

JS

perfectionnement

session du 06 avril 2021



Thomas Fritsch

thomas@uidlt.fr

Dev front & consultant

Enseignant & **formateur**

JS / React / React Native / Node.js



une
idée derrière
la **tech**

PROGRAMME

- 1. Rappels**
- 2. Environnement de debug**
- 3. ES6+**
4. POO
5. jQuery
6. Expressions régulières
7. Échange de données
8. APIs HTML5

JAVASCRIPT,
POURQUOI
TANT
DE
Haine ?

M.

, prof de JAVA à l'Université de Lille, le 14 mars 2018

“ *Franchement, JavaScript, c'est un langage
merdique !* ”

Mention a book that made you cry

PROGRAMMING JAVASCRIPT

**LIE
DOWN**



**TRY
NOT TO CRY**



**CRY
A LOT**



JavaScript

The Definitive Guide

POURQUOI ?

- pas de typage fort
- pas de POO
- scope incompréhensible
- "this" aléatoire
- pas d'outils de dev sérieux
- variables globales par défaut
 - pas de gestion des dépendances
 - ne marche pas dans tous les navigateurs
 - une sous-version pourrie de JAVA

ON VOUS **MENT !**

tout ça c'est du passé

1. UNIVERSEL

99% des terminaux compatibles

2. MULTI SUPPORT

browser / serveur / desktop / mobile / IOT

3. STANDARD DE L'INDUSTRIE...



Repositories

40

40 repository results

Sort: Most stars ▾

Code

Commits

Issues

Packages

Marketplace

Topics

Wikis

Users

Languages

JavaScript 13

Python 6

C++ 2

Go 2

Java 2

C 1

CSS 1

Dart 1

Jupyter Notebook 1

Rust 1

SUR LES 5 PREMIERS PROJETS GITHUB

★ 308k

3 UTILISENT JS

★ 249k

vuejs/vue

JavaScript

★ 155k

 Vue.js is a progressive, incrementally-adoptable JavaScript framework for building UI on the web.

[javascript](#) [vue](#) [framework](#) [frontend](#)

MIT license Updated 2 days ago 1 issue needs help

facebook/react

JavaScript

★ 142k

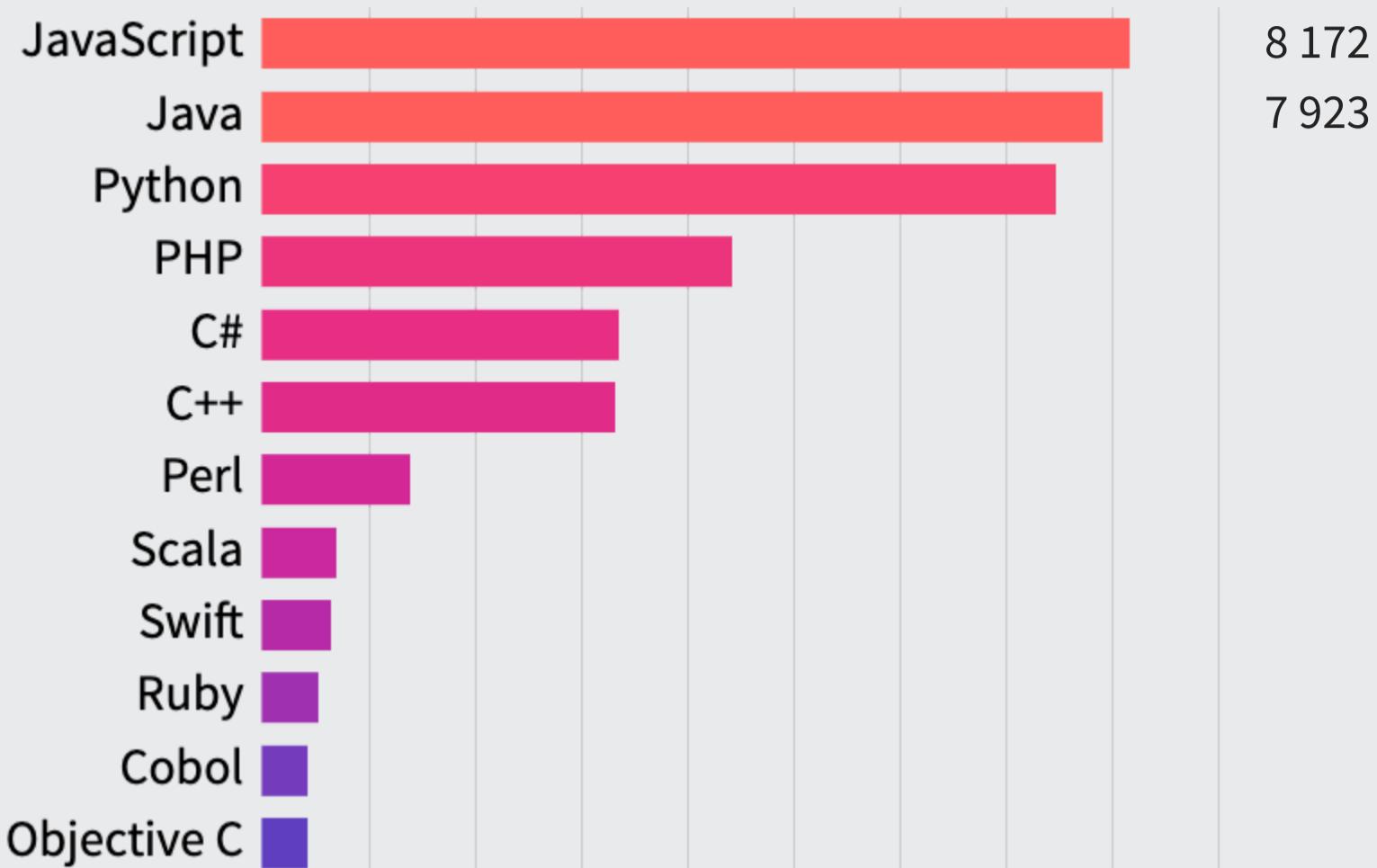
Notes :

