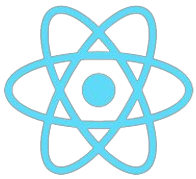
Tutoriel

Comment utiliser Bootstrap ou React.js pour une application adaptée aux usages ?



React

Bootstrap

Sonia Tual & Garance Vallat

Adaptation des interfaces à l'environnement - SI5 IHM

Anne-Marie Dery - Christian Brel

22 Octobre 2015

Présentation du sujet

Notre application est un site web de recettes de cuisine. Notre preuve de concept présente une page unique, avec une recette complète : les ingrédients, les étapes, et des informations complémentaires.

Notre objectif est de nous adapter en fonction des cas d'usages des utilisateurs. En effet, nous supposons qu'en fonction du support, on ne recherche pas les mêmes informations à propos d'une recette, même si on est sur le même site internet.

Par exemple, si vous consultez le site via mobile, nous considérons par défaut que vous êtes au supermarché, et que c'est donc de la liste des ingrédients dont vous avez besoin.

Si vous consultez le site via tablette, nous considérons par contre que vous êtes plutôt dans une phase de réalisation de la recette, c'est donc des étapes dont vous avez principalement besoin, avec les ingrédients concernés.

A l'inverse, si vous êtes sur un ordinateur de bureau, nous supposons que vous êtes à la recherche d'une recette, dans une phase de prise de décision. Nous vous présentons donc la recette dans son ensemble.

Pour réaliser cette application, nous avons choisi d'utiliser deux technologies : Bootstrap et React.js. Nous ne les utilisons pas de manière complémentaires, mais toutes les deux pour agir sur le html, et modifier l'apparence en fonction du support. Nous allons présenter d'abord comment réaliser ce projet avec Bootstrap, puis avec React.js.

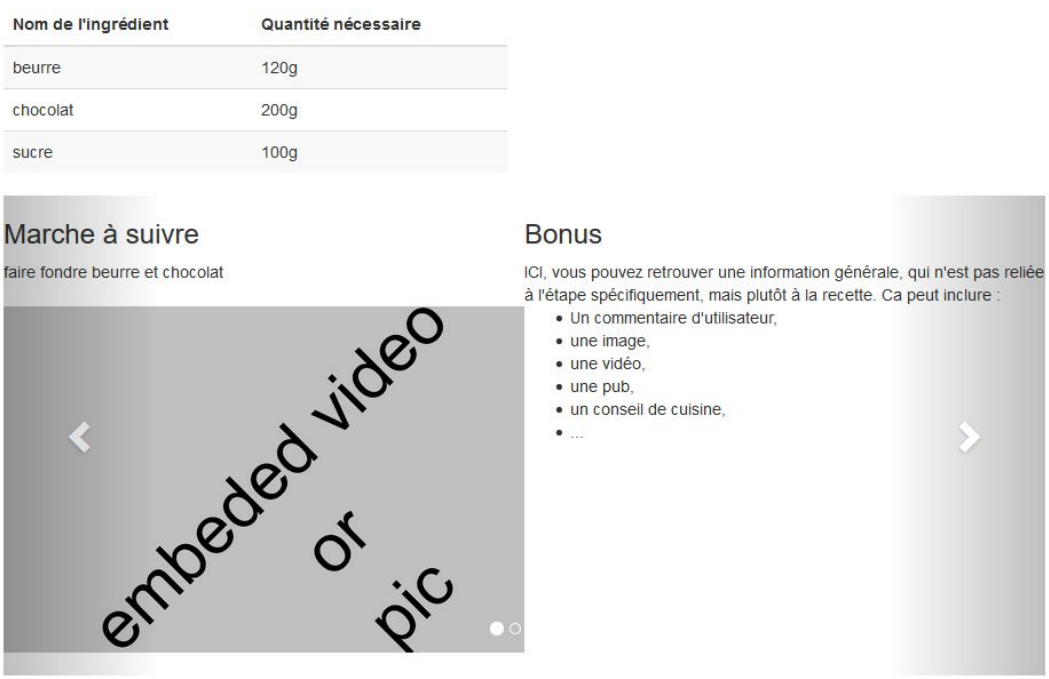
Bootstrap

Bootstrap est un cadriciel dont le code est ouvert, qui contient à la fois du html, du css et du javascript, pour former une collection d'outils qui facilitent la création de sites internets. Bootstrap privilégie la création de site internet pour mobile, qui peuvent s'adapter à différentes tailles d'écran : c'est la raison pour laquelle nous l'avons choisi.

Voici trois différentes vue de l'application, telle qu'elle est développée grâce à Bootstrap :

Vue mobile

Vue tablette



Vue Bureau

Outils de développement et de test

Se lancer

Pour développer avec Bootstrap, nous avons monté un environnement javascript, à l'aide du gestionnaire de dépendances npm.

