JavaScript, les notions basiques

Alvin Berthelot

Version 1.0.0

Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons.

Attribution - Partage dans les Mêmes Conditions 3.0 non transposé.

La licence, ses explications ainsi que les moyens de contribution et réappropriation sont détaillés à la fin.

L’histoire de JavaScript

Création du langage

- 1995 : Netscape crée JavaScript en partenariat avec Sun.
- 1996 : Microsoft implémente le même langage sous le nom de JScript.
- 1997 : Netscape standardise JavaScript sous le nom d'ECMAScript (Java étant une marque déposée) ayant comme standard ECMA-262.

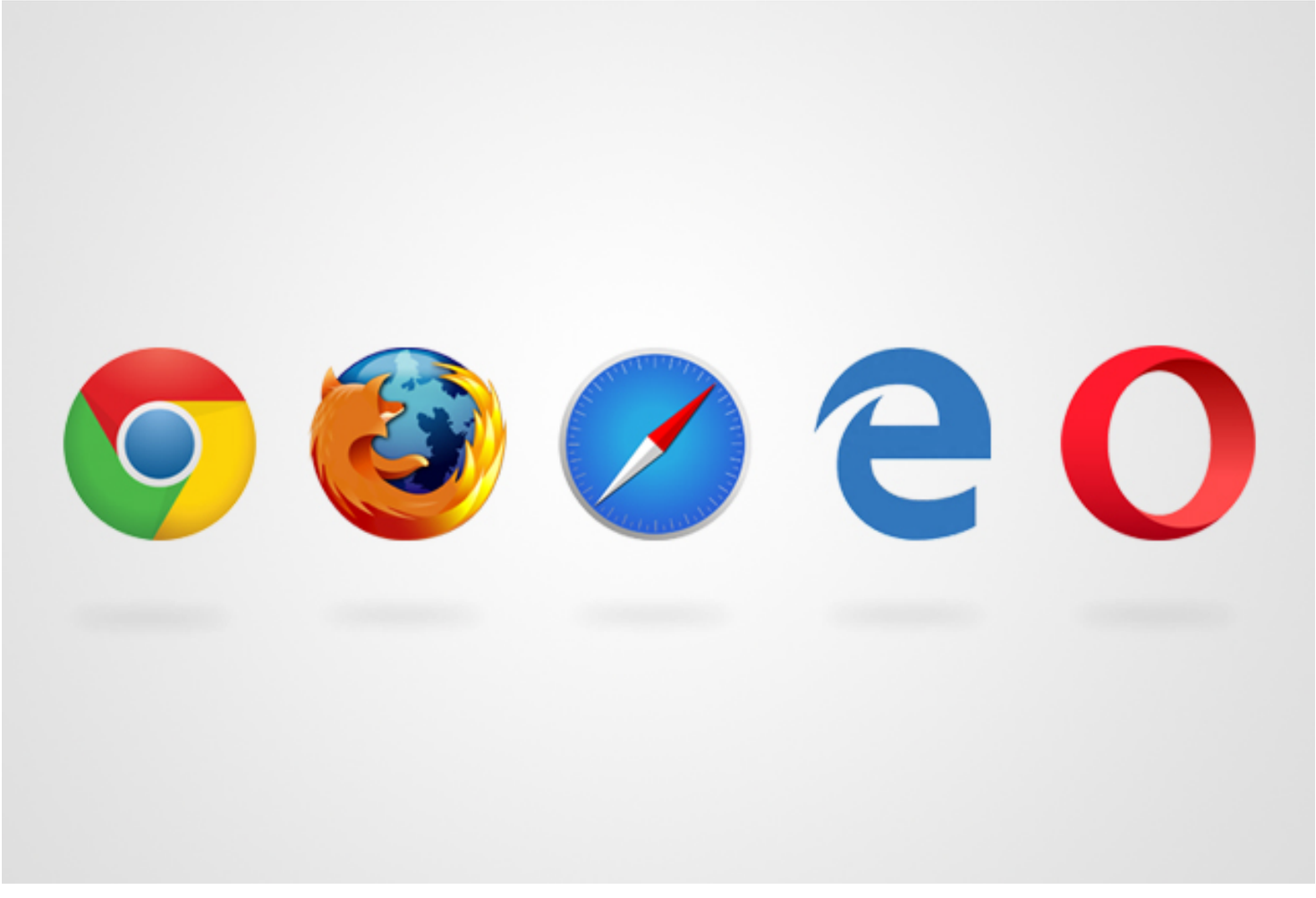
Les noms JavaScript et ECMAScript cohabitent encore, ECMAScript représentant plus le langage de programmation, tandis que JavaScript l'implémentation de celui-ci pour un navigateur Web.