Source : <https://github.com/search?o=desc&q=stars%3A%3E59500&s=stars&type=Repositories>

NB: n'est pas compté dans le classement le repo [996.ICU](#) dénonçant les conditions de travail dans certaines entreprises chinoises et qui n'est pas un projet de dev.

LANGAGES LES + DEMANDÉS

Offres d'emploi sur Indeed France - Jan. 2021



Notes :

Statistiques d'offres d'emploi en france sur indeed (janvier 2021) :

langage	source	nb emplois
javascript OR js OR typescript	https://fr.indeed.com/emplois?q=%22javascript%22+OR+%22js%22+OR+%22typescript%22&l=	8 172
java	https://fr.indeed.com/emplois?q=%22java%22&l=	7 923
python	https://www.indeed.fr/emplois?q=python&l=	7 468
PHP	https://fr.indeed.com/emplois?q=%22PHP%22&l=	4 428
c++	https://fr.indeed.com/emplois?q=%22c%2B%2B%22&l=	3 326
c#	https://www.indeed.fr/emplois?q=c%23&l=	3 361
Perl	https://www.indeed.fr/emplois?q=Perl&l=	1 401
Scala	https://www.indeed.fr/emplois?q=scala&l=	701
Swift	https://www.indeed.fr/emplois?q=swift&l=	660
Ruby	https://www.indeed.fr/emplois?q=Ruby&l=	535
Cobol	https://fr.indeed.com/emplois?q=%22cobol%22&l=	429
Objective C	https://www.indeed.fr/emplois?q=Objective+C&l=	426

[Premium](#)[Aide](#)[Télécharger](#)[S'inscrire](#)[Se connecter](#)

2.9

De la musique pour tous.

Des millions de titres. Aucune carte de crédit nécessaire.

[TÉLÉCHARGEZ SPOTIFY FREE](#)

Notes :

Spotify utilise JS pour le développement de son application Desktop

Source : <https://www.quora.com/How-is-JavaScript-used-within-the-Spotify-desktop-application-Is-it-packaged-up-and-run-locally-only-retrieving-the-assets-as-and-when-needed-What-JavaScript-VM-is-used/answer/Mattias-Petter-Johansson>

"The Spotify desktop client UI is completely built with JavaScript, resting on top of the same C++ core that the iOS and Android clients uses."

