DÉVELOPPEMENT JAVASCRIPT

PROGRAMME

1. Rappels
2. Premiers pas en JS
3. ES6 : le JavaScript nouveau
4. API DOM
5. Ajax
6. jQuery
7. La POO en JS
8. Pour aller plus loin...

THOMAS FRITSCH

thomas@kumquats.fr (mailto:thomas@kumquats.fr) ~ Fondateur &
Directeur technique de

kumquats
Développement web&mobile

- JS / HTML 5 / CSS 3 (React, Sass/Compass)
- PHP (Symfony3, WordPress)
- Mobile (React Native, Cordova/PhoneGap)

nquats.fr (http://kumquats.fr) ~ @kumquatsfr (http://twitter.com/ku

VOUS

Tour de table :

- Vos expériences en développement web
- Vos langages de programmation
- Vos environnements de développement
- Vos attentes

AVERTISSEMENT

J'aime les gifs


GAME OF THRONES


THE
WALKING DEAD

JS

1. **Rappels**
    2. Premiers pas en JS
    3. ES6 : le JavaScript nouveau
    4. API DOM
    5. Ajax
    6. jQuery
    7. La POO en JS
    8. Pour aller plus loin...

RAPPELS

- HTML
- CSS
- client / serveur
- requêtes HTTP
- Single Page App

HTML

HTML

- Utilisé pour décrire le contenu de l’interface utilisateur de la web app.
- Le code indique aussi les éléments de mise en forme (CSS, médias) ou les scripts (JS) à charger.

HTML

Le code HTML est composé d'un arbre de balises (XML) :

- converti en objets manipulables via l'API DOM
- affiché par le moteur de rendu du browser (webkit, gecko, blink, etc.) en attribuant un comportement différent à chaque balise selon son type

ex :

```
<img src="westeros-swag.gif">
```

produira une image à l'écran

HTML

Structure d’un document HTML type

```
<!doctype html>
<html>
<head>
  <title>L'histoire de Westeros</title>
  <meta charset="utf-8">
</head>
<body>
    L'histoire de Westeros en 587 chapitres
</body>
</html>
```

HTML

Balises :

- 1 balise ouvrante et 1 balise fermante
- Toujours (ou presque)

```
<p>Rick Grimes : We are the walking dead</p>
```

HTML

Attributs :

- Les balises ouvrantes peuvent contenir des attributs
- Dans certains cas les attributs sont obligatoires

```
<p class="citation">Rick Grimes : We are the walking dead</p>
```

HTML

Imbrication des balises :

- Permet d’indiquer une hiérarchie ou de regrouper les balises
- Génère une arborescence de nœuds (DOM)

```
<p>
    <span>Rick Grimes :</span>
    <strong>We are the walking dead</strong>
</p>
```

HTML

Les commentaires

- Annotations non affichées
- Commence par <!--
- Termine par -->
- Cas particulier des commentaires conditionnels (Internet Explorer uniquement)

```
<!-- probablement que personne ne verra ce texte, ah ah... -->
```

HTML

```
<!doctype html>
<html>
<head>
  <title>L'histoire de Westeros</title>
  <meta charset="utf-8">
  <meta name="description" content="L'histoire de Westeros en 587
chapitres">
  <meta name="author" content="TF">
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <h1>L'histoire de Westeros en 587 chapitres</h1>
  <p>The recorded history of Westeros extends back over
  12,500 years, according to tradition.</p>
  <p>The people that inhabit the known world do not possess
  objective knowledge about how...</p>
</body>
</html>
```

Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Validate by URI | Validate by File Upload | Validate by Direct Input

Validate by URI

Validate a document online:

Address:

▸ More Options

Check

This validator checks the markup validity of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as RSS/Atom feeds or CSS stylesheets, MobileOK content, or to find broken links, there are other validators and tools available. As an alternative you can also try our non-DTD-based validator.

I ♥ VALIDATOR

The W3C validators rely on community support for hosting and development.
Donate and help us build better tools for a better web.

Flattr


CSS

CSS

- Décrit la présentation des documents HTML
- Complète ou remplace les styles par défaut du navigateur

CSS : SYNTAXE

Styles composés de:

- Sélecteur (balise, id ou classe)
- Propriétés CSS
- Valeur

```
selecteur {
    propriete: valeur;
}
```

Notes :

Les styles CSS sont composés de sélecteurs (type de balise, ID html, ou classe CSS) auxquels sont associés un ensemble de paires "propriété : valeur"

- possibilité de combiner les sélecteur pour constituer des sélecteurs contextuels
  objectif : cibler précisément un élément du document
- possibilité de mutualiser les styles entre plusieurs documents via des feuilles externes

CSS : EXEMPLE

```
div.walker {
    color: gray;
    border: dashed 1px #cccccc;
    background-color: red;
    width: 100px;
    margin: 10px 15px;
}
```

CSS : SÉLECTEURS DE BASE

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

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

CSS : SÉLECTEURS ENFANTS

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

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

CSS : SÉLECTEURS MULTIPLES

Appliquer un style identique à plusieurs sélecteurs :

```
p, .dwarf, #king {
    ...
}
```

CSS : INTÉGRATION

Inline:

```
<p style="color: red">Mon texte...</p>`
```

Dans une balise style:

```
<style>
 p { color: red; }
</style>
```

Dans un fichier externe:

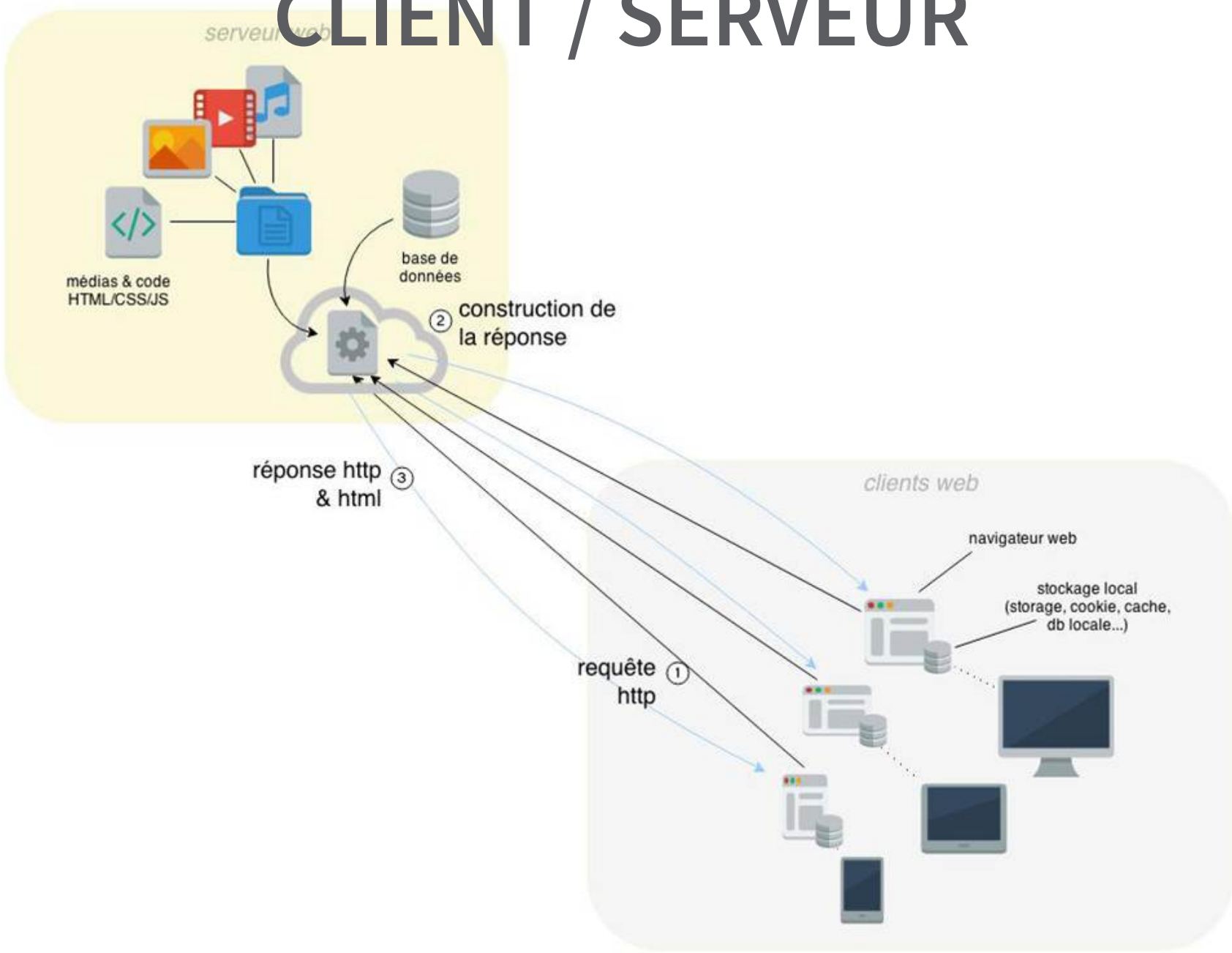
```
<link rel="stylesheet" href="css/styles.css">
```

CLIENT / SERVEUR ?

> *L'environnement client-serveur désigne un mode de communication à travers un réseau entre plusieurs programmes ou logiciels :*
> *l'un, qualifié de client, envoie des requêtes ;*
> *l'autre ou les autres, qualifiés de serveurs, attendent les requêtes des clients et y répondent.*

Wikipedia

CLIENT / SERVEUR



REQUÊTES HTTP

les protocoles de base du Web :

5. Application : HTTP, FTP, DNS
4. Transport : TCP
3. Réseau : IP
2. Liaison : Ethernet/TokenRing, etc.
1. Physique : Boucle locale

Notes :

En TCP/IP les couches traditionnelles du modèle OSi sont légèrement adaptées : les couches 5 Session et 6 Présentation ne sont pas unifiées et sont généralement déléguées à l'application.

