EVEROAD
ex Convargo

Cycle de vie d'une fonctionnalité en JavaScript

Xavier Dutreilh

JavaScript

- **Langage** de programmation **interprété**, **faiblement typé** et **dynamique**, à base de **prototype**, et supportant la **programmation objet**, **impérative**, et **fonctionnelle**
- **Invention** de **Brendan Eich** chez **Netscape** pour **Navigator** en **1995**
- Initialement, nommé **Mocha**, puis **LiveScript**, puis **JavaScript**
- **Standardisation** par l'**Ecma International** sous l'appellation **ECMAScript** (ECMA-262) en **1997**
    - **Implémentations majeures :**
        - **SpiderMonkey** par **Netscape** pour **Navigator**, puis **Mozilla** pour **Firefox**
        - **V8** par **Google** pour **Chrome** (**indirectement** pour **Opera** et **Node.js**)
        - **JavaScriptCore** par **Apple** pour **Safari**
        - **Chakra (JScript)** par **Microsoft** pour **Internet Explorer**
        - **Chakra (JavaScript)** par **Microsoft** pour **Edge**
        - **ActionScript** par **Adobe Systems** pour **Flash**
        - **QtScript** par **The Qt Company** pour **Qt**
- **Confusion** fréquente entre **JavaScript** et **ECMAScript**
    - **JavaScript** : **marque déposée** par **Netscape**, puis **Sun**, puis **Oracle**
    - **ECMAScript** : **marque déposée** par l'**Ecma International**

TC39

- **Comité** en **charge** de la **standardisation** du langage **ECMAScript** depuis **1996**
- **Composition** très **instable**, très **dépendante** des **agendas** (**politiques**) de ses **membres**
  - **Président** : **Rex Jaeschke**
  - **Vice-Présidents** : **Leo Balter** et **Daniel Ehrenberg**
  - **Secrétaire** : **Istvan Sebestyen**
  - **Membres majeurs** :
    - Les **fournisseurs** de **navigateur** comme **Apple**, **Google**, **Microsoft**, et **Mozilla**
    - Les **entreprises technologiques** comme **Facebook**, **IBM**, **Intel**, et **PayPal**
    - Les **universités publiques** comme l'**Imperial College London** et l'**Indiana University**
    - Les **laboratoires** de **recherche** comme l'**Inria**
- **Réunion** des **délégués** deux fois par mois avec la **présence** ponctuelle d'**experts**
- **Modularisation** du langage **ECMAScript** (ECMA-414) depuis **2016**
  - **Spécifications couvertes** :
    - **ECMAScript** (ECMA-262)
    - **ECMAScript Internationalization API** (ECMA-402)
    - **JSON Data Interchange Format** (ECMA-404)
    - **ECMAScript Test Suite** (ECMA TR/104)

Historique

ECMAScript

- **Version 1.0** (**ES1**) en **1997**
- **Version 2.0** (**ES2**) en **1998**
- **Version 3.0** (**ES3**) en **1999**
- ~~**Version 4.0** (**ES4**) en **2008**~~
- **Version 5.0** (**ES5**) en **2009**
- **Version 5.1** (**ES5.1**) en **2011**
- **Version 6.0** (**ES2015**) en **2015**
- **Version 7.0** (**ES2016**) en **2016**
- **Version 8.0** (**ES2017**) en **2017**
- **Version 9.0** (**ES2018**) en **2018**
- **Version 10.0** (**ES2019**) en *cours de spécification*

Historique

ECMAScript

- **Version 1.0** (**ES1**) en **1997**
- **Version 2.0** (**ES2**) en **1998**
- **Version 3.0** (**ES3**) en **1999**
- ~~**Version 4.0** (**ES4**) en **2008**~~
- **Version 5.0** (**ES5**) en **2009**
- **Version 5.1** (**ES5.1**) en **2011**
- **Version 6.0** (**ES2015**) en **2015**
- **Version 7.0** (**ES2016**) en **2016**
- **Version 8.0** (**ES2017**) en **2017**
- **Version 9.0** (**ES2018**) en **2018**
- **Version 10.0** (**ES2019**) en *cours de spécification*

JavaScript (**Netscape**, puis **Mozilla**)

- **Version 1.0** en **1996**
- **Version 1.1** en **1996**
- **Version 1.2** en **1997**
- **Version 1.3** en **1998** (**ES1** + **ES2**)
- **Version 1.4** en **1999**
- **Version 1.5** en **2000** (**ES3**)
- **Version 1.6** en **2005**
- **Version 1.7** en **2006**
- **Version 1.8** en **2008**
- **Version 1.8.1** en **????**
- **Version 1.8.2** en **2009**
- **Version 1.8.5** en **2010** (**ES5**)

