano build/index.html

Copie

Vous verrez quelque chose comme ceci :

digital-ocean-tutorial/build/index.html

<!doctype html><html lang="en"><head><meta charset="utf-8"/><link rel="icon" href="/favicon.ico"/><meta name="viewport" content="width=device-width,initial-scale=1"/><meta name="theme-color" content="#000000"/><meta name="description" content="Web site created using create-react-app"/><link rel="apple-touch-icon" href="/logo192.png"/><link rel="manifest" href="/manifest.json"/><title>React App</title><link href="/static/css/main.d1b05096.chunk.css" rel="stylesheet"></head><body><noscript>You need to enable JavaScript to run this app.</noscript><div id="root"></div><script>!function(e){function r(r){for(var n,a,p=r[0],l=r[1],c=r[2],i=0,s=[];i<p.length;i++)a=p[i],Object.prototype.hasOwnProperty.call(o,a)&&o[a]&&s.push(o[a][0]),o[a]=0;for(n in l)Object.prototype.hasOwnProperty.call(l,n)&&(e[n]=l[n]);for(f&&f(r);s.length;)s.shift()();return u.push.apply(u,c||[]),t()}function t(){for(var e,r=0;r<u.length;r++){for(var t=u[r],n=!0,p=1;p<t.length;p++){var l=t[p];0!==o[l]&&(n=!1)}n&&(u.splice(r--,1),e=a(a.s=t[0]))}return e}var n={},o={1:0},u=[];function a(r){if(n[r])return n[r].exports;var t=n[r]={i:r,l:!1,exports:{}};return e[r].call(t.exports,t,t.exports,a),t.l=!0,t.exports}a.m=e,a.c=n,a.d=function(e,r,t){a.o(e,r)||Object.defineProperty(e,r,{enumerable:!0,get:t})},a.r=function(e){"undefined"!=typeof Symbol&&Symbol.toStringTag&&Object.defineProperty(e,Symbol.toStringTag,{value:"Module"}),Object.defineProperty(e,"\_\_esModule",{value:!0})},a.t=function(e,r){if(1&r&&(e=a(e)),8&r)return e;if(4&r&&"object"==typeof e&&e&&e.\_\_esModule)return e;var t=Object.create(null);if(a.r(t),Object.defineProperty(t,"default",{enumerable:!0,value:e}),2&r&&"string"!=typeof e)for(var n in e)a.d(t,n,function(r){return e[r]}.bind(null,n));return t},a.n=function(e){var r=e&&e.\_\_esModule?function(){return e.default}:function(){return e};return a.d(r,"a",r),r},a.o=function(e,r){return Object.prototype.hasOwnProperty.call(e,r)},a.p="/";var p=this["webpackJsonpdo-create-react-app"]=this["webpackJsonpdo-create-react-app"]||[],l=p.push.bind(p);p.push=r,p=p.slice();for(var c=0;c<p.length;c++)r(p[c]);var f=l;t()}([])</script><script src="/static/js/2.c0be6967.chunk.js"></script><script src="/static/js/main.bac2dbd2.chunk.js"></script></body></html>

Le répertoire de construction prend tout votre code et le compile et le minimise dans le plus petit état utilisable. Peu importe qu'un humain puisse le lire, car il ne s'agit pas d'un morceau de code accessible au public. Minifier ainsi fera en sorte que le code prendra moins de place tout en lui permettant de fonctionner. Contrairement à certains langages comme Python, les espaces ne changent pas la façon dont l'ordinateur interprète le code.

## Conclusion

Dans ce didacticiel, vous avez créé votre première application React, en configurant votre projet à l'aide des outils de construction JavaScript sans avoir à entrer dans les détails techniques. C'est la valeur de Create React App : vous n'avez pas besoin de tout savoir pour commencer. Il vous permet d'ignorer les étapes de construction compliquées afin que vous puissiez vous concentrer exclusivement sur le code React.

Vous avez appris les commandes pour démarrer, tester et créer un projet. Vous utiliserez ces commandes régulièrement, alors prenez-en note pour les futurs didacticiels. Plus important encore, vous avez mis à jour votre premier composant React.