REQUÊTES HTTP

- Hypertext Transfer Protocol
- Protocole pour l’accès aux sites web
- Des requêtes / réponses
- Des méthodes : GET, POST, PUT, DELETE, HEAD, …
- Des entêtes : Content-Type, Last-Modified
- Un corps de réponse

REQUÊTES HTTP

```
GET / HTTP/1.1
Host: www.amc.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=
0.8
DNT: 1
Accept-Encoding: gzip, deflate, sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
```

Notes :

Sous windows,on peut utiliser le programme telnet pour lancer des requêtes http : dans un terminal / command prompt, copier coller :

```
telnet www.amc.com 80
GET / HTTP/1.1
Host: www.amc.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/;q=0.8
DNT: 1
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
```

Noter le double retour à la ligne à la fin qui indique la fin de requête. Noter également que dans cet exemple, et contrairement à la slide, on n'a pas la ligne Accept-encoding

RÉPONSE HTTP

```
HTTP/1.1 200 OK
Cache-Control: max-age=1800
Content-Type: text/html; charset=UTF-8
Server: Apache
Vary: Accept-Encoding
X-AMCN-TS: D=4251 t=1488464317972529
X-Powered-By: PHP/5.3.29
Age: 1744
Date: Thu, 02 Mar 2017 14:47:42 GMT
Last-Modified: Thu, 02 Mar 2017 13:55:51 GMT
Expires: Thu, 02 Mar 2017 14:48:38 GMT
X-IP-Address: 178.79.211.132
Connection: keep-alive
Content-Length: 135038

<!DOCTYPE html>
<html lang="en-US">
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=EDGE" />
        <meta name="viewport" content="width=device-width, initial-
scale=1" />
        <meta name="apple-itunes-app" content="app-id=1025120568">
        <link rel="shortcut icon"
              href="http://www.amc.com/wp-content/themes/amc-
tv/favicon.ico" />
        <title>AMC</title>
        ...
```

Brows

sign in

SINGLE PAGE APP (SPA)

- Une seule page HTML
- Navigation sans rechargement
- Modification du contenu de la page via JS
- Utilisation des fonctions HTML 5 (stockage, History API, etc.)

SINGLE PAGE APP : FONCTIONNEMENT

1. l'utilisateur clique sur un lien de la page
2. le JS envoie une requête AJAX vers le serveur
3. le serveur retourne les données au format brut (JSON/XML/...)
4. le JS parse les données reçues
5. le JS génère le code HTML et l'affiche dans la page (API DOM)

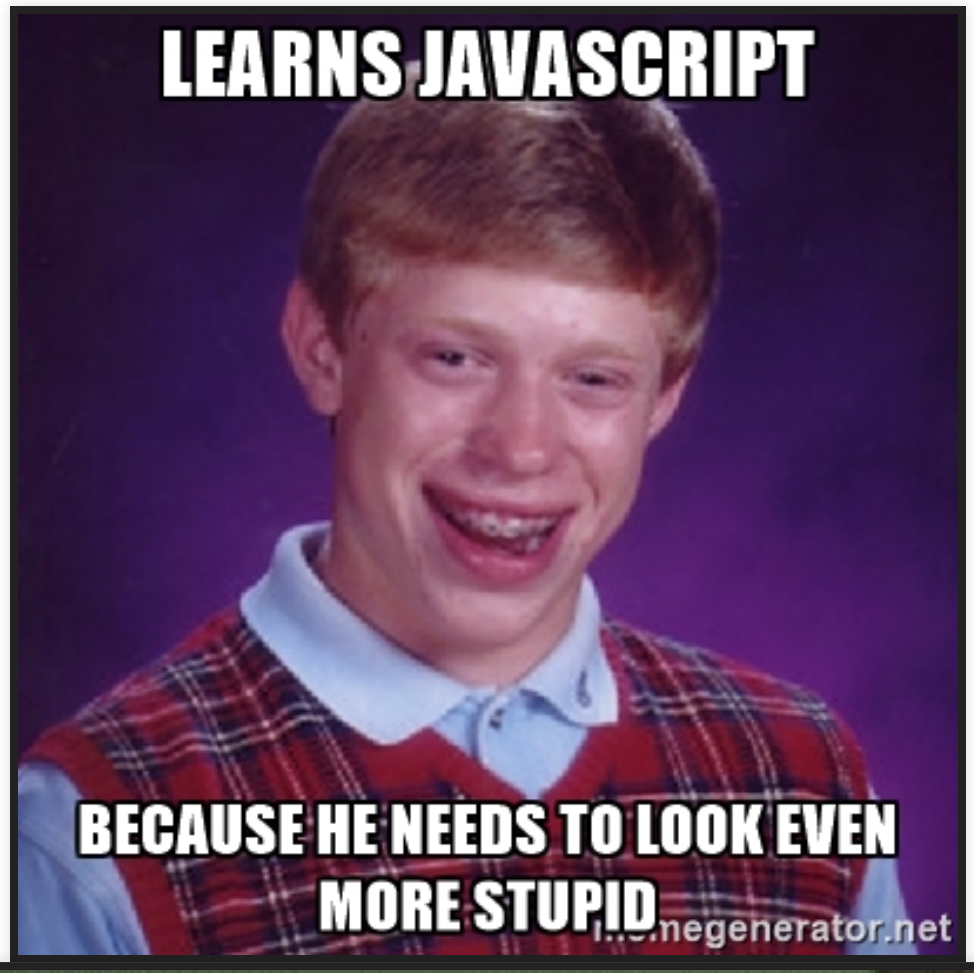
JS

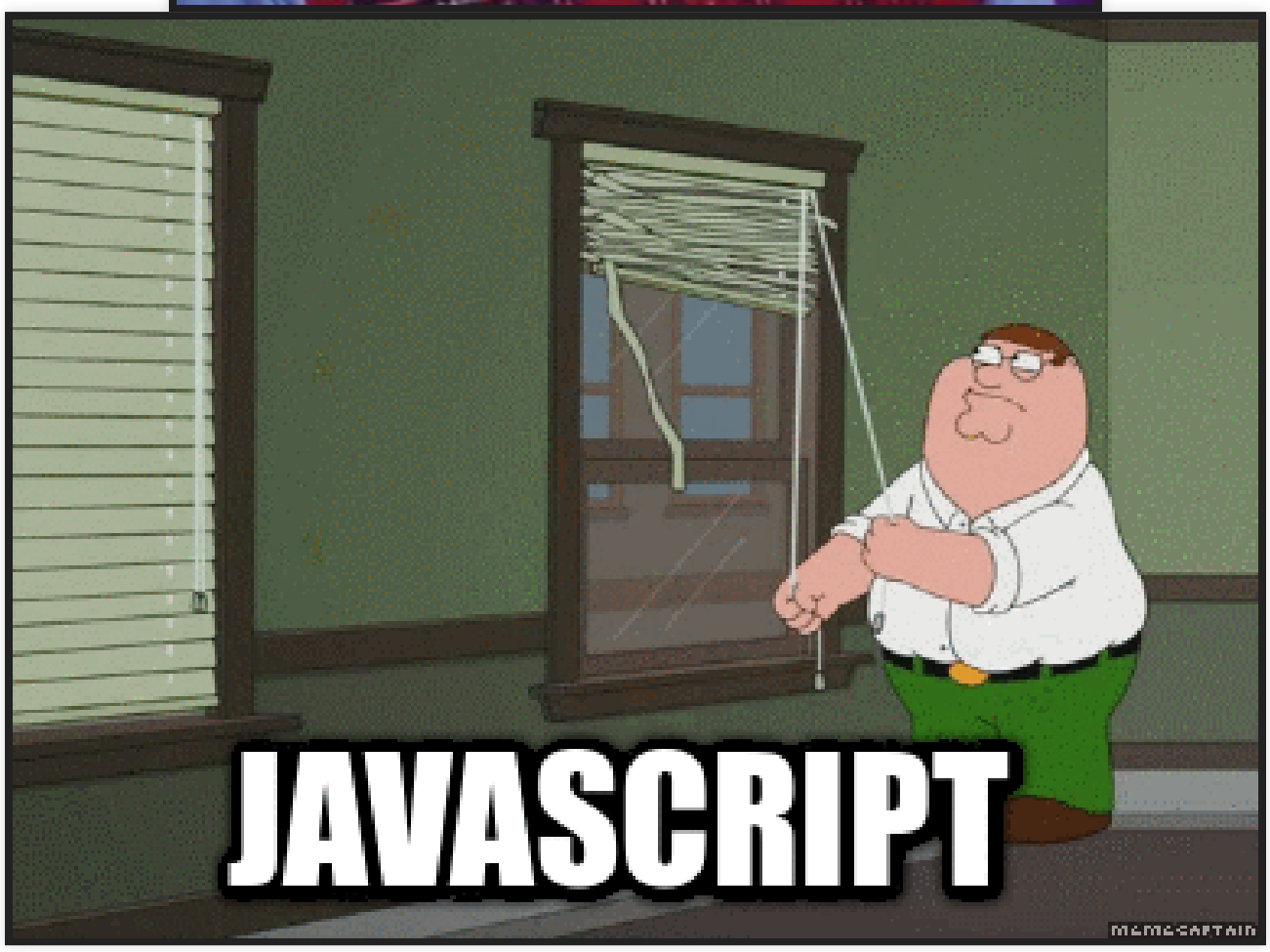
1. Rappels
2. **Premiers pas en JS**
3. ES6 : le JavaScript nouveau
4. API DOM
5. Ajax
6. jQuery
7. La POO en JS
8. Pour aller plus loin...

PREMIERS PAS EN JS

- Mythes & légendes
- Méthodes d'intégration
- Syntaxe de base
- Les variables
- Outils de debug
- Les types de données
- Opérateurs
- Structures de contrôle
- Fonctions
- La chaîne de portée

MYTHES ET LÉGENDES


LEARNS JAVASCRIPT
BECAUSE HE NEEDS TO LOOK EVEN MORE STUPID


JAVASCRIPT

MYTHES ET LÉGENDES

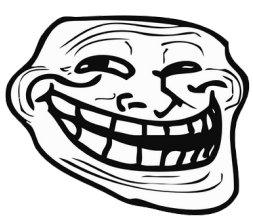
> *“ JavaScript ?? C'est pas un langage ça ! LOL ! ”*
>
> *Un développeur JAVA*

- pas de typage fort
- pas de vraie POO
- pas de block scope
- variables globales par défaut
- valeur de "this" aléatoire
- pas de gestion des dépendances
- ne marche pas dans tous les navigateurs
- une sous-version pourrie de JAVA
- JavaScript sucks !