Historique

ECMAScript

- **Version 1.0** (**ES1**) en **1997**
- **Version 2.0** (**ES2**) en **1998**
- **Version 3.0** (**ES3**) en **1999**
- ~~**Version 4.0** (**ES4**) en **2008**~~
- **Version 5.0** (**ES5**) en **2009**
- **Version 5.1** (**ES5.1**) en **2011**
- **Version 6.0** (**ES2015**) en **2015**
- **Version 7.0** (**ES2016**) en **2016**
- **Version 8.0** (**ES2017**) en **2017**
- **Version 9.0** (**ES2018**) en **2018**
- **Version 10.0** (**ES2019**) en *cours de spécification*

JScript (**Microsoft**)

- **Version 1.0** en **1996**
- **Version 2.0** en **1997**
- **Version 3.0** en **1997** (**ES1**)
- **Version 4.0** en **????**
- **Version 5.0** en **1999** (**ES2**)
- **Version 5.1** en **????**
- **Version 5.5** en **2000** (**ES3**)
- **Version 5.6** en **2001**
- **Version 5.7** en **2006**
- **Version 5.8** en **2009**
- **Version 9.0** en **2011** (**ES5**)

test262

- **Suite** de tests **automatisée**, **standardisée** mais **évolutive**, à base de **JavaScript** et de **Python**, et permettant de **vérifier** la **conformité** d'une **implémentation** par rapport au **standard**
- **Chantier** du **TC39** démarré en **2010** avec des **contributions majeures** de **Google** et **Microsoft**
  - **Spécifications couvertes** :
    - **ECMAScript** (ECMA-262)
    - **ECMAScript Internationalization API** (ECMA-402)
    - **JSON Data Interchange Format** (ECMA-404)
- **30 701 tests** définis **aujourd'hui**
  - **Navigateurs majeurs** :
    - **Firefox 59** : **87%** des **tests passés** avec **succès**
    - **Chrome 65** : **93%** des **tests passés** avec **succès**
    - **Opera 52** : **93%** des **tests passés** avec **succès**
    - **Safari 11** : **76%** des **tests passés** avec **succès**
    - **Edge 41** : **68%** des **tests passés** avec **succès**
- **Confusion** fréquente entre **couverture** et **qualité**
  - **Couverture** : **nombre** de **fonctionnalités implémentées**
  - **Qualité** : **complexité algorithmique** et **maintenabilité** de l'**implémentation**

ECMAScript 2015

- **Sixième version** du langage **ECMAScript** publiée au mois de juin **2015**
  - **Fonctionnalités ajoutées** :
    - **Variables** et **Scoping**
    - **Arrow Functions**
    - **Extended Parameter Handling**
    - **Template Literals**
    - **Extended Literals**
    - **Enhanced Regular Expression**
    - **Enhanced Object Properties**
    - **Destructuring Assignment**
    - **Modules** et **Classes**
    - **Symbols**, **Iterators**, et **Generators**
    - **Typed Arrays**, **Maps**, et **Sets**
    - **New Built-In Methods**
    - **Promises**
    - **Meta-Programming**
    - **Internationalization** et **Localization**

ECMAScript 2016

- **Septième version** du langage **ECMAScript** publiée au mois de juin **2016**
  - **Fonctionnalités ajoutées** :
    - **Array.prototype.includes()** par **Domenic Denicola** et **Rick Waldron**
    - **Exponentiation operator (**)** par **Rick Waldron**

ECMAScript 2017

- **Huitième version** du langage **ECMAScript** publiée au mois de juin **2017**
  - **Fonctionnalités majeures** :
    - **Async functions** par **Brian Terlson**
    - **Shared memory and atomics** par **Lars T. Hansen**
  - **Fonctionnalités mineures** :
    - **Object.entries()** et **Object.values()** par **Jordan Harband**
    - **String.prototype.padStart()** et **String.prototype.padEnd()** par **Rick Waldron**
    - **Object.getOwnPropertyDescriptors()** par **Jordan Harband** et **Andrea Giammarchi**
    - **Trailing commas in function parameter lists and calls** par **Jeff Morrison**

ECMAScript 2018

- **Neuvième version** du langage **ECMAScript** prévue pour le mois de juin **2018**
  - **Fonctionnalités majeures** :
    - **Asynchronous Iteration** par **Domenic Denicola** et **Kevin Smith**
    - **Rest/Spread Properties** par **Sebastian Markbåge**
  - **Fonctionnalités ajoutées** aux **expressions régulières** :
    - **RegExp named capture groups** par **Gorkem Yakin** et **Daniel Ehrenberg**
    - **RegExp Unicode Property Escapes** par **Mathias Bynens**
    - **RegExp Lookbehind Assertions** par **Gorkem Yakin**, **Nozomu Katō**, et **Daniel Ehrenberg**
    - **s (dotAll) flag for regular expressions** par **Mathias Bynens**
  - **Fonctionnalités mineures** :
    - **Promise.prototype.finally()** par **Jordan Harband**
    - **Template Literal Revision** par **Tim Disney**