Avec npm, nous importons grunt, qui permet de déployer un mini serveur, et d'observer les fichiers du projet à chaque changement, afin de développer et tester sans y penser.

Nous utilisons aussi le gestionnaire de dépendances bower, qui est spécialisé dans le front-end contrairement à npm. La seule dépendance que nous utilisons est Bootstrap. Passer par ce gestionnaire malgré tout permet d'accélerer les tests en cours de développement, comme la bibliothèque est enregistrée sur l'ordinateur, au lieu d'être importée sur internet à chaque chargement de la page.

La bibliothèque Bootstrap utilise intensivement des classes html avec un CSS prédéfini. Ainsi, pour créer l'espace "accueil" du site, nous avons intégré le texte de bienvenue dans une div à laquelle est associée la classe "`jumbotron`", et le cadre gris, avec la police d'écriture plus grande que le reste du texte sont définis de manière transparente pour nous, dans la bibliothèque.

Notre usage de Bootstrap

Pour que l'application s'adapte aux cas d'usages, nous utilisons intensément les classes d'aides de Bootstrap, mais nous ignorons les aides au design fournies par Bootstrap : ce n'est pas notre objectif.

Les classes d'aide de Bootstrap sont des classes prédéfinies dans la bibliothèque, qu'on peut ajouter sur n'importe quel noeud de notre html, pour modifier la façon dont il s'affiche.

Bootstrap distingue 4 catégories de taille d'écran, de très petit à grand, nous en avons utilisé 3 :

- très petit pour les téléphones (<768px),
- petit pour les tablettes (≥768px),
- moyen pour les ordinateurs de bureau (≥992px), qui contient également la catégorie supérieure.

Nous avons surtout exploité les classes "visible" et "hidden" pour nos éléments. En tant que développeur, cet usage pousse à distinguer dès le début de l'écriture du code les éléments qui sont faits pour chaque support.

Visible ou caché ?

On peut choisir d'avoir certains noeuds visibles ou invisibles pour une résolution d'écran seulement ou plusieurs. Nous avons utilisé les définitions de Bootstrap, en faisant confiance à ses choix concernant la nature des supports, en fonction des tailles d'écran.
Cette caractéristique permet de tester l'adaptation de notre interface directement à partir du navigateur de notre ordinateur de travail.

Ainsi, le carrousel est visible seulement pour les supports petits et moyens, et à l'intérieur du carrousel, certains noeuds html ne sont visibles que pour certains supports.

Eléments "adaptatifs"

Nous avons aussi mêlé dans notre application des éléments Bootstrap adaptatifs, qui ont donc un aspect différent en fonction de la taille de l'écran utilisé, tout en conservant toutes les informations.

C'est le cas par exemple de la liste d'ingrédients de la recette : elle est visible dans son intégralité aussi bien sur téléphone que sur un ordinateur de bureau. Cependant, la largeur des colonnes varie en fonction de l'espace disponible grâce à la classe "table" de Bootstrap, ce qui permet d'augmenter la lisibilité en un coup d'oeil sur petit support, dans la mesure du possible.

Comment déployer et exécuter

Pour déployer le projet, il faut d'une part télécharger les dépendances nécessaires, grâce à deux commandes : "npm install" pour avoir grunt à notre disposition, puis "bower install" pour avoir Bootstrap.

Grâce aux outils de développement que nous utilisons, l'exécution en "test" est très simple : la commande "grunt serve" permet de lancer un mini serveur qui sert l'application sur le port de notre choix.

Pour savoir comment tester les capacités d'adaptation de notre application, rendez-vous à la fin du document : les modalités sont identiques pour nos deux technologies.

React.js

React.js est un cadriciel javascript dont le code source est ouvert, qui est maintenu par Facebook et Instagram.

Outils de développement et de test

Se lancer

Pour utiliser React.js, nous avons commencé par installer des dépendances avec npm.

Nous importons react et react-dom qui sont les deux dépendances indispensables pour faire un projet avec React.js.

Puis nous utilisons gzippo et express qui nous serviront à créer un serveur Node.JS.

web.js

```
var gzippo = require('gzippo');
var express = require('express');
var app = express();

//app.use(express.logger('dev'));
app.use(gzippo.staticGzip("" + __dirname + ""));
app.use('/*', function(req, res){
  res.sendfile(__dirname + '/index.html');
});
app.listen(process.env.PORT || 10000);
```

Le serveur est réduit à sa plus simple expression : un gestionnaire spécifique n'est pas forcément nécessaire. Nous avons fait ce choix afin de montrer, entre cette application et la précédente, les différentes possibilités pour lancer une application web.