MYTHES ET LÉGENDES

En réalité :

- beaucoup des reproches sont des a priori (cf. suite de ce cours)
- universel / cross-platform
- disponible sur 99% des postes connectés au web
- innovation permanente
- de plus en plus d'applications en dehors du front (serveur, applis mobiles, applis desktop, spotify, Atom, Visual Studio Code, etc.)
- contrairement à Java, pas besoin de JVM



Notes :

cf. http://stackoverflow.com/questions/9478737/browser-statistics-on-javascript-disabled (http://stackoverflow.com/questions/9478737/browser-statistics-on-javascript-disabled)

RAPPELS JAVASCRIPT : INTÉGRATION

Inline

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

Dans une balise script

```
<script>alert('Bienvenue à Westeros');</script>
```

Dans un fichier externe

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

SYNTAXE DE BASE

- ";" à la fin de chaque instruction
- blocs délimités par des accolades "{...}"
- lowerCamelCase & sensible à la casse
- commentaires avec // et /* */
- code exécuté par le navigateur de haut en bas du code html

JavaScript reference

This part of the JavaScript section on MDN serves as a repository of facts about the JavaScript language. Read more about this reference.

Global Objects

This chapter documents all the JavaScript standard built-in objects, along with their methods and properties.

Value properties

These global properties return a simple value; they have no properties or methods.

- `Infinity`
- `NaN`
- `undefined`
- `null` literal

Function properties

These global functions—functions which are called globally rather than on an object—directly return their results to the caller.

Notes :

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference)

JavaScript Tutorial

‹ Home

Next ›

JavaScript is the programming language of HTML and the Web.

JavaScript is easy to learn.

This tutorial will teach you JavaScript from basic to advanced.

Examples in Each Chapter

With our "Try it Yourself" editor, you can change all examples and view the results.

Example

Notes :

https://www.w3schools.com/js/ (https://www.w3schools.com/js/)

LES VARIABLES

Déclaration d'une variable avec le mot clé 'var'

```
var handOfTheKing;
```

Déclaration et assignation immédiate

```
var handOfTheKing = 'Ned Stark';
```

Modification

```
handOfTheKing = 'Tyrion Lannister';
```

LES VARIABLES

Déclaration multiple

```
var king = 'Geoffrey', city = 'King\'s Landing';
```

Notes :

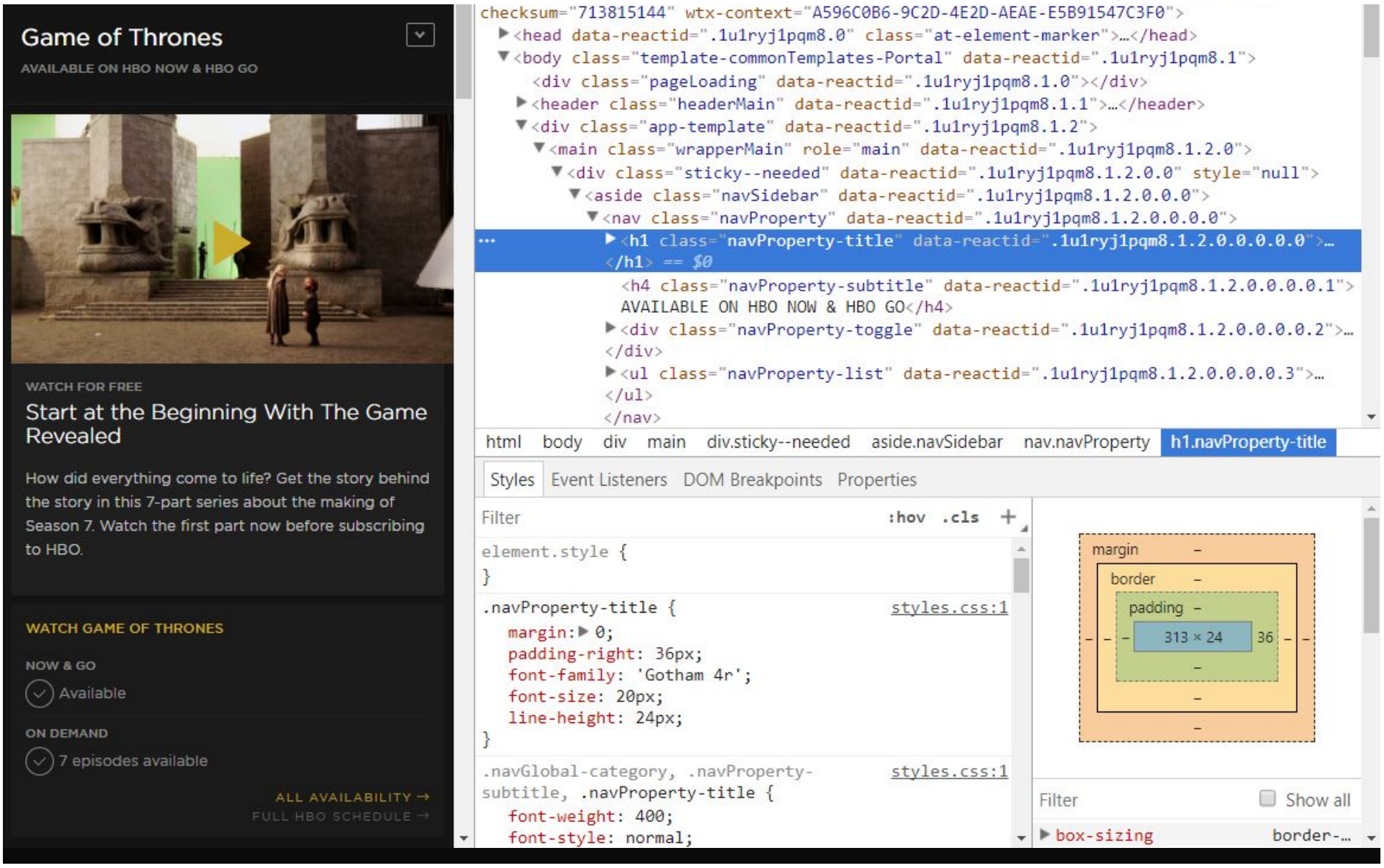
JavaScript ne dispose pas de typage fort, une variable peut contenir n’importe quel type de données. La création de variables occupe de l’espace mémoire et il faut prendre garde à ne pas en déclarer plus que nécessaire. Essayer de réutiliser les variables (pas de "var =" dans les boucles par exemple).

OUTILS DE DEBUG : CHROME DEVTOOLS

- L'inspecteur d'élements
- La console
- L'onglet sources

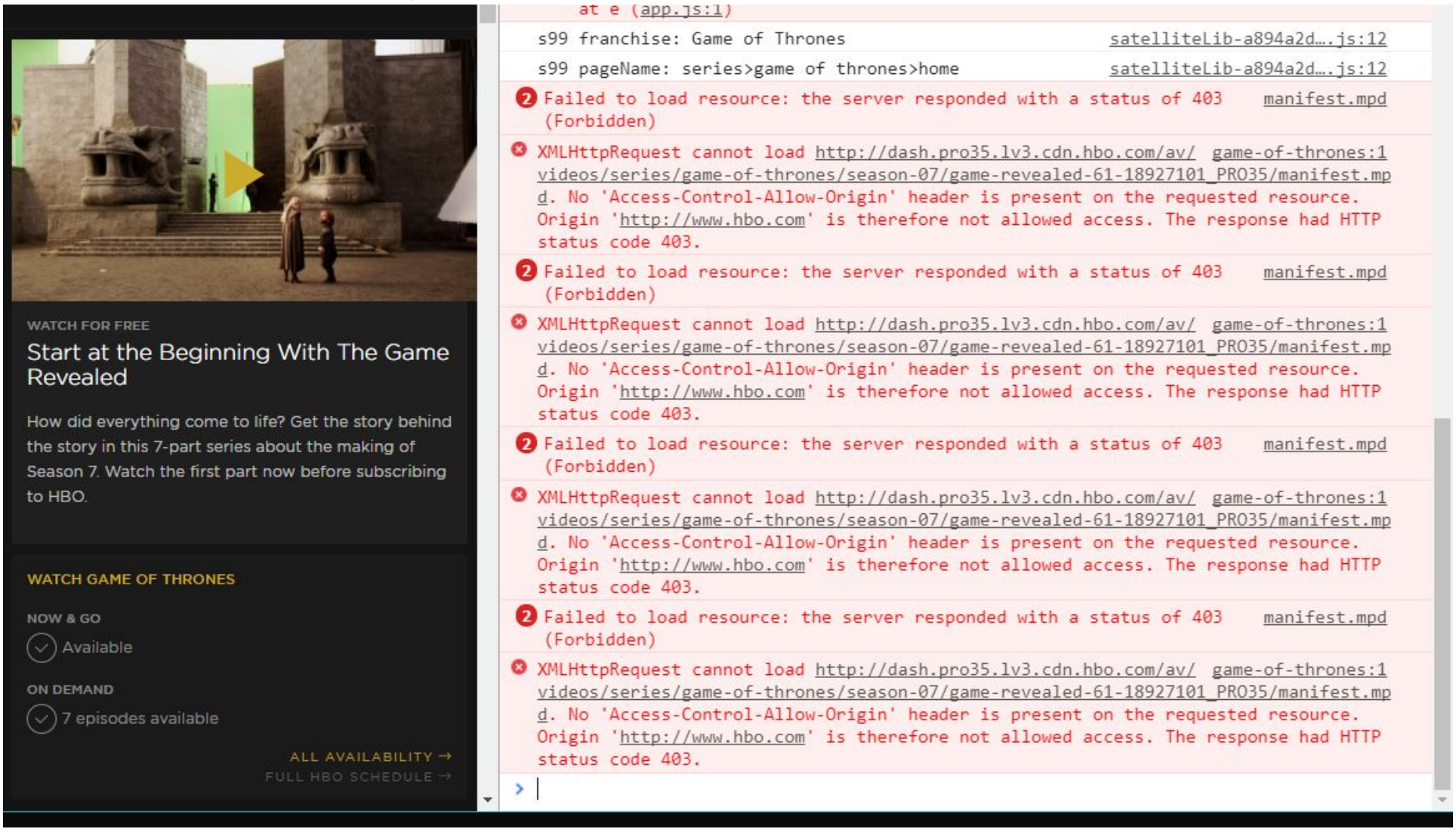
Notes :

L'inspecteur d'élements permet de consulter ET de manipuler le code html et css de la page.