Tirez le meilleur parti de Skype

Découvrez pourquoi des centaines de millions de personnes utilisent Skype pour discuter et appeler chaque jour.



Appels audio et vidéo HD

Un précis, vidéo HD... Bénéficiez d'une qualité optimale pour tous vos appels de groupes ou individuels. Et réagissez avec des émoticônes animées !



Messagerie intelligente

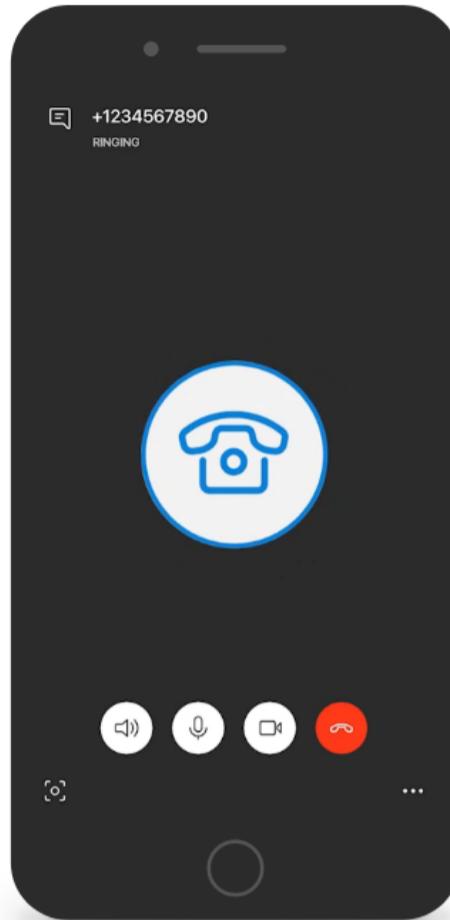
Repondez aux messages avec humour, grâce à la fonctionnalité de réaction. Attirez l'attention d'un contact avec la fonctionnalité @mentions.



Partage d'écran

Partagez facilement vos présentations, vos photos ou tout autre élément sur votre écran.

Le partage d'écran intégré est là pour ça.



Enregistrement d'appel et sous-titres en direct

Enregistrez vos appels Skype pour garder les moments. Notez les décisions importantes. Utilisez les sous-titres en direct pour afficher les mots prononcés.



Appels téléphoniques

Vous voulez joindre des amis sur leur ligne téléphonique ? C'est possible, avec un tarif d'international avantageux vers les mobiles et les fixes.



Conversations privées

Gardez vos conversations confidentielles privées avec notre chiffrement de bout en bout conforme aux normes en vigueur.