Notre usage de React

Pour cette application, nous utilisons les composants de React pour découper les éléments de notre page. Cela permettra de choisir les composants que l'on veut afficher en fonction de la taille de l'écran. Nous avons utilisé les media queries de CSS3 pour cacher et rendre visibles les composants.

Les composants React

L'outil babel qu'il faut installer sert à compiler le JSX, qui est une extension de syntaxe javascript, qui nous permet d'utiliser la syntaxe particulière de React de manière simple et intuitive qu'on peut voir en dessous, tout en restant dans un fichier javascript.

```
var Table = React.createClass({
    render: function () {
        return (
            <div className="table">
                <table>
                    <caption>Ingredients</caption>
                    <tbody><tr>
                        <td>beurre</td>
                        <td>120g</td>
                    </tr>
                    </tbody>
                </table>
            </div>
        );
    }
});
```

Un composant React est assez simple à déclarer, il faut créer une classe comme “table” ci-dessus. A l’intérieur, on peut trouver simplement des balises html avec du texte, mais on peut aussi y attacher des états, des propriétés et des évènements.

Un composant peut être utilisé dans un autre ou affiché directement dans la page en l’envoyant au render de ReactDOM. Pour utiliser le composant Table par exemple, il suffit de placer la balise “<Table />” à l’endroit voulu, c’est à dire dans un autre composant ou directement dans le “render” du DOM.

Au final, un composant React est un noeud html. Cependant, tous ces noeuds sont intégrés aux fichiers html seulement après avoir été “calculés”, avec une seule modification du DOM. Cette modification se fait avec un simple appel à la méthode “`getElementById`”.

Les media queries CSS

Une media query est un moyen en CSS de définir différents comportements pour nos noeuds html en fonction d’une taille d’écran que nous définissons nous-mêmes.
Nous avons choisi d’utiliser les mêmes jalons que Bootstrap, afin de rester cohérentes entre nos applications.

L’intersection entre les classes définies par composant avec React et les media queries nous a permis de retrouver un système proche de celui fait en transparence par Bootstrap. Cependant nous aurions pu, de manière plus simple qu’avec Bootstrap, choisir d’autres points de rupture, spécifiquement pensés pour notre application.

Comment déployer et exécuter

Pour déployer le projet, il faut d’une part télécharger les dépendances nécessaires, grâce à deux commandes : “npm install”, puis “npm install --global babel” pour avoir babel. Ensuite il faut utiliser la commande “babel src --watch --out-dir build” pour générer le Javascript dans un dossier build à partir des fichiers contenant du JSX dans le dossier src.

Pour lancer le serveur, il suffit de faire la commande “node web.js” et l’application sera sur le port 10000.

Comment peut-on tester les capacités d'adaptation ?

Pour nos deux applications, nous avons utilisé des technologies entièrement "Web", avec Javascript et du html. Tous les tests peuvent donc se faire sur navigateur, quel que soit le support. Pour éviter de tester sur tous les dispositifs un à un, deux possibilités s'offrent à nous :

- On peut utiliser des sites spécialisés pour tester le rendu selon le format. (la plupart du temps, seulement si le site est déployé sur un serveur distant)
- On peut également utiliser un mode adaptatif sur son navigateur, pour simuler différentes tailles d'écran.

Nous avons utilisé le site http://cybercrab.com/screencheck, qui permet de simuler la résolution de différents supports en fonction des marques, avec les orientations paysage et portrait.

Enfin, les différents navigateurs n'implémentent pas tous les mêmes normes html et css, une vérification par navigateur est également nécessaire.

Pour notre application, on peut voir au fur et à mesure du document les vues en fonction des tailles d'écran. Pour ce qui est de la compatibilité inter-navigateur, comme l'adaptation, des outils existent pour simuler les tests sur les différents navigateurs, pour pallier à la question d'avoir tous les navigateurs installés sur son poste de travail. Cependant, ces outils nécessitent d'avoir un site déployé sur un serveur distant.

Conclusion

Pour concevoir une application qui s'adapte à différents usages, nous avons donc choisi de prendre en compte l'utilisation de différents supports. Cette décision permet de faciliter le développement. En effet, cela permet de rester dans le champ du code et des outils techniques, plutôt que de s'interroger sur les utilisateurs, et de communiquer avec eux sur leurs besoins spécifiques. Dans le cadre d'un petit site internet, cette solution nous semble satisfaisante, bien que pas idéale. Elle ne remplace pas une étude poussée des usages, et la réalisation d'applications différentes en fonction des différents besoins exprimés.

Les deux technologies que nous avons étudiées permettent chacune de remplir le contrat, avec plus de difficulté pour React, en raison de la pile technologique à maîtriser, qui est beaucoup plus importante. React permet de faire un projet moins rapidement, mais qui sera plus performant pour le long terme, grâce aux composants réutilisables notamment.