ECMAScript 2015

Les outils

La compatibilité

Kangax

- http://kangax.github.io/compat-table/es6/

- Contient un tableaux de toutes les features avec leur support
- Mis à jour régulierement

Recap

- Chrome: 98% (98%)
- Edge : 79% (90%)
- Firefox : 90% (93%)
- Safari/iOs : 53% (100%)
- NodeJs : 93%
- Babel : 74%
- IE11 : 15% ☹
- Android 4.4 : 11% ☹
- Android 5.1 : 29% ☹

Polyfills et transpilers

Polyfill

- Permet d'émuler une API manquante
- Se présente sous la forme d'une librairie JavaScript à inclure
- Le nom vient de la pate « polyfilia » qui permet de boucher des trous dans n'importe quelle surface

  Un polyfill bouche un trou dans le navigateur/nodeJs

- Il existe de nombreux polyfills pour les APIs HTML5, ES5 et ES2015

Premier outil sympa pour le navigateur : Modernizr

Transpileurs

- Lorsque la syntaxe change en profondeur les polyfills atteignent leurs limites
- Transpileur = Tranformer + compileur
- Un transpileur compile un langage de haut niveau en un autre :
  - Java -> JavaScript
  - Java → C
  - CoffeeScript → JavaScript
  - TypeScript → JavaScript
  - JavaScript → JavaScript

- L'idée est de coder en suivant les standards sans s'occuper de compatibilité et d'utiliser un transpiler pour transformer notre code en une version compatible avec notre cible

  ES2015 → ES5 → ES3

Transpilers : les noms

- Babel
  - Le plus connu et utilisé
  - Soutenu par Facebook
  - Anciennement connu sous le nom ES6to5
  - Fonctionne via des plugins
- Traceur
  - Moins complet que Babel mais possède l'essentiel
  - Uniquement es2015 -> ES5
- Closure
  - Le moins complet des 3
  - But principal : optimiser le javascript

Babel

Principe

- Transpiler le plus connu
- Transforme du JavaScript en un autre JavaScript
- Depuis la version 6, par défaut ne fait rien
- Utilisation de plugins pour définir ce qui doit être transformé en quoi
- Il existe des packs de plugins appelés « presets »
- Principalement utilisé pour ES2015 mais il existe de nombreux plugins (pour compiler le JSX si vous faites du React par exemple)
- Se lance ne ligne de commande…
- … ou via des outils de build (gulp, grunt, etc.) …
- … ou via webpack …
- … ou via un hook nodeJS

L'installer

- Dans tous les cas il vous faudra :

  npm install --save-dev babel-polyfill

  npm install --save-dev babel-preset-es2015

- Un fichier **.babelrc** à la racine de votre projet avec la configuration :

```
{
   presets": ["es2015"]
}
```

Ligne de commande

- L'installer :

  npm install -g babel-cli

- La lancer (compile le répertoire src et le foichier main.js dans le répertoire /lib) :

  babel src main.js –d lib

Hook nodeJs

- Renommez main.js en main-es2015.js
- Créer un fichier main.js avec :

```
require('babel-register');
require('babel-polyfill');

require('./main-es2015.js');
```

- Charge le hook babek
- Charge le polyfill babel
- A partir de ce point le code est compiler à chaud lorsque vous importez un module
- Charge et exécute le fichier **main-es2015.js**

Dans le navigateur ?

- Précompile ses fichiers lors de la mis en production
- Selon la version du navigateur nécessite des polyfills + compilation
- Par défaut compile vers ES5
- Si besoin peut compiler vers ES3 pour IE < 9 mais tout n'est pas disponible
- La compilation à un coût en terme de poids et d'overhead
- Vous aurez également besoin d'un système pour gérer les modules (mais nous verrons ça en temps voulut)

TP : mettre en place Babel

ESLint

https://medium.com/@dan_abramov/lint-like-it-s-2015-6987d44c5b48#.vvg8nlcv7