Évolution du langage

- 2007 : ECMAScript 4 (ES4), version aujourd'hui abandonnée.
- 2009 : ECMAScript 5 (ES5), version tendant à disparaître mais encore grandement utilisée.
- 2015 : ECMAScript 6 (ES6 ou ES2015), version de référence aujourd'hui pour développer mais attention à la compatibilité navigateur.
- 2016 : ECMAScript 7 (ES7 ou ES2016), dernière version, extrême prudence sur la compatibilité.

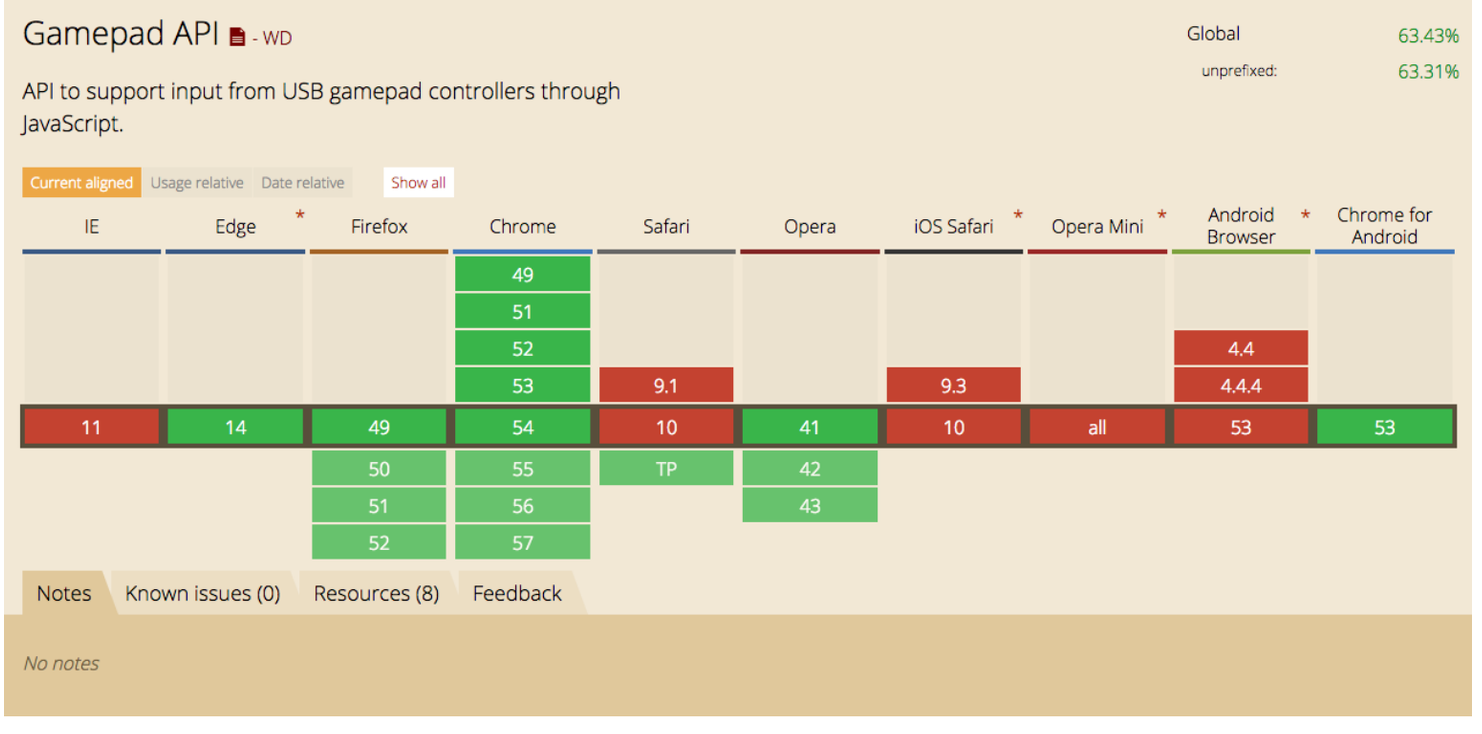
Le processus de normalisation est désormais axé sur une version par an.

Normalisation Vs Implémentation



"Can I Use" à la rescousse

Le site caniuse.com est votre boussole pour vous y retrouver dans ce que vous pouvez faire ou pas.



