React

"create-react-app", emballé c'est pesé !

React est un framework javascript permettant, entre autres, de créer des applications web sans rechargement de page. Ce qu'on appelle les SPA, pour Société Protectrice des Animaux… Euh pardon, plutôt pour **Single Page Application**.

Dans un site classique, quand on clique sur un lien, une requête est envoyée vers un serveur qui renvoie, après un temps de traitement plus ou moins long, une réponse au navigateur. Cette réponse inclue généralement du contenu HTML. Pendant cette communication entre le navigateur et le serveur, la page du navigateur est en attente de chargement : page blanche, spinner qui tourne… C'est ce que l'on appelle le SSR pour **Server Side Rendering**.

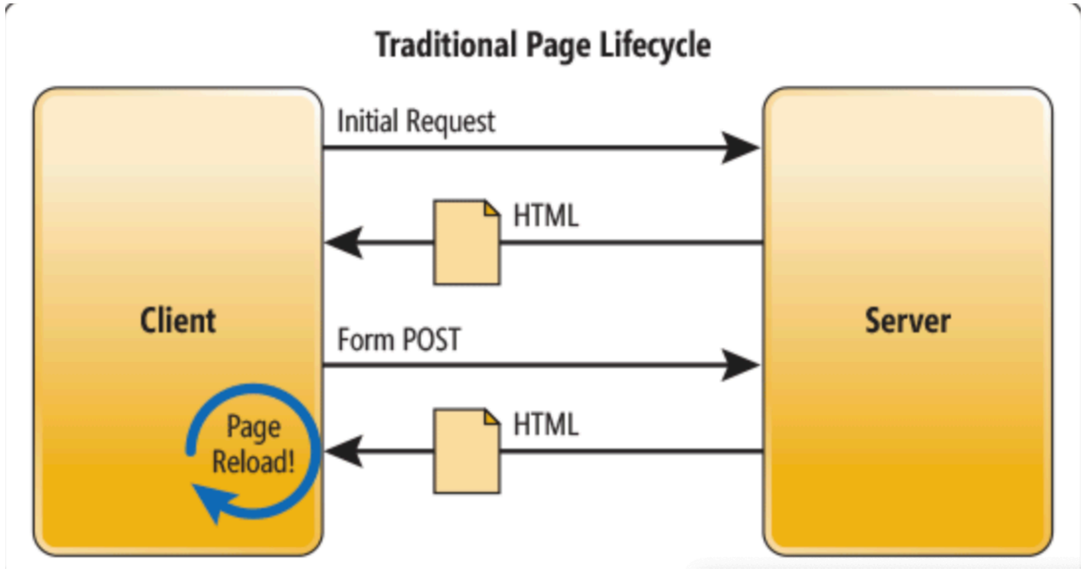

Image : https://docs.microsoft.com/

Les SPA eux gèrent cette communication de manière différente. Quand on clique sur un lien, ce n'est pas la page entière qui est rechargée : le contenu de la page actuelle est juste remplacé par le contenu de la prochaine page, sans que les contenus communs aux deux pages ne soient modifiés (menu, footer etc). Les nouvelles données affichés sur les pages sont généralement récupérées via des requêtes AJAX, et le rendu HTML incluant ces nouvelles données en Javascript, donc côté navigateur.

De cette manière, la navigation entre les pages est beaucoup plus fluide pour l'utilisateur.