Notes :

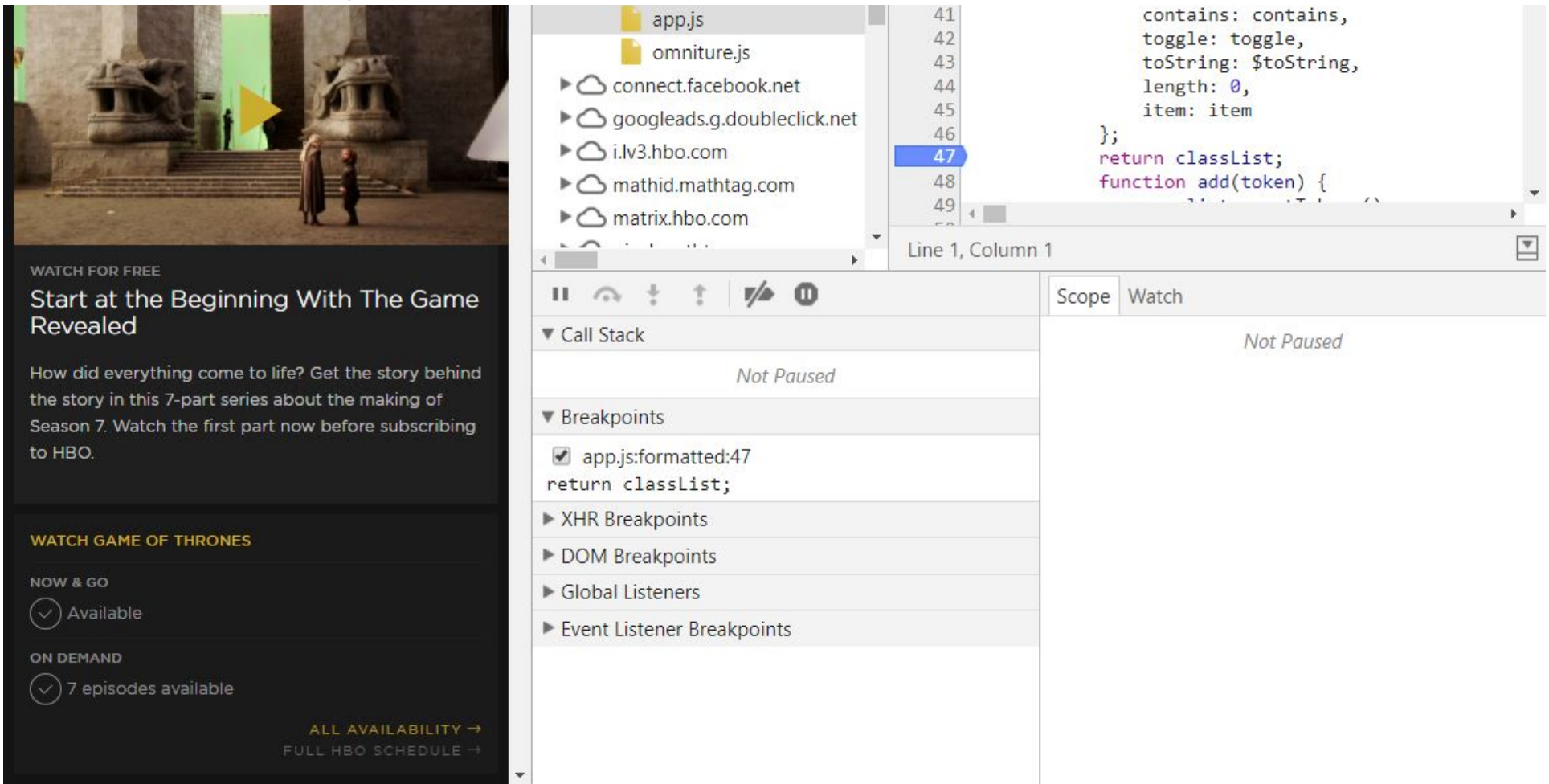
La console permet de voir les messages d'erreur générés par le JavaScript de la page. Certaines erreurs http (404) ou CSS peuvent aussi apparaitre (selon configuration). La console dispose aussi d'un champs de saisie qui permet d'exécuter du code javascript à la demande.



Notes :

L'onglet sources permet d'inspecter le code JavaScript de la page, de placer des breakpoints et de stopper l'exécution du code quand une erreur survient. Quand l'exécution du JS est stoppée, on peut consulter les valeurs des variables locales et globales, de voir la call-stack, etc.

C'est probablement l'onglet des devtools le plus important lorsqu'on développe en JavaScript.



OUTILS DE DEBUG : CONSOLE.LOG

- Permet d'afficher du contenu dans la console.
- Utile pour consulter la valeur d'une variable ou d'une expression
- Peut recevoir plusieurs paramètres affichés les uns après les autres
- Des variantes console.warn, console.error

```
var what = 'door';
console.log('Hold', 'the', what );
```

OUTILS DE DEBUG : DEBUGGER

- Instruction JS qui ajoute un breakpoint automatique
- Stoppe l'exécution du JS si l'onglet sources est ouvert

```
var wylis;
debugger; /* déclenche un breakpoint */
wylis = 'hodor';
```

LES TYPES DE DONNÉES

- Principaux types de données :
  - Number
  - String
  - Boolean
  - Object
  - Array
  - Date
  - Function
- Typage faible (enfin, pour l'instant...) et dynamique

LES TYPES DE DONNÉES : 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 chaines de caractères de plusieurs manières, avec des guillemets simples ou 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 chaines de caractères.

LES TYPES DE DONNÉES : STRING

déclaration multiligne

```
s1 = 'on peut déclarer une chaine sur plusieurs lignes'
    +'en utilisant '
    +'la concaténation.';

s2 = 'ou alors on utilise l\'opérateur "\" \
    avant \
    chaque saut de ligne.';
```

LES TYPES DE DONNÉES : STRING

Principales méthodes

```
var serie = 'The Walking Dead';
console.log(
    serie.length,

    serie.search( 'Walking' ),
    serie.search( 'LOL' ),

    serie.split( ' ' ),

    serie.substring( 4, 8 ),

    serie.toUpperCase()
);
```

```
var serie = 'The Walking Dead';
console.log(
    serie.length, // 16 (longueur de la chaine)

    serie.search( 'Walking' ), // 4 (position de la chaîne recherchée)
    serie.search( 'LOL' ), // -1 (chaîne introuvable)

    serie.split( ' ' ), // ['The', 'Walking', 'Dead' ] (découpe la
chaîne)

    serie.substring( 4, 8 ), // 'Walk' (portion d'une châine)

    serie.toUpperCase() // 'THE WALKING DEAD'
);
```

LES TYPES DE DONNÉES : NUMBER

Déclaration

```
// Nombre entier
var age = 9;

// Nombre à virgule flottante
var price = 12.5;
```

LES TYPES DE DONNÉES : NUMBER

+ : addition de nombres / concaténation de chaines.

Attention à la conversion de type implicite

```
42 + 1337         // 1379
42 + "1337"       // "421337"
42 + true         // 43
```

- * / % : opérations sur les nombres. (re)Attention !

```
42 - "12"         // 30 ...
```

Notes :

```
/
```

LES TYPES DE DONNÉES : NUMBER

Méthodes

```
var price = 12.5;
console.log(
    price.toFixed(2),

    Math.min( 1337, 42 ),

    Math.max( 1337, 42 ),

    Math.pow( 8, 3 ),

    Math.random(),
);
```

```
var price = 12.5;
console.log(
    price.toFixed(2), // "12,50" (conversion en chaîne)

    Math.min( 1337, 42 ), // 42 (minimum de 2 valeurs)

    Math.max( 1337, 42 ), // 1337 (maximum de 2 valeurs)

    Math.pow( 8, 3 ), // 512 (puissance)

    Math.random(), // un nombre aléatoire entre 0 et 1
);
```

LES TYPES DE DONNÉES : BOOLEAN

déclaration

```
var isThisSpoil = false;
var isHodorDead = true;
```

LES TYPES DE DONNÉES : BOOLEAN

Opérateurs logiques

- == === != !== : test de l'égalité (ou non) de valeurs
- < > >= <= : comparaison de valeurs inf. ou sup.
- ?: : Opérateur ternaire a ? b : c;

```
var willDieInNextEpisode = Math.random() > 0.5 ? 'yes' : 'no';
```

Notes :

Les opérateurs ternaires sont utilisés soit en remplacement d'instructions 'if' soit pour faciliter des affectations conditionnelles.

LES TYPES DE DONNÉES : ARRAY

déclaration

```
var emptyArray = [];

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

accès à une valeur

```
var aliveBrother = brothers[ 1 ]; // "Bran"
```

Notes :

La notation var a = new Array(b, c, d); est déconseillée car elle peut conduire à des comportements non souhaités du fait de l'autre constructeur new Array(length);

LES TYPES DE DONNÉES : ARRAY

Méthodes

```
var brothers = [ 'Robb', 'Bran', 'Rickon' ];
console.log(
    brothers.length,

    brothers.join(' and '),

    brothers.push( 'Jon' ),

    brothers
);
```

```
var brothers = [ 'Robb', 'Bran', 'Rickon' ];
console.log(
    brothers.length, // 3

    brothers.join(' and '), // "Robb and Bran and Rickon"

    brothers.push( 'Jon' ), // 4

    brothers // [ 'Robb', 'Bran', 'Rickon', 'Jon' ]
);
```

LES TYPES DE DONNÉES : OBJECT

- utilisé en POO (cf. chapitre associé)
- ou comme dictionnaire (tableau associatif)
- classe parente de toutes les autres

Objet littéral

```
var o = {}
```

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

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

LES TYPES DE DONNÉES : OBJECT

constructeur Object

```
var o = new Object();

o.propriete = valeur;

o["propriete"] = valeur;
```

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

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

arya.friend = mycah;
```

LES TYPES DE DONNÉES : TYPES VIDES

- null
- undefined

```
var hodor;
console.log(hodor)
null == undefined
null === undefined
null == false
null == 0
false == 0
```

```
var hodor;
console.log(hodor)  // undefined
null == undefined   // true
null === undefined  // false
null == false       // false
null == 0           // false
false == 0          // true
```