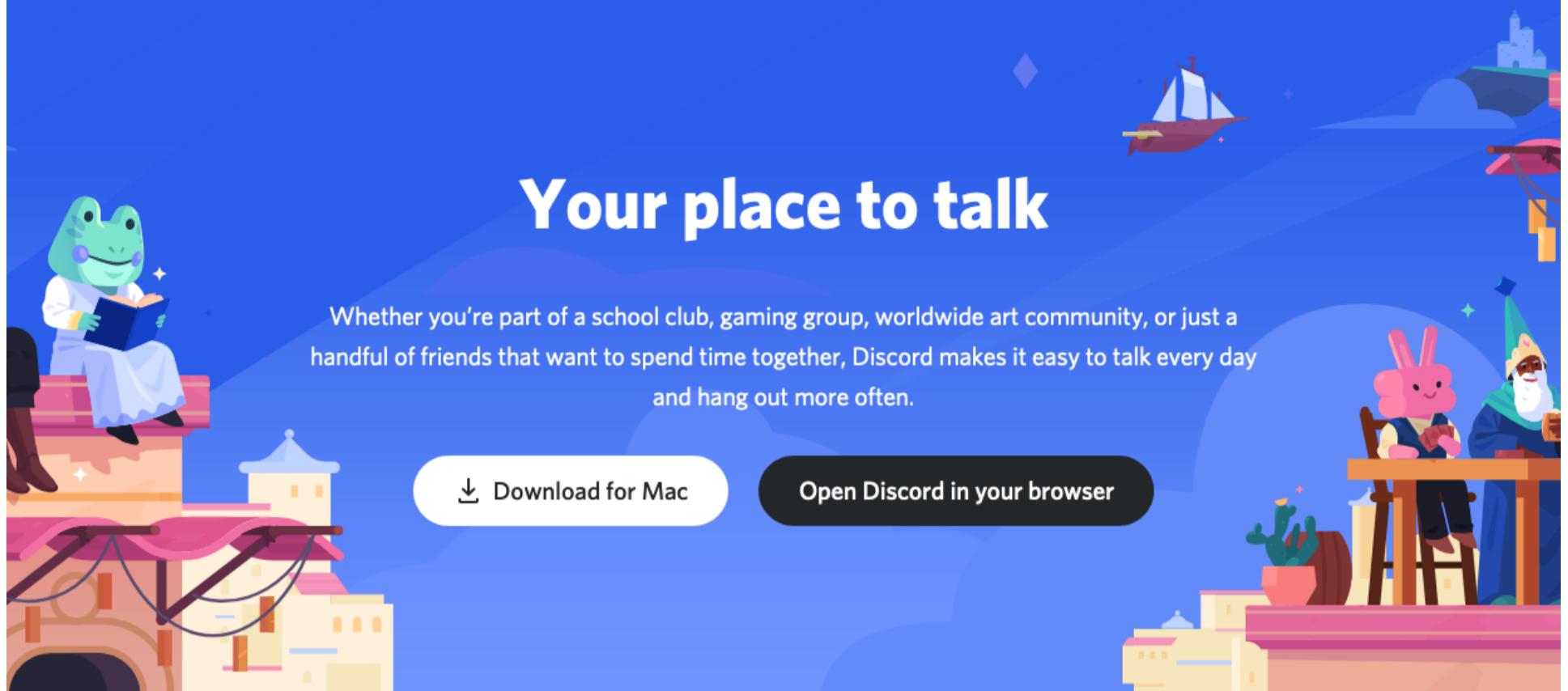
Un seul compte Skype pour tous vos appareils

Notes :

<https://www.skype.com/fr/features/>

Les applis Skype desktop et mobile sont codées en JS

Sources : <https://electronjs.org/apps/skype> & <https://facebook.github.io/react-native/showcase.html>

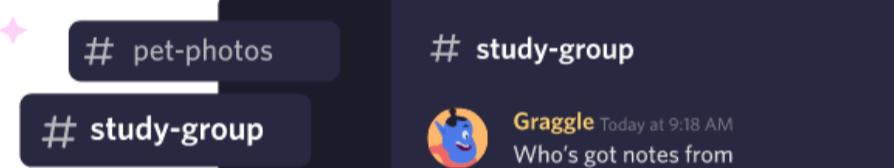


Your place to talk

Whether you're part of a school club, gaming group, worldwide art community, or just a handful of friends that want to spend time together, Discord makes it easy to talk every day and hang out more often.

[Download for Mac](#)

[Open Discord in your browser](#)



**An invite-only
place with plenty**

Notes :

Le client desktop de Discord ainsi que l'appli mobile sont aussi développés avec JavaScript (Electron)

Source : [https://en.wikipedia.org/wiki/Discord_\(software\)#cite_ref-15](https://en.wikipedia.org/wiki/Discord_(software)#cite_ref-15) & <https://facebook.github.io/react-native/showcase.html>

Chauffeurs

Passagers

Uber E

Works

Vélo

Uber Elevate

Prenez la route augmentez vos revenus

Rejoignez la plateforme bénéficiant du plus grand réseau de passagers actifs.

[S'inscrire en tant que chauffeur](#)[En savoir plus sur les courses et les livraisons](#)

Nous aidons plus de

Notes :

Uber utilise JavaScript côté serveur avec Node.JS

Source : <https://foundation.nodejs.org/wp-content/uploads/sites/50/2017/09/Nodejs-at-Uber.pdf>

Films, séries TV et bien plus en illimité.

Où que vous soyez. Annulez à tout moment.

Adresse e-mail

ESSAYEZ GRATUITEMENT PENDANT 30 JOURS >

Prêt à regarder Netflix ? Saisissez votre adresse e-mail pour créer votre compte ou y accéder

Regardez Netflix sur votre TV.

Regardez Netflix sur votre Smart TV.