Compatibilité du langage

Pour assurer la compatibilité du langage, vous devez prendre en considération :

- Les différentes versions d'ECMAScript.
- Les différentes implémentations des navigateurs et leurs versions.
- Les navigateurs réellement utilisés par vos utilisateurs.

JavaScript, les moments forts

Si la normalisation du langage a été longue par moment, les cas d'utilisation de JavaScript eux n'ont cessé d'évoluer :

- 1995 : HTML dynamique
- 1999 : XMLHttpRequest
- 2001 : JSON
- 2005/2006 : Ajax & jQuery
- 2008/2009 : V8 & Node.js
- 2010 : Applications SPA

JavaScript, le vent en poupe

JavaScript est avant tout le langage de programmation universel des navigateurs Web, ce qui lui confère un positionnement stratégique.

Mais ces dernières années l'ont fait apparaître sur de nouvelles plateformes lui permettant d'être présent partout :

- Node.js
- Cordova/PhoneGap
- Electron
- Tessel

Les règles élémentaires

Les espaces et commentaires

L'interprétation du code source ne souffrira pas de la présence des caractères "espace". Utilisez les donc pour formater votre code JavaScript.

Pour les commentaires JavaScript propose deux possibilités, commenter un bloc de lignes `/* */` ou ligne par ligne `//`.

```
/*
console.log('Hello World !');
console.log('Bye Bye World !');
*/

// console.log('Hello World !');
// console.log('Bye Bye World !');
```

En bloc cela peut être fragilisé par le contenu commenté (s'il contient `*/` par exemple).

Les règles de nommage

Les règles de nommage en JavaScript sont assez souples, on peut utiliser les caractères suivants : `A-Za-z0-9_`

Par contre certains mots sont réservés au langage : `abstract boolean break byte case catch char class const continue debugger default delete do double else enum export extends false final finally float for function goto if implements import in instanceof int interface long native new null package private protected public return short static super switch synchronized this throw throws transient true try typeof var volatile void while with`

Pour le nommage, le mieux est encore d'appliquer une convention au sein de l'équipe (Convention recommandée par Douglas Crockford).

Les valeurs

Les types de valeur

JavaScript définit 6 types de valeur :

- Les chaînes de caractères : `string`
- Les nombres : `number`
- Les booléens : `boolean`
- Les objets : `object`
- Les fonctions : `function`
- Les valeurs indéfinies : `undefined`

Les chaînes de caractères

Une chaîne de caractères est une séquence de caractères encadrés par des apostrophes ' ou des guillemets ".

```
var message = 'Hello World !';
```

L'objet "wrapper" `String` apporte un ensemble de méthodes et propriétés pour manipuler les chaînes (Cf. MDN).

```
"Hello World !".length; // 13
```

On peut appliquer des opérateurs sur les chaînes de caractères mais il faut se rappeler qu'elles sont immuables.

Les nombres

JavaScript représente TOUS les nombres sous une forme unique : un nombre flottant sur 64 bits. Donc Pas de format "entier" spécifiquement défini mais cela n'empêche pas de travailler avec une précision garantie.

```
var sum = 40 + 2 // retourne 42
```

Pour les flottants la précision n'est pas garantie.

```
1.1 + 0.8 === 1.9 // false
1.2 + 0.7 === 1.9 // true
```

Idem un objet "wrapper" `Number` apporte un ensemble de méthodes et propriétés pour manipuler les nombres (Cf. MDN).

Les booléens

JavaScript représente également des booléens avec `false` et `true`, mais il connaît également des règles particulières pour d'autres valeurs :

Ce qui est faux : `false`, `null`, `''`, `0`, `NaN`, `undefined`.

Ce qui est vrai : tout le reste.

Si on y ajoute les conversions de type, cela peut devenir franchement déroutant.

```
"0" == false // true
"0" === false // false
```

Les objets

Les objets en JavaScript sont des ensembles de propriétés, elles-mêmes définies comme des paires clé/valeur où la clé est une chaîne de caractère et la valeur, toute valeur possible en JavaScript.

```
var cat = {
  "name": 'Grumpy',
  "age": 3,
  "male": true,
  "saySomething": function() {
    console.log('Not happy');
  }
};
```

Les objets sont bien plus complexes que cela, on y reviendra plus tard.

Les fonctions

Les fonctions en JavaScript permettent d'organiser et de regrouper des comportements au sein de notre code afin de les réutiliser.