Notes :

```
/
```

LES TYPES DE DONNÉES : TYPEOF

- récupérer le type avec l'opérateur typeof
- fonctionne avec les types littéraux
- retourne une chaîne de caractères

```
var s = 'je suis une chaîne', s2 = new String('je suis un objet de type
chaîne'),
    n = 1337, n2 = new Number(1337),
    a = [], a2 = new Array(),
    o =  {}, o2 = new Object();

console.log(
    typeof s,
    typeof s2,
    typeof n,
    typeof n2,
    typeof a,
    typeof a2,
    typeof o,
    typeof o2
);
```

```
var s = 'je suis une chaîne', s2 = new String('je suis un objet de type
chaîne'),
    n = 1337, n2 = new Number(1337),
    a = [], a2 = new Array(),
    o =  {}, o2 = new Object();

console.log(
    typeof s,      // "string"
    typeof s2,     // "object"
    typeof n,      // "number"
    typeof n2,     // "object"
    typeof a,      // "object"
    typeof a2,     // "object"
    typeof o,      // "object"
    typeof o2      // "object"
);
```

LES TYPES DE DONNÉES : INSTANCEOF

- tester la classe avec l'opérateur instanceof
- ne fonctionne pas avec les objets primitifs
- utile donc en POO (pour plus tard)

```
var s = 'je suis une chaîne', s2 = new String('je suis un objet de type
chaîne'),
    n = 1337, n2 = new Number(1337),
    a = [], a2 = new Array(),
    o =  {}, o2 = new Object();

console.log(
    s instanceof String,
    s2 instanceof String,
    n instanceof Number,
    n2 instanceof Number,
    a instanceof Array,
    a2 instanceof Array,
    o instanceof Object,
    o2 instanceof Object
);
```

```
var s = 'je suis une chaîne', s2 = new String('je suis un objet de type
chaîne'),
    n = 1337, n2 = new Number(1337),
    a = [], a2 = new Array(),
    o =  {}, o2 = new Object();

console.log(
    s instanceof String,    // false
    s2 instanceof String,   // true
    n instanceof Number,    // false
    n2 instanceof Number,   // true
    a instanceof Array,     // true
    a2 instanceof Array,    // true
    o instanceof Object,    // true
    o2 instanceof Object    // true
);
```

STRUCTURES DE CONTRÔLE

IF

```
if ( condition ) {
    // traitement si la condition est vraie
} else {
    // traitement dans le cas contraire
}
```

STRUCTURES DE CONTRÔLE

SWITCH

```
switch ( variable ) {
    case value1 :
        // ...
        break;
    case value2 :
        // ...
        break;
    default :
        // ...
        break;
}
```

STRUCTURES DE CONTRÔLE

FOR

```
for ( var i = 0 ; i<10 ; i++ ) {
    // ...
}
```

```
var o = { key1: 'value1', key2: 'value2', key3: 'value3' }

for ( var key in o ) {
    console.log( key, o[key] );
}
```

FONCTIONS

Permettent d’exécuter un traitement :

- plusieurs fois (ne pas recopier du code)
- ou lorsqu’un événement se produit (click, etc.)

```
function makeNewEpisode( heroToKill, newCharacters ) {
    // ...
}

var makeNewEpisode = function( heroToKill, newCharacters ) {
    // ...
    return episode;
}

var newEpisode = makeNewEpisode( randomHero, [ witch, dragon ] );
```

LA CHAÎNE DE PORTÉE

Référence à une variable :

1. Recherche de la variable dans la fonction courante
2. Recherche dans la fonction ayant déclaré la fonction courante
3. Recherche dans la portée globale
4. Exception de type ReferenceError

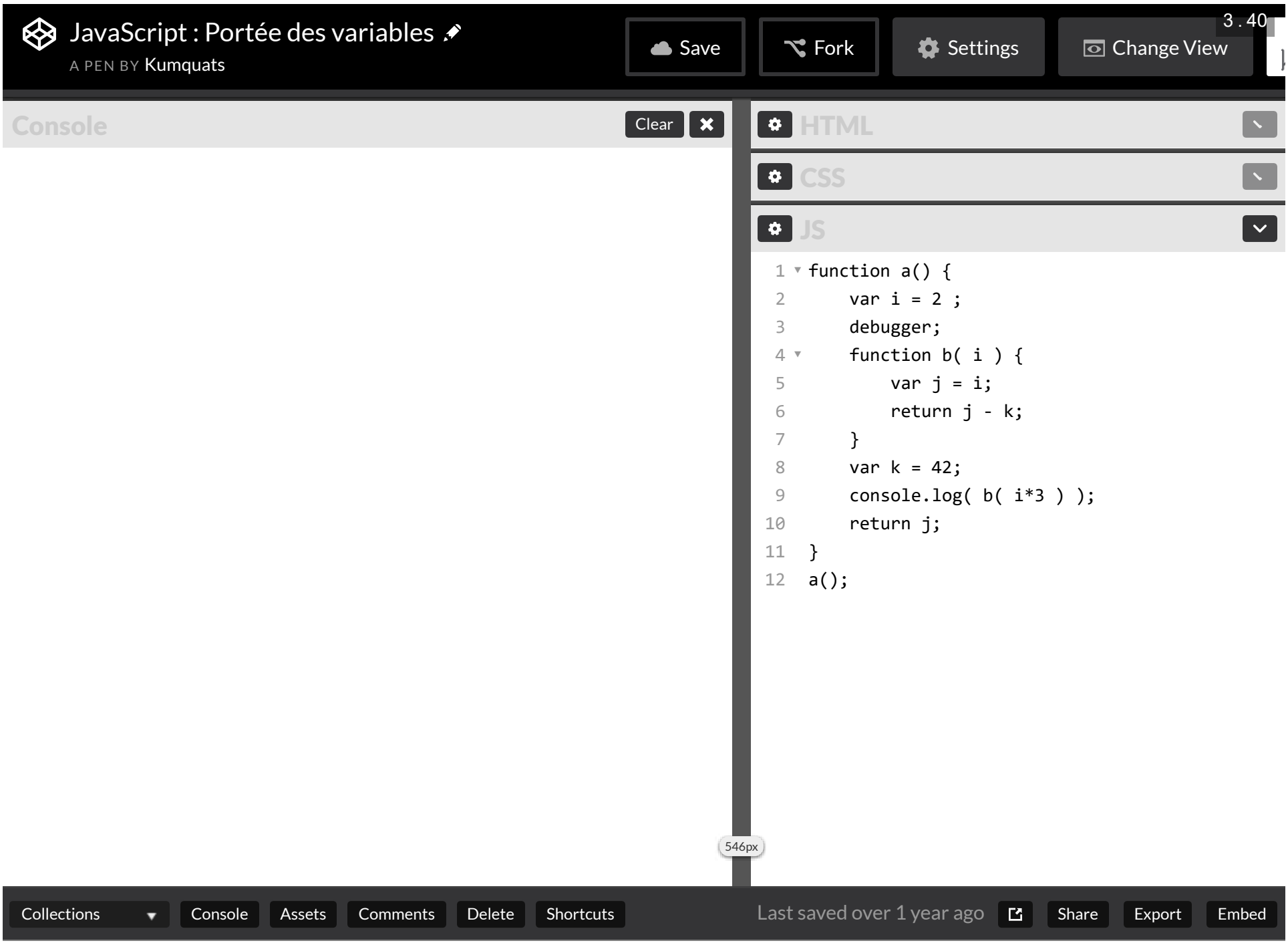
Notes :

La chaîne de portée représente la disponibilité d'une variable dans notre code. En JavaScript les variables ne sont de base accessibles qu'au sein des fonctions dans lesquelles elles sont définies. Il y existe cependant des variables globales qui peuvent être utilisées dans toutes les méthodes.

LA CHAÎNE DE PORTÉE

```
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();
```

Quelle valeur s'affiche dans la console ?



Notes :

http://codepen.io/kumquats/pen/xGpLer?editors=0011 (http://codepen.io/kumquats/pen/xGpLer?editors=0011)

JS

1. Rappels
2. Premiers pas en JS
3. **ES6 : le JavaScript nouveau**
4. API DOM
5. Ajax
6. jQuery
7. La POO en JS
8. Pour aller plus loin...

ES6 : LE JAVASCRIPT NOUVEAU

- ECMAScript
- Nouveautés de la syntaxe ES6
- Compatibilité avec Babel
- Typage avec Flow

ECMASCRIPT (ES)

- Spec gérée par ECMA (ex. European Computer Manufacturers Association)
- Google, Intel, IBM, Microsoft, etc.
- Spécification de langage de script : ECMA-262
- Suivie par JavaScript mais aussi ActionScript (Flash/Flex) et JScript (IE3-9)

Notes :

La liste des membres de ECMA peut se trouver sur le site de l'organisation : http://www.ecma-international.org/memento/members.htm (http://www.ecma-international.org/memento/members.htm)

List of Ecma standards

4.4

From Wikipedia, the free encyclopedia

This is a list of standards published by Ecma International, formerly the European Computer Manufacturers Association.