Notes :

Les applis serveur de Netflix utilisent aussi Node.JS

Source : <https://medium.com/netflix-techblog/node-js-in-flames-ddd073803aa4>

mais alors...

CES GENS LÀ AIMENT-ILS LE CODE SALE ?

ECMASCRIPT ?

- la Spec suivie par JavaScript
- beaucoup ne connaissent qu'ES5 (2009)
- ES6 sorti en 2015 ! nouvelle syntaxe, nouvelles possibilités
- depuis, une nouvelle version / an : ES7, ES8, ES9, ES10...
- POO, gestion des dépendances, block scope, Promises/Futures, ...

JS

1. Rappels

- 2. Environnement de debug
- 3. ES6+
- 4. POO
- 5. jQuery
- 6. Expressions régulières
- 7. Échange de données
- 8. APIs HTML5

1. RAPPELS

- **Intégration**
 - Les types de données
 - Fonctions & chaîne de portée
 - API DOM
 - Les événements

HTML + JS : INTÉGRATION

Inline

```
<a href="#" onclick="alert('Welcome to Albuquerque');return false;">  
    BB  
</a>
```

Dans une balise script

```
<script>alert('Welcome to Albuquerque');</script>
```

Dans un fichier externe

```
<script src="welcome.js"></script>
```

HTML + JS : MAIS T'ES OÙ ?

```
<!doctype html>
<html>
<head>
  <title>J. P. Wynne High School Presentation</title>
  <meta charset="utf-8">
  <script src="build/main.js" defer></script> <!-- B : 🏆 -->
</head>
<body>
  <p>Public high school located in Albuquerque, New Mexico. It's the
      perfect place to steal methlab equipment.</p>
</body>
</html>
```

Notes :

Il ya plusieurs façons d'intégrer des balises script dans une page html mais depuis quelques temps déjà c'est l'attribut "defer" que je vous recommande d'utiliser sur une balise dans le <head>.

Avec cet attribut :

- le chargement du JS ne bloquera pas le parsing de la page HTML
- le code qu'il contient ne sera exécuté que quand la page sera intégralement chargée

Voir : <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script#attr-defer>

Presque toutes les versions en circulation de presque tous les navigateurs supportent cet attribut, il peut donc être utilisé sans soucis en production : <https://caniuse.com/#feat=script-defer>

IMPACT SUR L'UX

(SKYWALKER)

<script> à la fin du body

parsing HTML

chargement JS



<script> dans le head

parsing HTML

chargement JS



<script> dans le head + defer

parsing HTML

chargement JS



Notes :

L'intégration de l'attribut defer permet de ne pas bloquer le parsing de la page html, mais en même temps de n'exécuter son code qu'une fois que toute la page html est parsée pour éviter les bugs d'accès au contenu de la page.

Plus de précisions sur les différentes possibilités : <https://www.growingwiththeweb.com/2014/02/async-vs-defer-attributes.html>

1. RAPPELS

- Intégration
- **Les types de données**
- Fonctions & chaîne de portée
- API DOM
- Les événements

LE TYPE STRING

déclaration

```
/* guillemets simples */  
var s1 = 'je suis une chaîne avec des single quotes';  
  
/* ou guillemets doubles */  
var s2 = "je suis une chaîne avec des double quotes";
```

opérateur de concaténation : +

```
var episodeName = 'The door';  
var title = '<h1>' + episodeName + '</h1>';
```

Notes :

On peut déclarer les chaînes de caractères de plusieurs manières, avec des guillemets simples, doubles.

En revanche on évite d'utiliser l'instruction `new String('ma chaine')` car elle consomme plus de ressources et provoque des résultats inattendus notamment lors de la comparaison de deux chaînes de caractères.

On verra plus tard, qu'ES6 a apporté une nouvelle façon de déclarer les chaînes, beaucoup plus pratique !