Vous pourrez trouver deux formes de syntaxes :

```
function sayHello(name) {
  console.log('Hello ' + name);
};

var sayGoodbye = function(name) {
  console.log('Goodbye ' + name);
};

sayHello('Grumpy');
sayGoodbye('Grumpy');
```

Les fonctions sont bien plus complexes que cela, on y reviendra plus tard.

La valeur indéfinie

JavaScript permet d'indiquer qu'une valeur est indéfinie avec `undefined`.

La valeur `undefined` n'a pas la même signification que `null`, dans le premier cas il s'agit d'un type de valeur en soit dans le deuxième cas une affectation de valeur pour un type de valeur `object`.

```
undefined == null // true
undefined === null // false
```

L'opérateur `typeof`

JavaScript permet de connaître le type d'une valeur avec l'opérateur `typeof` en retournant une chaîne de caractères.

```
typeof('Hello World !'); // "string"
typeof(42); // "number"
typeof(false); // "boolean"
typeof({}); // "object"
typeof(function(){}); // "function"
typeof(undefined); // "undefined"

// LE piège
typeof(null); // "object"
```

Les tableaux

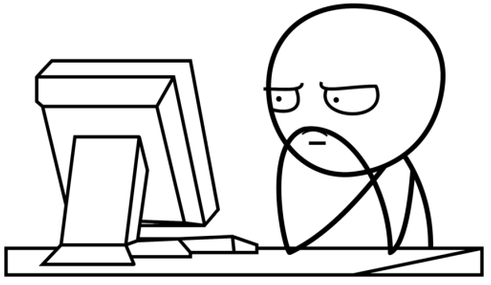
Et les tableaux dans tout ça ? JavaScript permet bien évidemment d'utiliser des tableaux, ceux-ci sont définis comme des types `object`, pouvant contenir eux mêmes n'importe quel type de valeur.

```
var meltingTypes = ['Hello World !', 42, function() {console.log('FUNCTION !')}];

meltingTypes[1]; // 42

typeof(meltingTypes); // "object"
typeof(meltingTypes[0]); // "string"
```

S'agissant d'un objet, ils disposent de nombreuses propriétés et méthodes permettant de les manipuler (Cf. MDN).



Exercices

On va s'appuyer sur la plateforme NodeSchool.io, pour cela vous devez installer le TP "javascripting".

⇒ INTRODUCTION
⇒ VARIABLES
⇒ STRINGS
⇒ STRINGS LENGTH
⇒ NUMBERS
⇒ ROUNDING NUMBERS
⇒ NUMBERS TO STRING

Les instructions

Les différents types d'instructions

JavaScript dispose de différents types d'instructions :

- L'instruction déclarative : `var`
- Les instructions conditionnelles : `if`, `switch`
- Les instructions itératives : `while`, `for`, `do`
- Les instructions d'interruption : `break`, `return`, `throw`

L'instruction déclarative `var`

L'instruction déclarative `var` permet de conserver l'état d'une valeur.

```
var message = 'Hello World !';
```

L'instruction conditionnelle `if`

L'instruction conditionnelle `if` (et son pendant `else`) permet d'orienter les instructions d'un programme selon l'évaluation d'une condition.

```
if(score > 10)
  message = 'You win';
else
  message = 'Try again';

if(score > 10) {
  message = 'You won';
  newtLevel = true;
}
```

L'instruction conditionnelle `switch`

L'instruction conditionnelle `switch` évaluer une expression et lui associe les instructions correspondantes.

```
switch (background) {
  case 'white':
    color = '#FFFFFF';
    break;
  case 'blue':
    color = '#0000FF';
    break;
  case 'red':
    color = '#FF0000';
    break;
  default:
    color = '#000000';
}
```

Les instructions itératives `while`, `for` et `do`

Les instructions itératives permettent de répéter les instructions selon certaines conditions.

```
while(ifPresent) {
  console.log('Hello, I\'m here');
}

for (var ite = 0; ite < names.length; ite++) {
  console.log('Hello ' + names[i]);
}

do {
  console.log('Hello, I\'m here');
} while (ifPresent);
```

Les instructions d'interruption `break`, `return` et `throw`

Les instructions d'interruption permettent d'interrompre l'exécution d'un programme selon certaines conditions.

- `break` : terminer l'exécution d'une boucle ou d'une instruction `switch`.
- `return` : terminer l'exécution d'une fonction et définir la valeur à retourner.
- `throw` : lever une exception et terminer l'exécution de la fonction courante pour remonter la file d'appels jusqu'au premier bloc `catch`.

```
var getWelcomeCustomerMessage = function(customer) {
  if(!customer.name)
    throw new ExceptionCustomer('Customer has no name.');
  return 'Welcome in our store dear ' + customer.name;
}
```

Les opérateurs

Les opérateurs d'affectation

JavaScript dispose de plusieurs opérateurs d'affectation (Cf. MDN).

