ES6+ maintenant !

Speaker : Christophe Porteneuve
Format : Conférence
Date : 21 avril 2016
Slides : http://tdd.github.io/devoxx-es6-maintenant/

Ancien développeur Java, Christophe développe en JavaScript depuis 1995.
Il comme par rappeler qu'ECMAScript est la standardisation de JavaScript.

Historique : d'ES3 à ES7

- Mai 1995 : JS 1.0 (en 10 jours). Netscape 2.0 beta 3 en 12/95. Conçu en même temps que Java. Pourquoi diable avoir choisi ce nom ? => Trop de confusions au niveau RH.
- Juin 1997 : ECMAScript (ES1, ECMA-262), standard officiel => Utilisé par IE8
- Décembre 1999 : ES3. JScript 1.5 à peu près à ce niveau (IE4-8)
- Décembre 2009 : ES5. Baseline de compatibilité actuelle. IE9+, Node, etc. Peu de gros changements.
- Juin 2015 : ES6 / ES2015. **Enormément** de nouveautés de **langage pur**.
- Juin 2016 : ES7 / ES2016 (prévu ; versions annuelles désormais, dans le cadre d'ES.Next, d'où le changement de nom)

Prises-en charge native d'ES6 ?

- Navigateurs : de 90% à 98% sur les navigateurs Evergreens. Safari est resté sur 53%. D'après Christophe, Apple bride volontairement son navigateur afin de promouvoir le développement d'applications natives.
- Serveurs : Node LTS (4.4.3) à 48%, Node stable (5.10.1) à 58%

Babel

- Transpile ES6+ en ES5
- En pratique : IE9+, navigateurs Evergreens, tous les Nodes.js / io.js
- Intégration avec tout :
    - Builders : grunt, gulp

- Runtime : Node, Require JS
- Frameworks, IDE, outils de test

Christophe se veut rassurant. Babel est mature. Il compile intégralement les lignes de code de Facebook

ES6, pourquoi ?

- Plus facile : moins de piège, mode strict, isolation de mauvaise pratique
- Plus puissant
- Plus fiable
- Plus performant

Objets & Classes

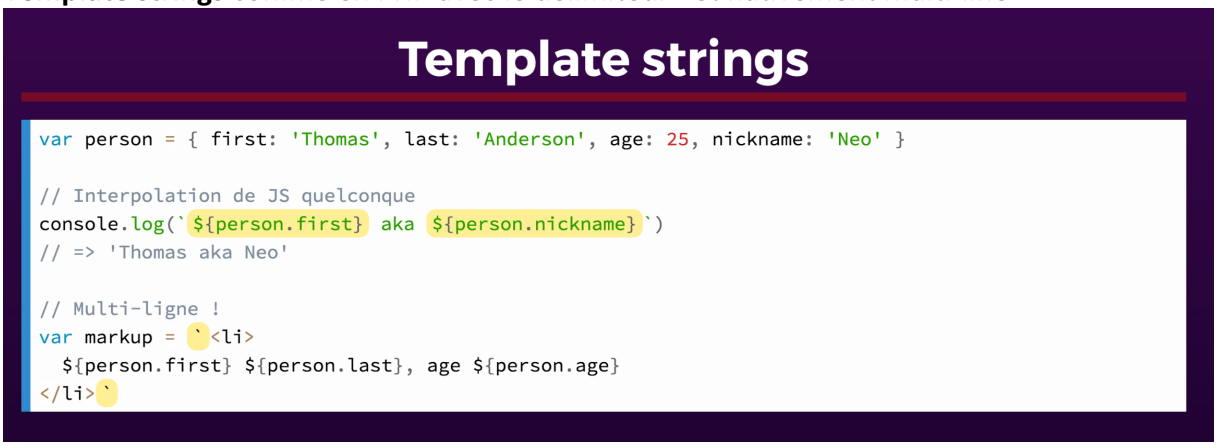
- Mêmes fonctionnalités mais nouveaux objets
- Littéraux objets
- Propriétés calculées
- Mots clés : class, extends, constructor, super, static. Oblige l'appel au constructeur hérité (ex : super(props))
- Accesseurs transparents comme en C# et Delphi : get et set