Proposition

- **Stage 0** - **Strawman** : la **proposition** est **décrite** sous **forme** d'**ébauche**. Elle est **formulée** par un **membre** du **TC39** ou un **contributeur** du **TC39**. Elle doit être **validée** en **réunion** de **travail**.
- **Stage 1** - **Proposal** : la **proposition** est **décrite** de **manière** plus **formelle**. Le **TC39** peut **aider** à la **conception** de la **proposition**. Il **nomme** également un **champion**, un **membre responsable** de la **proposition**. Des **polyfills** et des **démonstrations** sont **nécessaires** pour **passer** au **prochain stage**. A ce stade, des **changements majeurs** peuvent **encore survenir**.
- **Stage 2** - **Draft** : la **première version** de la **proposition** est **terminée**. Le **TC39** s'**attend** à **recevoir** une **description formelle** (**même partielle**) de la **syntaxe** et de la **sémantique**. **Deux implémentations expérimentales** (dont **une** dans un **transpiler**) sont **nécessaires** pour **passer** au **prochain stage**. A ce stade, seuls des **changements mineurs** peuvent **encore survenir**.
- **Stage 3** - **Candidate** : la **proposition finale** est **presque terminée**. Le **TC39** s'**attend** à **recevoir** des **retours** d'**expérience**. **Deux implémentations réelles** sont **nécessaires** pour **passer** au **prochain stage**. A ce stade, seuls des **changements critiques** de **dernière minute** peuvent **encore survenir**.
- **Stage 4** - **Finished** : la **proposition finale** est **terminée**. Elle sera **introduite** dans **une prochaine version** d'**ECMAScript**. Des **tests** pour **test262** sont **nécessaires** et ils doivent **passer** avec **succès** sur **deux implémentations réelles**.

Babel

- **Programme** de **traduction** d'un **code source écrit** en **JavaScript vers** du... **Javascript**
  - Idéalement, **deux versions différentes** de **JavaScript**
- **Invention** de **Sebastian McKenzie** sur son **temps libre** en **2014**
- Initialement, nommé **6to5**, puis **Babel**
- **Passage** à une **gestion** de **projet communautaire** en **2015**
- **Fonctionnement** à base de **plugins** (depuis la **version 5.0**) et de **presets** (depuis la **version 6.0**)
  - **Presets officiels** :
    - **env** : **traduction** d'un **code source écrit** en **JavaScript** (**ES*YYYY***) **vers** du **JavaScript** (**ES5**)
      - **Traduction possible** vers une **autre version** de **JavaScript** (**ex** : **ES2016**, **ES2017**)
    - **react** : **traduction** d'un **code source écrit** en **JSX vers** du **JavaScript** (**ES5**)
    - **flow** : **traduction** d'un **code source écrit** en **Flow vers** du **JavaScript** (**ES5**)
- **Confusion** fréquente entre **plugin** et **preset**
  - **Plugin** : **module** de **transformation** à **exécuter** pour **simuler** une **fonctionnalité** (**ex** : **arrow functions**, **classes**, **templates literals**, **async functions**)
  - **Preset** : **ensemble ordonné** de **plugins** à **exécuter** pour **simuler** un **ensemble** de **fonctionnalités** (**ex** : **ECMAScript 2015**, **ECMAScript 2016**, **ECMAScript 2017**, **ES.Next**)

Recommandation

- **Apprendre** à **utiliser** le langage **JavaScript** et **suivre** ses **évolutions** dans le temps
  - **En particulier** :
    - **Surveiller** les **publications annuelles** du **TC39**, au mois de juin de chaque année
    - **Expérimenter** les **nouvelles fonctionnalités** sur des **prototypes**
    - **Jouer** avec **Babel REPL** dans votre navigateur (babeljs.io/repl)
    - **Surveiller** les **fonctionnalités** en **cours** de **proposition** (prop-tc39.now.sh)
    - **Contribuer** aux **fonctionnalités** en **Stage 0 - Strawman**
- **Identifier** les **plateformes cibles** à **supporter** dans les projets
  - **Si besoin** :
    - **Utiliser Babel** pour **bénéficier** du **support** des **fonctionnalités manquantes**
    - **Configurer Babel correctement** en **choisissant** les **bons plugins** et **presets**
    - **Privilégier** les **plugins** et **presets officiels**
    - **Ne jamais utiliser** en **production** des **fonctionnalités** en **Stage 0 - Strawman**, **Stage 1 - Proposal**, et **Stage 2 - Draft**
    - **Faire** des **mises à jour** fréquentes de **Babel**, de ses **plugins** et de ses **presets**
- **Expérimenter** d'autres **langages** de **programmation objet**, **impérative**, et **fonctionnelle**