LE TYPE STRING

Principales méthodes

```
var serie = 'Game Of Thrones';
console.log(
  serie.length, // 15 (longueur de la chaîne)

  serie.search( 'Of' ), // 5 (position de la chaîne recherchée)
  serie.search( 'Hodor' ), // -1 (chaîne introuvable)

  serie.split( ' ' ), // [ 'Game', 'Of', 'Throne' ] (découpe)

  serie.substring( 5, 7 ), // 'Of' (portion d'une chaîne)

  serie.toUpperCase() // 'GAME OF THRONES'
);
```

LE TYPE NUMBER

Attention à la conversion de type implicite

```
42 + 12      // 54
42 + true    // 43
42 + "12"    // "4212"
42 - "12"    // 30 !
```

LE TYPE ARRAY

déclaration

```
var emptyArray = [ ];  
  
var brothers = [ 'Robb', 'Bran', 'Rickon' ];
```

accès à une valeur

```
var littleBrother = brothers[ 2 ]; // "Rickon"
```

Méthodes

```
var brothers = [ 'Robb', 'Bran', 'Rickon' ];  
console.log(  
    brothers.length, // 3  
  
    brothers.join(' and '), // "Robb and Bran and Rickon"  
  
    brothers.push('Jon'), // 4 (nouvelle length)  
  
    brothers, // [ 'Robb', 'Bran', 'Rickon', 'Jon' ]  
  
    brothers.map(function(value){ return value.toUpperCase() } )  
        // [ 'ROBB', 'BRAN', 'RICKON', 'JON' ]  
);
```

OBJET LITTÉRAL

structure de données / dictionnaire (*HashMap*)

```
var o = {
    propriete: valeur,
    ...
}
```

```
var arya = {
    age: 9,
    name: 'No one',
    friend: {
        name: 'Mycah',
        isDead: true
    }
}

console.log(
    arya.name, // "No one"
    arya['na'+'me'], // "No one"
    arya.friend.isDead // true
);
```

Notes :

Au lieu d'utiliser la notation littérale pour déclarer des objets on peut aussi utiliser le constructeur de la classe Object :

```
var o = new Object();
o.propriete = valeur;
o['propriete'] = valeur;
```

Cette notation est beaucoup plus verbeuse que l'objet littéral, c'est la raison pour laquelle elle n'est presque jamais employée. Par exemple le code précédent pourrait donner quelque chose comme ceci avec le constructeur `new Object()` :

```
var arya = new Object();
arya.age = 9;
arya.name = 'No one';

var mycah = new Object();
mycah['name'] = 'Mycah';
var propertyName = 'Dead';
mycah['is' + propertyName] = true

arya.friend = mycah;
```

FONCTIONS

```
function makeEpisode(hero) { // fonction nommée  
    return hero + ' is dead !';  
}
```

```
var makeEpisode = function(hero) { //fonction anonyme  
    return hero + ' is dead !';  
}
```

```
var e = makeEpisode('Benjen Stark');
```

LA CHAÎNE DE PORTÉE

Référence à une variable :

1. Recherche de la variable dans la fonction courante
 2. Recherche dans la fonction qui a déclaré la fonction courante...
 3. ... et ainsi de suite
 4. jusqu'à la portée globale (objet window)
 5. Exception de type ReferenceError
-

Notes :

La chaîne de portée représente la disponibilité d'une variable dans notre code.

En JavaScript les variables déclarées avec `var` ne sont accessibles qu'au sein de la fonction dans laquelle elles sont définies ainsi que dans les fonctions qu'elle contient.

Nous verrons plus tard qu'avec ES6 sont apparues de nouvelles façons de déclarer les variables qui modifient ce comportement !

LA CHAÎNE DE PORTÉE : EXERCICE

```
function a() {  
    var i = 2 ;  
    function b( i ) {  
        var j = i;  
        return j - k;  
    }  
    var k = 42;  
    console.log( b( i*3 ) );  
    return j;  
}  
a();
```

Notes :

Le scope des `var` correspond à la fonction dans laquelle elles sont déclarées. Les fonctions qui ont été déclarés à l'intérieur de celle qui contient la `var` ont aussi accès aux valeurs.

En revanche, une fois la fonction terminée, la variable est détruite et inaccessible pour le reste du code !

Dans cet exercice, la console affiche en fait 2 choses :

- le résultat du `console.log(b(i*3))` à savoir : **-36**
- une exception "**Uncaught ReferenceError: j is not defined**" car la fonction a essaie d'accéder à la variable `j` (`return j`) qui n'existe qu'à l'intérieur de la fonction `b` !