Contents

- 1 ECMA-1 – ECMA-99
- 2 ECMA-100 – ECMA-199
- 3 ECMA-200 – ECMA-299
- 4 ECMA-300 – ECMA-399
- 5 ECMA-400 – ECMA-499
- 6 See also
- 7 External links

ECMA-1 – ECMA-99

- ECMA-1 – Standard for a 6-bit Input/Output character code (withdrawn)
- ECMA-6 – 7-bit coded character set (same as ISO/IEC 646/ITU-T T.50) (successive editions in 1965, 1967, 1970, 1973, and 1984)
- ECMA-10 – Data Interchange on punched tape (Nov 1965) (withdrawn)
- ECMA-13 – File Structure and Labelling of Magnetic Tapes (later ISO 1001)
- ECMA-17 – Graphic Representation of Control Characters of the ECMA 7-bit Coded Character Set for Information Interchange (Nov 1968)
- ECMA-35 – Character Code Structure and Extension Techniques (ISO/IEC 2022)
- ECMA-43 – 8-bit coded character set (same as ISO/IEC 4873)
- ECMA-48 – ANSI escape codes (same as ISO/IEC 6429)
- ECMA-55 – Minimal BASIC (January 1978)
- ECMA-58 – 8-inch floppy disk
- ECMA-59 – 8-inch floppy disk
- ECMA-66 – 5¼-inch floppy disk
- ECMA-69 – 8-inch floppy disk
- ECMA-70 – 5¼-inch floppy disk

Notes :

https://en.wikipedia.org/wiki/List_of_Ecma_standards (https://en.wikipedia.org/wiki/List_of_Ecma_standards) ECMAScript n'est qu'une des spécifications gérées par ECMA. On note les formats JSON (ECMA-404), Open XML (ECMA-388), le langage C# (ECMA-334), ou encore la spécification des volumes FAT12/16 (ECMA-107) et des CD-ROM ! (ECMA-119)

ECMASCRIPT : UN RYTHME IRRÉGULIER

- 1995 : Netscape LiveScript
- 1996 : renommé en JavaScript et soumis à ECMA
- 1997 : ES1
- 1998 : ES2
- 1999 : ES3
- ...
- 2008 : abandon de ES4
- 2009 : sortie de ES5 (ES3.1)
- ...
- juin 2015 : ES6 / ES2015
- juin 2016 : ES7 / ES2016
- juin 2017 : ES8 / ES2017

Notes :

http://www.benmvp.com/learning-es6-history-of-ecmascript/ (http://www.benmvp.com/learning-es6-history-of-ecmascript/)

ECMASCRIPT : UN NOUVEAU CYCLE

- géré par le TC39 (Technical Committee 39)
- TC39 process : une version tous les ans depuis ES6
- nouvelle numérotation : ES20XX
- éviter les mises à jours trop lourdes (comme ES6)

Notes :

Les compte-rendus des réunions du TC39 sont accessibles en ligne sur github : https://github.com/tc39/tc39-notes (https://github.com/tc39/tc39-notes) On apprend par exemple dans le compte-rendu de la réunion du 25 mai 2017 (https://github.com/tc39/tc39-notes/blob/master/es8/2017-05/may-25.md) que la prochaine version de la norme est publiée en interne à ECMA et qu'elle sera rendue publique au plus tard le 28 juin 2017.

cf. http://2ality.com/2015/11/tc39-process.html (http://2ality.com/2015/11/tc39-process.html)

ECMASCRIPT : UN NOUVEAU CYCLE

plusieurs stades de normalisation :

stage 0 : strawman	Idée non formalisée
stage 1 : proposal	Proposition formelle et argumentée
stage 2 : draft	Première version de la spécification
stage 3 : candidate	En attente de retour d'implémentation
stage 4 : finished	Sera intégrée à la prochaine norme

Notes :

- stage 0 : doit provenir d'un membre du TC39 ou un contributeur enregistré
- stage 1 : proposition doit expliquer le problème qu'elle résoud, proposer des exemples de code et une proposition d'API. Identifie les obstacles éventuels, les polyfills possibles etc.
- stage 2 : à ce stade la spec a de grandes chances de se retrouver dans le standard
- stage 3 : la feature doit avoir été implémentée et testée au minimum 2 fois
- stage 4 :la feature est validée et sera intégrée à la prochaine yearly release

cf. https://tc39.github.io/process-document/ (https://tc39.github.io/process-document/) La liste des features d'ECMAScript en cours de spécification et leur stage peut se consulter sur le repo github du TC39 : https://github.com/tc39/proposals (https://github.com/tc39/proposals)

LET

- remplace "var"
- scopée

```
let i = 0;
for (let i = 1; i < tableau.length; i++) {
    // ...
}
console.log(i); // 0
```

Notes :

Fonctionne exactement comme le var traditionnellement employé. Cependant, le let a la faculté d'être scopé, c'est à dire que son existence se limite au scope dans lequel il se trouve. Cela peut être dans une boucle ou dans une condition.

ES6 : CONST

- déclaration de constante
- scopée

```
function testDeConst()
{
    const hello; // Erreur : une constante doit avoir une valeur !
    const hello = 'Hello';

    if (true) {
        const hello = 'Bonjour';
        hello = 'Hola'; // Erreur : une constante ne peut être modifiée
!
        console.log(hello); // "Bonjour"
    }

    console.log(hello); // "Hello"
}
```

Notes :

const permet de déclarer une constante. De plus, comme le let, une variable const est scopée, c'est à dire que son existence se limite au scope dans lequel elle se trouve. Cela peut être dans une boucle ou dans une condition.

ES6 : LES TEMPLATES STRINGS

- Nouveau délimiteur de chaîne : `
- Injection d'expressions (variables ou appels à des fonctions)
- multiline

```
const greet = `World`;
function maFonction(valeur) {
    return valeur;
}

let chaine = `Hello ${ greet } !`; // Hello World !

chaine = `Hello ${ maFonction('World') } !`; // Hello World !
```

```
function renderLink( target='_self', cssClass, href ) {
    return `<a href="${ href.toLowerCase() }"
                class="${ cssClass }"
                target="${ target }">
                    Cliquez ici !
            </a>`; // multiline !
}
```

ES6 : DESTRUCTURING : OBJETS

- Déclarer des variables qui ont le même nom qu'une propriété d'objet
- Allège le code

```
// En ES6
const personnage = { nom: 'Lannister', prenom: 'Cersei', age: 43 };

const { nom, prenom, age } = personnage;
```

```
// Equivalent en ES5
var personnage = { nom: 'Lannister', prenom: 'Cersei', age: 43 };

var nom = personnage.nom,
    prenom = personnage.prenom,
    age = personnage.age;
```

Notes :

Le destructuring permet de faciliter les affectations de variables issues d'objets ou de tableaux.
http://codepen.io/kumquats/pen/dXWxzW?editors=0011 (http://codepen.io/kumquats/pen/dXWxzW?editors=0011)

ES6 : DESTRUCTURING : TABLEAUX

- Déclarer des variables correspondant à des cellules d'un tableau
- Ordre de déclaration détermine index de la cellule

```
// En ES6
const tableau = [ 'Lannister', 'Jaime' ];

const [ nom, prenom ] = tableau;
```

```
// Equivalent en ES5
var tableau = [ 'Lannister', 'Jaime' ];

var nom = tableau[0],
    prenom = tableau[1];
```

ES6 : DESTRUCTURING : FONCTIONS

```
function renderLink( {target='_self', cssClass, href} ) {
    return '<a href="'+href+'" class="'+cssClass+'" target="'+target
        +'">Cliquez ici !</a>';
}

let link = {
    target: '_blank',
    cssClass: 'characterList',
    href: 'http://gameofthrones.wikia.com/wiki/Category:Characters',
    unused: 12,
}
document.body.innerHTML = renderLink( link );
```

Notes :

https://codepen.io/kumquats/pen/GEqeJL?editors=0010 (https://codepen.io/kumquats/pen/GEqeJL?editors=0010)

ES6 : OPÉRATEUR SPREAD ...

Explose un objet/array dans plusieurs emplacements

```
// appel de fonction avec un tableau
let params = [ 'bonjour', 'tout', 'le', 'monde' ];
maFonction(...params);
```

```
// concaténation de tableaux
let t1 = [ 'e02', 'e03' ];
let t2 = [ 'e01', ...t1, 'e04' , 'e05' ];
//['e01','e02','e03','e04','e05']
```

```
// fusion d'objets
let o = { c: 42, d: 'toto' };
let obj = { a:1, b:'test', ...o }; // { a: 1, b: 'test', c: 42, d:
'toto' }
```

Notes :

https://codepen.io/kumquats/pen/bReyXb?editors=0011 (https://codepen.io/kumquats/pen/bReyXb?editors=0011)

ES6 : OPÉRATEUR REST ...

Récupère les paramètres d'une fonction dans un tableau

```
function maFonction( ...parametres ) {
    console.log( parametres[0] ); // 'bonjour'
    console.log( parametres[1] ); // 'tout'
    console.log( parametres[2] ); // 'le'
    console.log( parametres[3] ); // 'monde'
}
maFonction('bonjour', 'tout', 'le', 'monde');
```

```
function maFonction( mot1, ...parametres ) {
    console.log( mot1 ); // 'bonjour'
    console.log( parametres[0] ); // 'tout'
    console.log( parametres[1] ); // 'le'
    console.log( parametres[2] ); // 'monde'
}
maFonction('bonjour', 'tout', 'le', 'monde');
```

Notes :

rest est une évolution de la variable magique arguments disponible dans toutes les fonctions. Comme l'opérateur spread, il permet de faciliter l'écriture de la syntaxe, notamment dans les opérations d'affectation ou de déclaration de fonction.

ES6 : ARROW FUNCTIONS

- déclaration de fonction simplifiée
- return implicite
- scope préservé

```
// Déclaration de fonctions anonymes
// en ES5
var add = function( a, b ) {
    return a + b;
}

// et en ES6 (opérateur "fat arrow")
let add = ( a, b ) => a + b;
let square = x => x * x;
    // si un seul paramètre, pas besoin de parenthèses
```

Notes :

L'utilisation de l'opérateur Arrow permet de simplifier l'écriture de fonctions simples.