Néanmoins le plus courremment utilisé est =.

```
var magicNumber = 42;
```

Pour des affectations par défaut on utilise l'opérateur binaire ||.

```
var target = /*w3c*/ e.target || /*IE*/ e.srcElement;
```

Les opérateurs de comparaison

JavaScript dispose de plusieurs opérateurs de comparaison (Cf. MDN).

Voici les principaux :

- `==` : égalité, `===` : égalité stricte, `!=` : inégalité et `!==` : inégalité stricte.
- `>` : supériorité stricte et `>=` : supériorité ou égalité.
- `<` : infériorité stricte et `<=` : infériorité ou égalité.

```
console.log('42' == 42); // true
console.log('42' === 42); // false
```

L'égalité stricte `===` et l'inégalité stricte `!==` sont fortements conseillées pour s'éviter bien des mauvaises surprises.

Les opérateurs arithmétiques

JavaScript dispose de plusieurs opérateurs arithmétques (Cf. MDN).

Voici les principaux :

- `*` : multiplier
- `/` : diviser
- `%` : modulo
- `+` : additionner ou concaténer
- `-` : soustraire ou négation

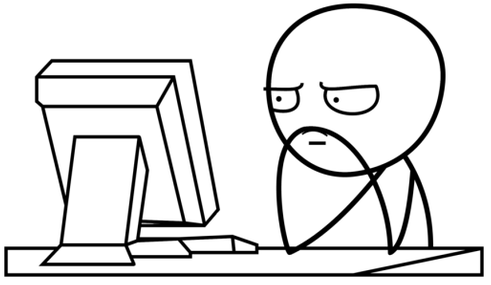
```
console.log('Magic number is ' + 6 * 7);
```

Les opérateurs logiques

JavaScript dispose des opérateurs logiques classqiues (Cf. MDN).

- `&&` : ET logique
- `||` : OU logique
- `!` : NON logique

```
console.log('42' == 42 && !('42' === 42)); // true
```



Exercices

On va s'appuyer sur la plateforme NodeSchool.io, pour cela vous devez installer le TP "javascripting".

⇒ IF STATEMENT
⇒ FOR LOOP
⇒ ARRAYS
⇒ ARRAY FILTERING
⇒ ACCESSING ARRAY VALUES
⇒ LOOPING THROUGH ARRAYS

Licence

Pour toutes questions, réclamations ou remarques, merci d'envoyer un message à alvin.berthelot@webyousoon.com.

Explications licence CC BY-SA 3.0

Cette licence permet aux autres de remixer, arranger, et adapter votre œuvre, même à des fins commerciales, tant qu'on vous accorde le mérite en citant votre nom et qu'on diffuse les nouvelles créations selon des conditions identiques.

Cette licence est souvent comparée aux licences de logiciels libres, "open source" ou "copyleft".

Toutes les nouvelles œuvres basées sur les vôtres auront la même licence, et toute œuvre dérivée pourra être utilisée même à des fins commerciales.

C'est la licence utilisée par Wikipédia ; elle est recommandée pour des œuvres qui pourraient bénéficier de l'incorporation de contenu depuis Wikipédia et d'autres projets sous licence similaire.

Contribution et réappropriation

Ce fichier PDF est généré avec Asciidoctor à partir d'un dépôt Git se trouvant sous GitHub.

https://github.com/alvinberthelot/slides-js

Cela signifie que vous n'avez pas besoin de vous battre avec un fichier binaire (le PDF) pour **contribuer**, **vous réapproprier le contenu** ou **modifier le thème** de présentation.

Contribution

Vous voulez **contribuer au contenu** car :

- Il y a une erreur (ça arrive à tout le monde), de typographie, de compréhension, ou tout autre chose.
- Vous souhaitez apporter une précision.

Il vous suffit de contribuer au projet via Git par le moyen d'une "pull request" sur le dépôt Git.

Réappropriation

N'oubliez pas les conditions de la licence.

Vous voulez vous **réapproprier le contenu** car :

- Vous souhaitez donner un style différent.
- Vous souhaitez enlever/ajouter/modifier des sections dans votre contexte.

Il vous suffit de "forker" le dépôt Git et d'y apporter vos propres modifications, puis de générer par vous même le nouveau PDF.