Voir le résultat sur : <https://codepen.io/uidlt/pen/ZELewGj?editors=0011>

1. RAPPELS

- Intégration
- Les types de données
- Fonctions & chaîne de portée
- **API DOM**
- Les événements

DOM

Document Object Model

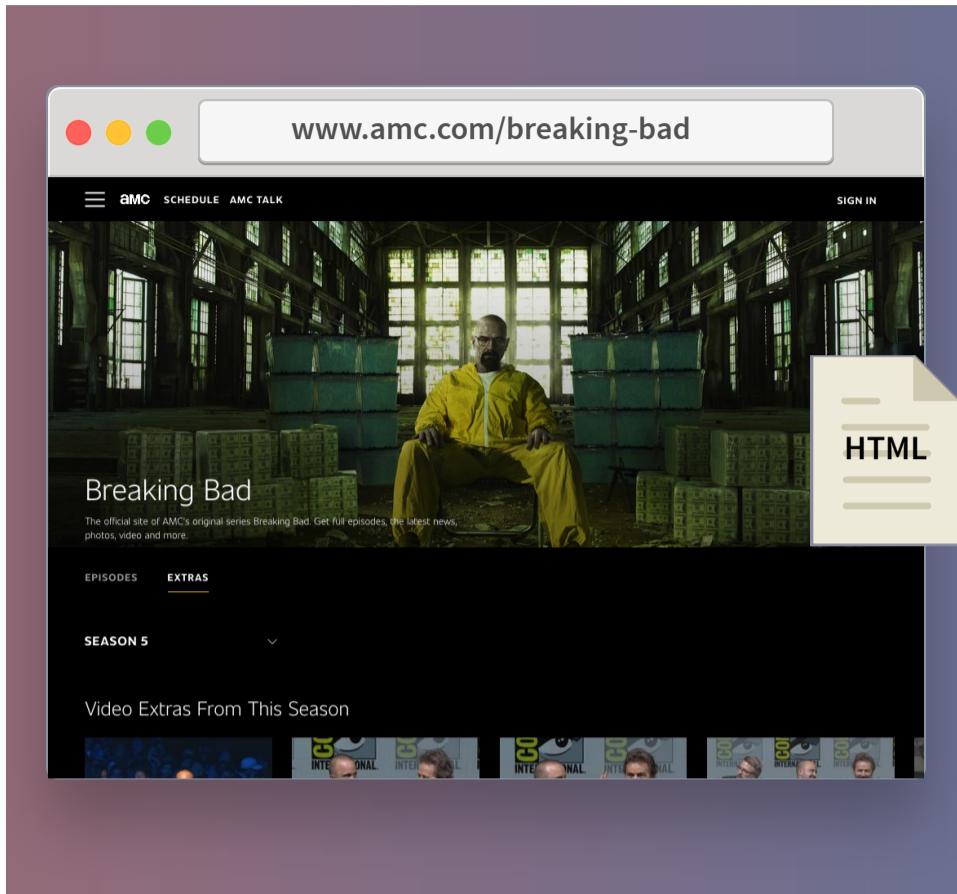
Notes :

L'API DOM est une API JS implémentée par les navigateurs

- Standard du W3C
- Permet de lire / modifier la page
- Chaque balise = un objet (branche/feuille)
- Le DOM = l'arbre des balises du document

<https://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html> Spec de l'API DOM sur le site du W3C

LES OBJETS DE L'API DOM



window :
fenêtre/onglet du navigateur

window.document :
document html complet (head & body)

window.document.body :
balise <body>

SÉLECTION : QUERYSELECTOR

```
var element = document.querySelector( selecteur );
```

Retourne le 1er élément qui correspond au sélecteur CSS

```
var elementsArray = document.querySelectorAll( selecteur );
```

Retourne un tableau de tous les éléments qui correspondent au sélecteur CSS

Notes :

L'API DOM offre 3 façons de sélectionner un élément de la page :

1. propriété `element.children`
2. méthodes `element.getElementsByTagName()`
3. méthode `element.querySelector()`

Comme nous le verrons par la suite, `querySelector` qui est la plus récente est celle que je vous recommande car elle est beaucoup plus pratique.