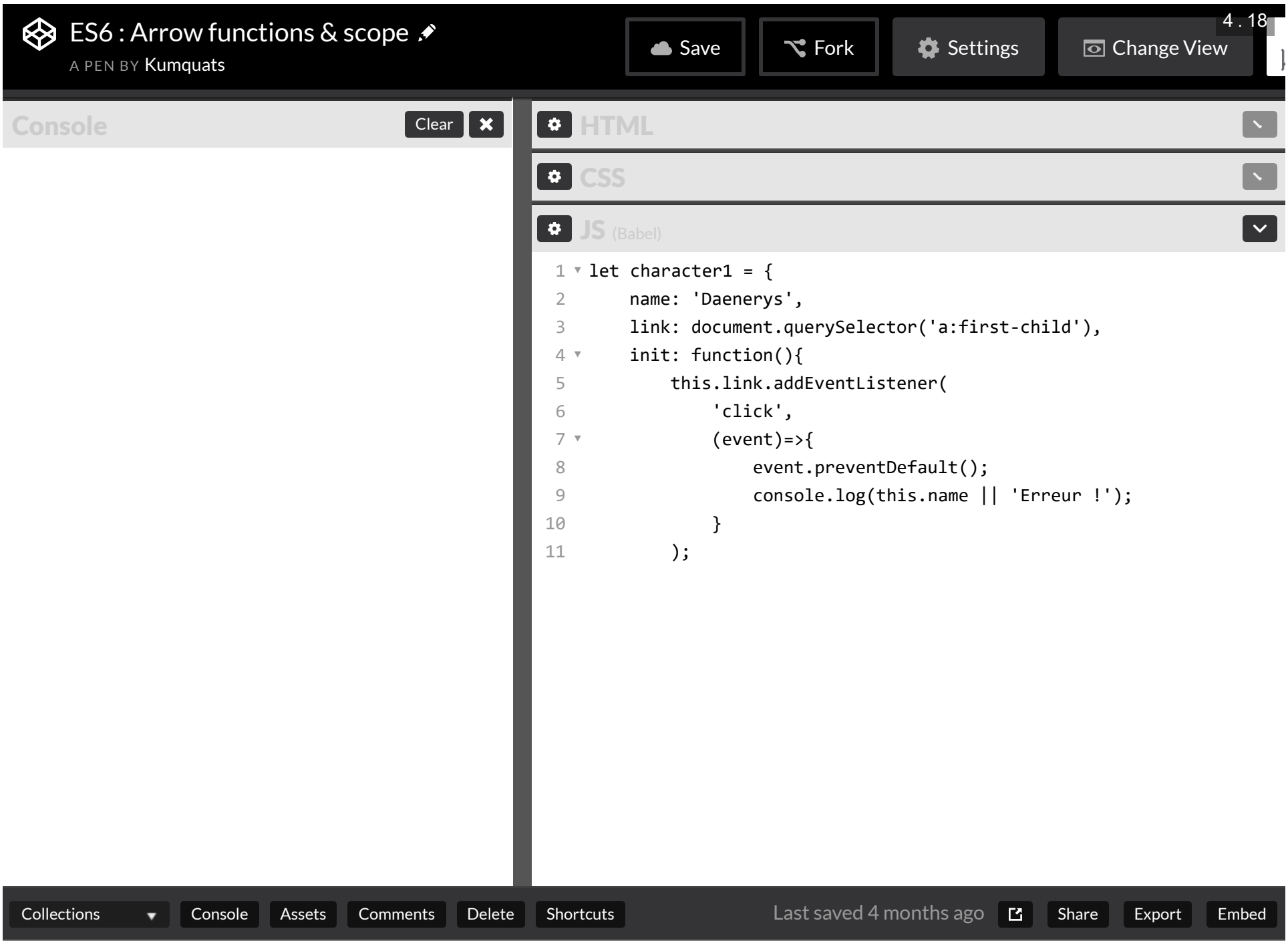
ES6 : ARROW FUNCTIONS : SCOPE

Le **this** dans la arrow function est toujours celui dans lequel elle est déclarée :

```
let character1 = {
    name: 'Daenerys',
    link: document.querySelector('a:first-child'),
    init: function(){
        this.link.addEventListener(
            'click',
            (event)=>{
                event.preventDefault();
                console.log(this.name);
            }
        );
    }
}
character1.init();
```

Notes :

Le scope this, correspond au this parent (pas besoin de se soucier du binding).



Notes :

https://codepen.io/kumquats/pen/bRwgme?editors=0011 (https://codepen.io/kumquats/pen/bRwgme?editors=0011)

PARAMÈTRES PAR DÉFAUT

Valeurs par défaut pour les paramètres des fonctions

```
function killRandomHero( lastname = 'Stark' ){
    let c = getCharacter(lastname);
    c.isDead = true;
}
```

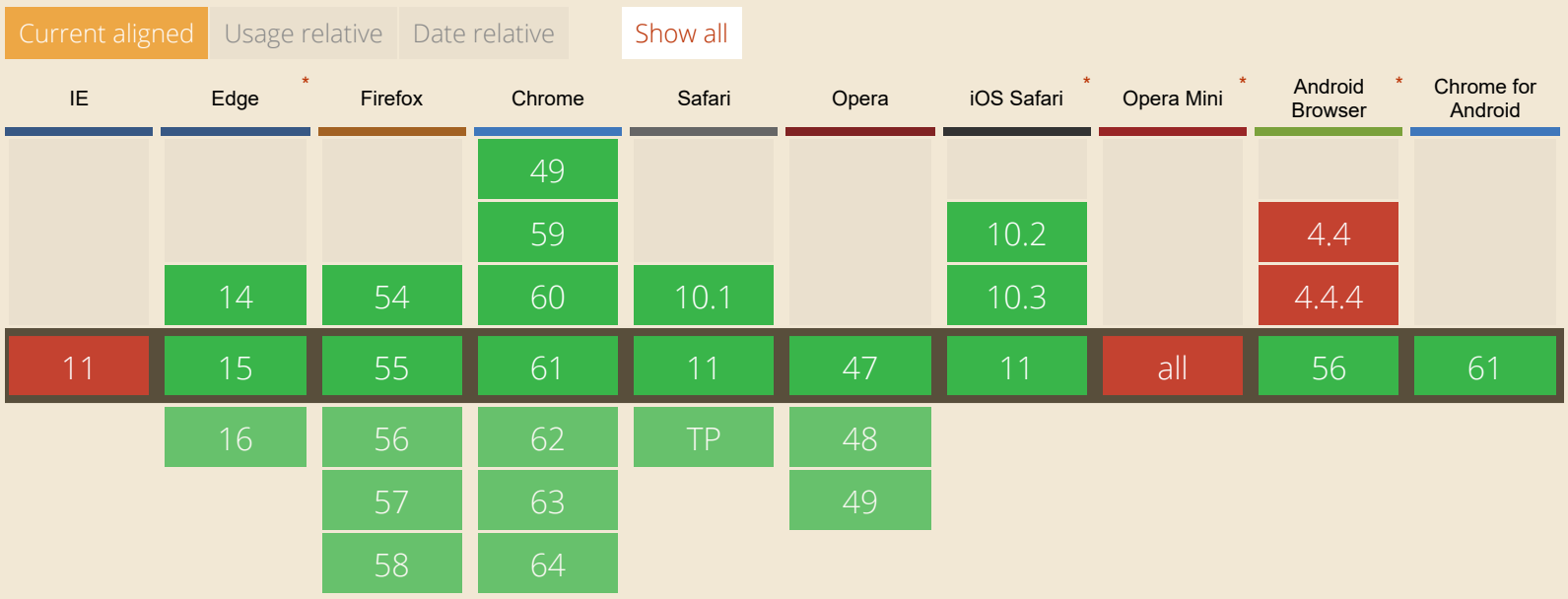
```
// Equivalent en ES5
function killRandomHero( lastname  ){
    if ( lastname === undefined ) {
        lastname = 'Stark';
    }
    // ou
    lastname = lastname || 'Stark';
    let c = getCharacter(lastname);
    c.isDead = true;
}
```

ES6 : ENCORE DU SUCRE SYNTAXIQUE

- les classes
- les modules
- les générateurs
- les fonctions asynchrones
- ...

COMPATIBILITÉ NAVIGATEUR

Problème : les navigateurs implémentent chacun à leur rythme les différentes features



ES6 Number - OTHER

Global 87.28% + 2.23% = 89.51%

Extensions to the `Number` built-in object in ES6, including constant properties `EPSILON`, `MIN_SAFE_INTEGER`, and `MAX_SAFE_INTEGER`, and methods `isFinite` `isInteger` `isSafeInteger` and

---

Notes :

Le support navigateur est très partiel, notamment sur IE. Il faut noter que chaque feature ES6 dispose de son propre support navigateur de manière indépendante. http://caniuse.com/#search=es6 (http://caniuse.com/#search=es6)

COMPAT ES ECMAScript 5 **6** 2016+ next intl non-standard compatibility table

by kangax Gratipay & webbedspace & zloirock

Sort by Engine types ▼ Show obsolete platforms ☐

Show unstable platforms ☑

V8 SpiderMonkey JavaScriptCore

Minor difference (1 point) Small feature (2 p

Large feature (8 points)

Feature name ▸	Current browser 97%	Compilers/polyfills: Traceur 56%	Babel + core-js[2] 71%	Closure 48%	TypeScript + core-js 59%	es6-shim 17%	Kong 4.14[3] 5%	IE 11 11%	Edge 14 93%	Ed 1
Optimisation										
proper tail calls (tail call optimisation) ▸	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0
Syntax										
default function parameters ▸	7/7	4/7	4/7	5/7	5/7	0/7	0/7	0/7	7/7	7
rest parameters ▸	5/5	4/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5
spread (...) operator ▸	15/15	15/15	13/15	12/15	4/15	0/15	0/15	0/15	15/15	15
object literal extensions ▸	6/6	6/6	6/6	4/6	6/6	0/6	0/6	0/6	6/6	6
for..of loops ▸	9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	7/9	9
octal and binary literals ▸	4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4	4/4	4
template literals ▸	5/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5
RegExp "y" and "u" flags ▸	5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5
destructuring, declarations ▸	22/22	20/22	21/22	20/22	15/22	0/22	0/22	0/22	21/22	22
destructuring, assignment ▸	24/24	23/24	24/24	21/24	19/24	0/24	0/24	0/24	23/24	24

Notes :

http://kangax.github.io/compat-table/es6/ (http://kangax.github.io/compat-table/es6/) Tableau détaillé, navigateur par navigateur du support des différentes fonctionnalités ES6. Pour ES2016 voir http://kangax.github.io/compat-table/es2016plus/ (http://kangax.github.io/compat-table/es2016plus/) et pour les deatures encore en cours de normalisation : http://kangax.github.io/compat-table/esnext/ (http://kangax.github.io/compat-table/esnext/)

BABEL

.

https://babeljs.io/ (https://babeljs.io/)

Notes :

L'ES6 n'étant pas encore supporté de manière native par tous les navigateurs actuels, il est nécessaire d'utiliser un préprocesseur. Le principe est simple, on écrit son code en ES6 et le préprocesseur le convertit en JavaScript ES5 qui lui est reconnu par les navigateurs. Il existe plusieurs préprocesseurs mais le plus connu est Babel (https://babeljs.io/ (https://babeljs.io/))

```
// Arrow function
let square = x => x * x;

// const
const i = square(42);
//i = 12; //erreur de compilation (constante)

// let scopé
for ( let i = 0; i < 12 ; i++ ){
  console.log(i);
  if (true){
    let i = 5;
    console.log(i);
  }
}

// destructuring
o = {
  a: "a",
  b: "b",
};
let {a, b} = o;
```

```
"use strict";

var _extends = Object.assign || function (target) { for (

// Arrow function
var square = function square(x) {
  return x * x;
};

// const
var i = square(42);
//i = 12; //erreur de compilation (constante)

// let scopé
for (var _i = 0; _i < 12; _i++) {
  console.log(_i);
  if (true) {
    var _i2 = 5;
    console.log(_i2);
  }
}