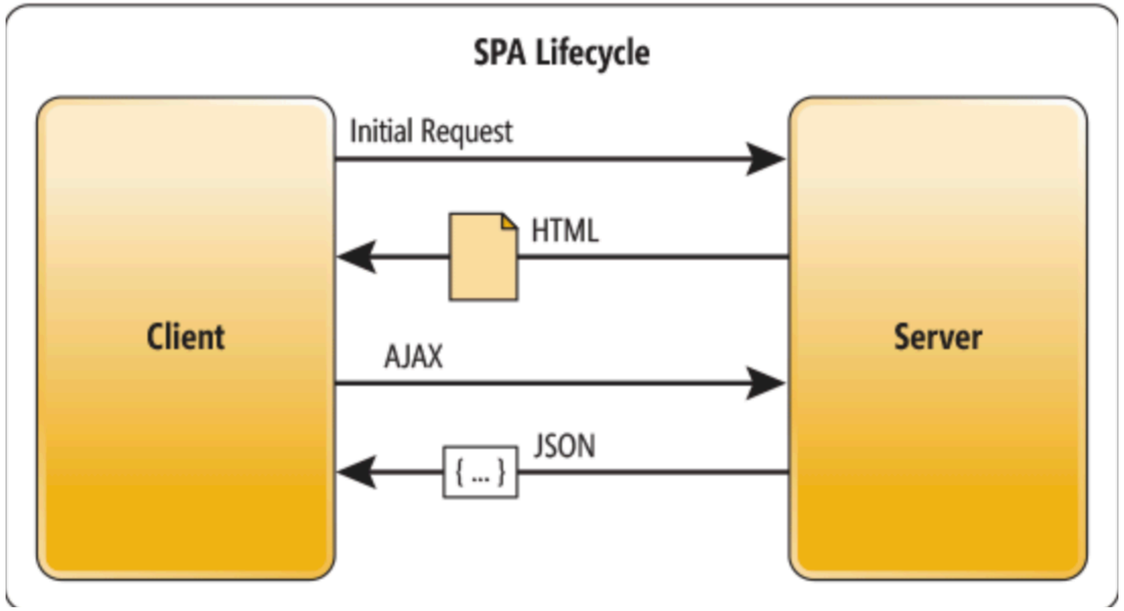

Image : https://docs.microsoft.com/

Ce que permettent les SPA n’a pourtant rien de nouveau : remplacer le contenu d’un bout de page en Javascript, c’est à dire modifier le DOM, est possible depuis longtemps. Idem pour faire des appels AJAX pour récupérer des données sur un serveur.

Mais la quantité de code javascript à écrire pour faire fonctionner un site entier en SPA et l’impossibilité d’organiser correctement son code en module étaient très problématique avant 2010, même en utilisant la librairie la plus populaire de l’époque : jQuery.

Exemples :
- https://developer.mozilla.org/fr/docs/Web/API/Document/createElement
- https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_append_ref)

Autour de 2010 l'écosystème javascript à commencer à exploser avec la création d'un nouveau moteur javascript bien plus performant pour les navigateurs: le V8 engine de Chrome. S'en est suivi la création de **Node.js** et de frameworks / librairies permettant de manipuler le DOM plus facilement et donc faire des SPA : Backbone, Knockout, Angular, Vue ou encore bien sûr React…
L'arrivée de NPM et de Yarn (des packages managers) a aussi permis d'installer ces frameworks / librairies plus facilement.

Chaque outil à sa spécificité : Angular est plus stricte et impose un cadre de travail plus fort, React est très proche du javascript, Vue est facile d'utilisation et permet de développer rapidement. Les outils moins modernes et performants comme Knockout ou Backbone JS eux disparaissent progressivement.

Aujourd'hui l'écosystème continue de s'enrichir avec des outils comme Svelte, Next.js, Nuxt, Remix, Remotion et bien d'autres à tel point que le langage javascript est omniprésent et sur tous les fronts.

Pour installer React, l'installation de Node.js et de NPM n'est pas obligatoire, mais très conseillé : cela permet d'installer les différents outils dont React a besoin, mais aussi par exemple de créer un serveur local qui se recharge automatiquement à chaque modification de code.

React sur le web a notamment besoin de **React** et **React DOM**. Pour convertir le code **ES6** (c'est à dire le dernier standard d'écriture du Javascript, utilisé dans React) en Javascript que tous les navigateurs peuvent lire il faut utiliser un outil que l'on appelle **Babel** (un « transpiler »). Pour regrouper (« bundler ») les différents fichiers Javascript utilisés dans React dans un fichier il faut aussi utiliser **Webpack**.

Sans Node.js et NPM, l'installation de ces outils serait très (trop) fastidieuses.

La configuration du serveur de développement et des différents outils que React utilise, notamment Webpack et Babel peut devenir un véritable calvaire avec des fichiers de configuration à rallonge et pas évidents à comprendre.

Pour cette raison, React a mis un place un outil d'installation permettant d'installer et de configurer en une seule ligne de commandes tous les outils dont React a besoin. Et ce grâce à NPM et la commande reliée *npx* qui permet d'installer et d'exécuter des packages :

npx create-react-app *nom-du-projet*

Démarrez ensuite votre serveur react avec :

npm start

Si cela fonctionne bien, l'adresse d'un serveur local vous sera indiqué dans le résultat de la ligne de commande. Utilisez cette adresse et vous pourrez voir la page d'accueil par défaut de votre application React !