CSS : SÉLECTEURS DE BASE

```
div {...} /* type de balise */  
.dwarf {...} /* classe css */  
#king {...} /* id */
```

```
<div>Characters :</div>  
<div class="dwarf">Tyrion</div>  
<div id="king">Tommen</div>
```

CSS : SÉLECTEURS ENFANTS

```
section p {...} /* les <p> contenus dans <section> */
section > p {...} /* les <p> contenus immédiatement dans <section> */
```

```
<section>
  <p>Characters :</p>
  <div class="dwarf">
    Tyrion
    <p>best character ever !</p>
  </div>
  <div id="king">Tommen</div>
</section>
```

Notes :

querySelector vs getElementsByTagName :

```
var elements = document.querySelectorAll('#container .menu li');

var menus = document
  .getElementById('container')
  .getElementsByClassName('.menu');
var elements = [];
menus.forEach( menu => {
  elements = elements.concat( menu.getElementsByTagName('li') );
});
```

querySelector permet de simplifier grandement la sélection d'éléments de la page HTML. C'est aujourd'hui LA méthode recommandée pour accéder à des éléments (on n'utilise donc plus les méthodes getElementsByTagName).

Le support navigateur de querySelector est très bon (IE9+), il n'est plus justifié aujourd'hui de s'en passer : <http://caniuse.com/#feat=queryselector>

MODIFIER UN ÉLÉMENT

- **element.innerHTML** : lire/modifier le contenu html
- **element.getAttribute() / setAttribute()** : lire/modifier un attribut html

```
<div id="king">Joffrey</div>

<script>
  var king = document.querySelector( '#king' );
  console.log( king.innerHTML + ' is dead ! \u263a\u263a' );
  king.innerHTML = 'Tommen';
  king.setAttribute( 'class', 'willDieSoon' );
</script>
```

```
<div id="king" class="willDieSoon">Tommen</div>

<script>
  var king = document.querySelector( '#king' );
  console.log( king.innerHTML + ' is dead ! \u263a\u263a' );
  king.innerHTML = 'Tommen';
  king.setAttribute( 'class', 'willDieSoon' );
</script>
```

1. RAPPELS

- Intégration
- Les types de données
- Fonctions & chaîne de portée
- API DOM
- **Les événements**

ÉVÉNEMENTS

- détection en HTML : attributs `onload`, `onclick`...
- détection en JS : `element.addEventListener()`
- propriétés `target` et `currentTarget`
- méthode `event.preventDefault()`

```
<a href="#" onclick="onLinkClick();return false">Ceci est un lien</a>
<script>
    function onLinkClick(){
        console.log( 'on a cliqué sur le lien !' );
    }
</script>
```

```
<a href="#">Ceci est un lien</a>
<script>
    function onLinkClick(event){
        event.preventDefault();
        link.innerHTML = 'OMG, on m\'a cliqué dessus !';
    }
    var link = document.querySelector( 'a' );
    link.addEventListener('click', onLinkClick );
</script>
```

Notes :

Pour capter les événements déclenchés par le DOM on utilise des écouteurs d'événement, cela peut se faire de deux manières :

- via des attributs HTML (`onClick`, `onLoad`, `onChange`, `onSubmit`, ...)
- en programmation avec `addEventListener()`

En général on évite les attributs html `onXXXXX=""` car ils "polluent" le code HTML avec du code JS. La méthode `addEventListener()` a l'avantage de permettre de bien séparer les responsabilités entre le contenu (*le document HTML*) et l'applicatif (*le JS*).

HTML

```
1 <a class="button" href="https://www.hbo.com/game-of-thrones">
2 WHERE ARE  MY DRAGONS ?!</strong>
3 </a>
4
5 <section class="dragonsContainer">
6 <div class="dragon">Drogon</div>
7 <div class="dragon">Rhaegal</div>
8 <div class="dragon">Viserion</div>
9 </section>
```

CSS

```
1 .dragon {
2   display:none;
3 }
```

4
5
6
7
8
9
10
11
12

WHERE ARE MY DRAGONS ?!

???

Notes :

Exercice DOM API & event Listeners : Exemple de mise en oeuvre de l'API DOM pour :

1. détecter le clic sur un lien
2. afficher des éléments jusque là masqués

L'exercice : <https://codepen.io/uidlt/pen/VwjJJNp>

Ma proposition de solution : <https://codepen.io/uidlt/pen/vYKqqoJ>