Fonctionnalités

- **Déstructuration** : on arrive à des paramètres nommés

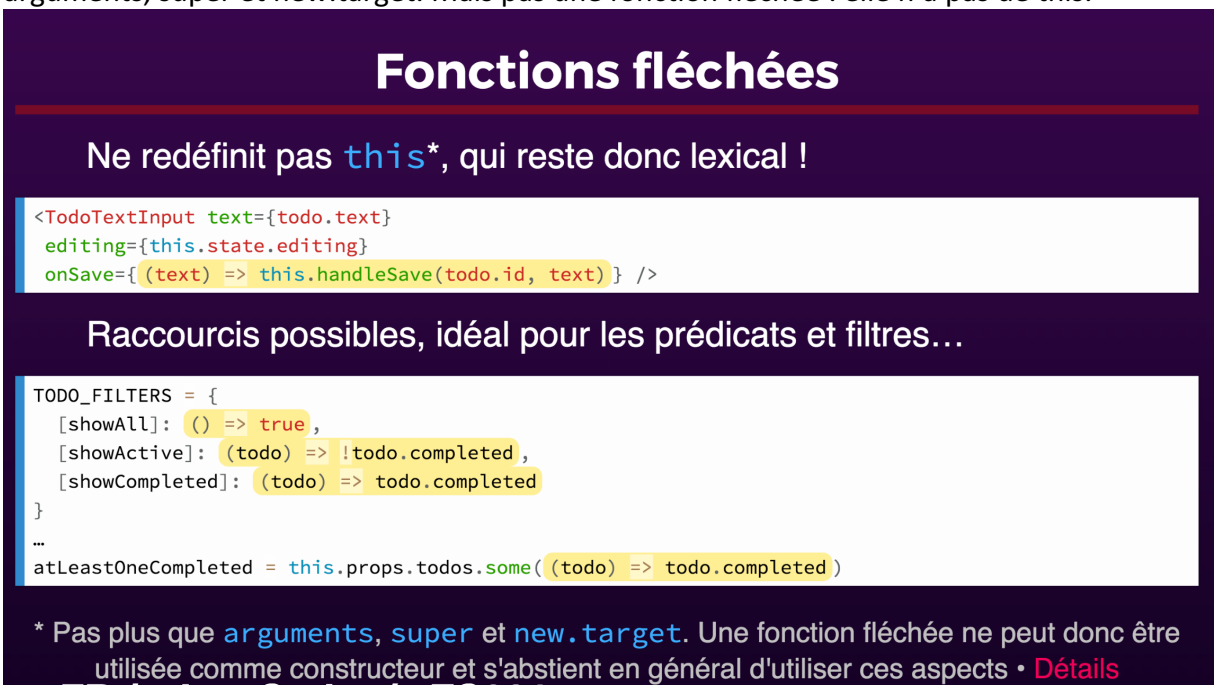
- **Rest & Spread** : le … comme en Java.

- **Valeur par défaut**, se déclenche en cas d'undefined. La valeur par défaut peut être un appel de fonctions ou être composés des autres paramètres
- **Template strings** comme en PHP avec le délimiteur ` et nativement multi-line

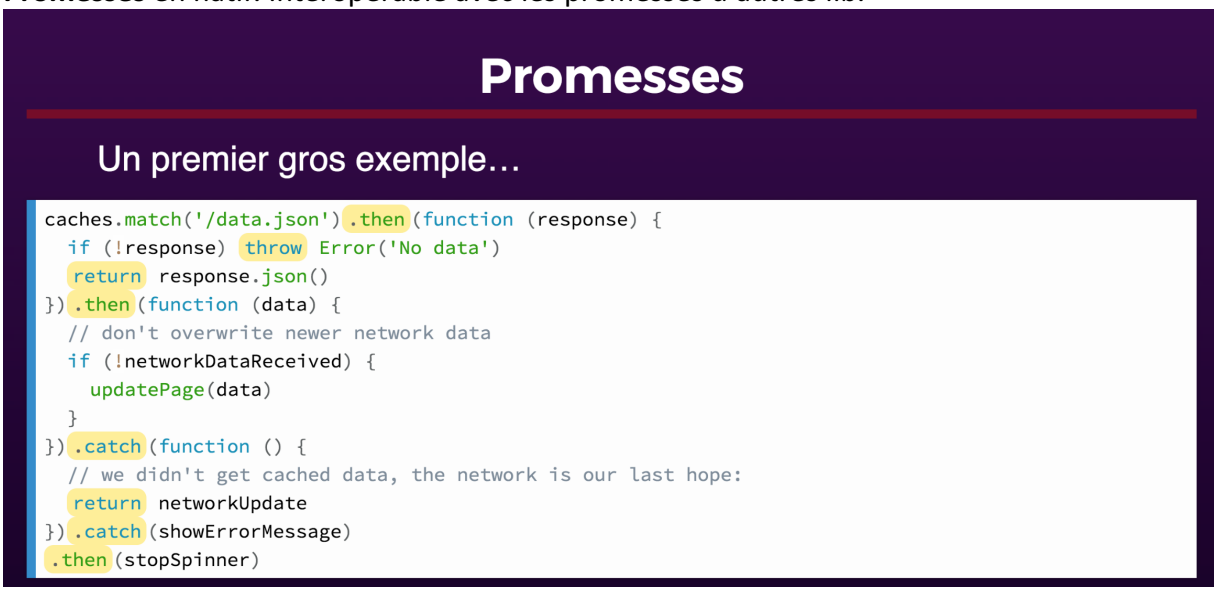


- Le mot clé var a disparu. Utiliser **let & const**. En général on utilise const car les variables ne changent pas de valeur. Let est réservé aux boucles. Attention const <> d'immutable (deep freeze)
- **Litéraux étendus** : octaux/binaires, unicode avec 2 codes caractères UTF-16

- **Fonctions fléchées** : elles ne bind pas le this !! Une fonction normale redéfinit this, arguments, super et new.target. Mais pas une fonction fléchée : elle n'a pas de this.



- De vrais **modules**
  - Reprend tout ce qui était bien dans les autres systèmes de module
  - 1 module = 1 fichier (comme CommonJS)
- **Chargement asynchrone** (Loader API) : pas encore bien pris en charge
- **Promesses** en natif. Interopérable avec les promesses d'autres lib.



- Mots clés **async** / **await** piqué à C# 5. Complémente les promesses sans les remplacer. Code asynchrone non bloqué. On peut récupérer facilement les exceptions => ES 2017 déjà supporté par Babel
- **Proxies** (ES207) : interception possible de toute manipulation d'un objet. Non supporté par Babel. Pattern decorator. Permet de lever une ReferenceError.
- **Décorateurs** @autobind, @memoize, @override (ES207) : va plus loin que les annotations

Le projet Lebab permet d'identifier du code ES5 éligible à du ES6 (l'inverse de Babel).

Question sur TypeScript : très bien, pour les gens qui ont besoin de typage fort (les développeurs qui viennent de langages fixes). En alternatif : flow est un type checker intéressant. Les syntaxes sont presques ISO. La bascule est facile.