```

Notes :

Babel dispose d'un outil en ligne qui permet de tester la conversion de code ES5 en ES6. Pratique pour se rendre compte de l'intérêt de la nouvelle syntaxe notamment pour son côté beaucoup plus concis. babel repl (https://babeljs.io/repl) babel repl pré-rempli avec des exemples de code ES6 (https://babeljs.io/repl/#?babili=false&evaluate=false&lineWrap=false&presets=es2015%2Cstage-2&targets=&browsers=&builtIns=false&debug=false&code_lz=PTAEEECdIewd1AMwK4DsDGAXAljVAoAGwFNNQBnAR2MagmhsTCtTrEAFABYATAEo-w4POYBGDQJDFoxZJFAATBmIC2AB2yEaOPKBXSJNVJmItIRESMnIxJwBL_EQYaxVQULkFAAYBeQAeUGNQdIBqPNAtAGul3STimivQW_mFGF0VmC3ABWpFA9Fk5Cc9BoNi8vnqDAAzEEfGQnHQaA5yAdElVTgByOqEfGgea6VFIZDkXCoA74ACQT

BABEL : INSTALLATION DE NODEJS

- Permet d'exécuter du JavaScript côté serveur
- Dispose du gestionnaire de paquets NPM
  - Permet de gérer les dépendances logicielles d'un projet JS
  - Regroupe l'ensemble des librairies JavaScript
- Nécessaire au fonctionnement de Babel

https://nodejs.org (https://nodejs.org)

Notes :

NodeJS est un outil permettant d'interpréter et d'exécuter du JavaScript côté serveur. Il a connu une ascension fulgurante auprès des développeurs web qui ont vu là la possibilité de créer des applications qui partegeraient du code côté client et côté serveur. Aujourd'hui il est surtout populaire pour son écosystème et son gestionnaire de dépendance NPM permettant de téléchargement simplement la majorité des librairies JavaScript. Beaucoup d'outils sont ajourd'hui développés en JavaScript et Babel ne fait pas exception ici. C'est pour cette raison que NodeJS et ajourd'hui quasiment indispensable pour construire des applications JS modernes. Lien de téléchargement: https://nodejs.org (https://nodejs.org)

BABEL : MISE EN ŒUVRE

Installation

```
npm install --save-dev babel-cli babel-loader babel-core
    babel-preset-es2015
```

Création du fichier .babelrc

```
{
    // Permet d'activer la compilation du code ES6
    "presets": [ "es2015" ]
}
```

Utilisation

```
./node_modules/.bin/babel dossierSource -d dossierDestination
```

Notes :

- babel-cli : package nécessaire pour exploiter la ligne de commande
- babel-loader : loader nécessaire pour que Webpack interprète l'ES6
- babel-core : ensemble des éléments de base nécessaire pour que babel-loader fonctionne

Pour lancer la compilation, il est possible comme ici d'utiliser directement le binaire de babel mais la documentation recommande l'emploi d'un script npm configuré dans le package.json. cf. https://babeljs.io/docs/setup/#babel_cli (https://babeljs.io/docs/setup/#babel_cli)

BABEL : INTÉGRATION

Babel peut s'interfacer avec beaucoup d'outils existants :

Notes :

https://babeljs.io/docs/setup/ (https://babeljs.io/docs/setup/) Il est intéressant de noter que Babel fournit la possibilité d'implémenter sa solution dans beaucoup de technologies différentes.

flow

- Permet le typage en JavaScript
- CLI qui affiche les erreurs de typage
- maintenu par Facebook

```
npm install --save-dev babel-preset-flow flow-bin
./node_modules/.bin/flow init
```

Fichier .babelrc :

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

Notes :

Flow est simplement capable de vérifier le typage, pas de compiler, mais il peut s'intègrer avec Babel.

FLOW : EXEMPLE D'UTILISATION


```
// @flow
// la ligne ci-dessus est nécessaire pour que Flow teste le fichier

// On spécifie ici que la fonction "testFirstChild" retourne un string,
// le point d'intérrogation indiquant que la fonction peut également
// retourner un résultat null
function testFirstChild(firstChild: Node): ?string {
    if(firstChild.youAreTheBest) {
        return firstChild.toString();
    }
}
```

```
./node_modules/.bin/flow
```

Notes :

Le typage des variables fonctionne de la même manière que TypeScript, cependant Flow apporte une chose supplémentaire: le typage du retour d'une fonction. Le point d'interrogation signifie que le retour peut-être null.

NB: Flow peut s'utiliser également avec webpack React : https://flowtype.org/docs/react.html (https://flowtype.org/docs/react.html) Tutoriel d'intégration avec webpack & React : https://blog.iansinnott.com/getting-started-with-flow-and-webpack/ (https://blog.iansinnott.com/getting-started-with-flow-and-webpack/)

JS

1. Rappels
2. Premiers pas en JS
3. ES6 : le JavaScript nouveau
4. **API DOM**
5. Ajax
6. jQuery
7. La POO en JS
8. Pour aller plus loin...

API DOM

- Manipuler la page HTML
- Les événements
- Formulaires

MANIPULER LA PAGE : API DOM

PRINCIPAUX TYPES D'ÉLÉMENTS

- window : fenêtre/onglet du navigateur
- document : contenu de la page
- element : nœud de l’arbre
- nodeList : liste d’objets "element"
- attribute : attribut d’un element

API DOM

SÉLECTIONNER UN NOEUD

- getElementById
- getElementsByTagName
- getElementsByName
- getElementsByClassName

Notes :

Ces quatre méthodes sont les plus couramment employées pour sélectionner un noeud XML existant. Toutes à l'exception de getElementById retournent un objet NodeList car plusieurs éléments peuvent potentiellement correspondre à la recherche. Comme il ne peut y avoir qu'un seul élément avec un id donné, getElementById retourne directement un objet Element.

API DOM

QUERYSELECTOR / QUERYSELECTORALL

- récupère un ou plusieurs éléments à partir d'un sélecteur CSS
- allège le code !

```
document.getElementById('container')
    .getElementsByTagName('ul')[0]
    .getElementsByTagName('li');


document.querySelectorAll('#container ul:first-child li');
```

QUERYSELECTOR : SUPPORT NAVIGATEUR

Notes :

Le support navigateur de querySelector est très bon (IE9+), il n'est plus justifié aujourd'hui de s'en passer.

API DOM

ACCÉDER AUX VALEURS D'UN ÉLÉMENT

- Lire des valeurs :
  - Propriété `innerHTML`
  - Méthode `getAttribute( [attribute] )`
- Modifier des valeurs :
  - Propriété `innerHTML`
  - Méthode `setAttribute( [attribute], [value] )`

Notes :

La propriété `nodeValue` permet d'obtenir la valeur texte d'un élément ou d'un attribut. Pour obtenir la valeur d'un attribut nous pouvons utiliser la méthode `getAttribute( [attribute] )` qui retourne directement la valeur ou la méthode `getAttributeNode( [attribute] )` qui retourne un nœud xml dont nous pouvons lire la valeur via la propriété `nodeValue`.

Pour modifier la valeur d'un élément il suffit simplement d'affecter une nouvelle chaîne de caractères à `nodeValue`. Pour la modification de la valeur d'un attribut cela se fait via la méthode `setAttribute` avec comme premier paramètre le nom de l'attribut et comme second la valeur.

API DOM

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

<script>
    document.getElementById( 'king' ).innerHTML = 'Tommen';
</script>
```


5.9

API DOM

SUPPRIMER / REMPLACER

- Supprimer un élément :
  - `removeChild( [childNode] )`
  - `removeAttribute( [attribute] )`
  - `removeAttributeNode( [node] )`
- Remplacer un élément :
  - `replaceChild( [newChildNode], [oldChildNode] )`
  - `replaceData ( [offset], [length], [string] )`

API DOM

CRÉER UN ÉLÉMENT

- Créer un élément :
  - `createElement( [element] )`
  - `createAttribute( [attribute] )`
  - `createTextNode( [string] )`
  - `createCDATASection ( [string] )`
  - `createComment( [string] )`
- Cloner un élément :
  - `cloneNode( [boolean] )`

Notes :

Pour créer un élément vide il faut utiliser la méthode `createElement` avec comme paramètre le nom de l'élément. La méthode est similaire pour créer un attribut via `createAttribute`.

Les méthodes `createTextNode` et `createCDATASection` permettent d'insérer des valeurs dans les éléments, la seconde méthode insère du texte qui ne sera pas parsé lors du traitement du XML.

La méthode `createComment` permet d'insérer un nœud de commentaire.

Nous pouvons également utiliser la méthode `cloneNode` afin de dupliquer un élément ses attributs et enfants si le booléen en paramètre est à true.

API DOM

INSÉRER UN ÉLÉMENT

- `appendChild( [childNode] )`
- `insertBefore( [newChild], [referenceChild] )`
- `insertData( [offset], [string] )`

Notes :

Lorsque nous avons créé des éléments nous désirons les insérer au XML, cela se fait via les méthodes suivantes :

- `appendChild` qui insère l'objet en paramètre dans un nœud existant ;
- `insertBefore` afin d'ajouter un nouveau nœud avant un autre ;
- `insertData` pour ajouter des données dans un nœud existant.

CRÉATION ET DIFFUSION D'ÉVÉNEMENTS

- Ajout d’écouteurs d’événements :
  - Attributs HTML : `onload`, `onclick`... ;
  - Programmation : `element.addEventListener`.

Notes :

Afin de pouvoir capter les événements déclenchés par le DOM il est nécessaire d’ajouter des écouteurs d’événement, cela peut se faire de deux manières : via les attributs HTML par exemple :

```
<body onload="body_loadedHandler()">
```

body_loadedHandler étant le nom de la méthode à exécuter lorsque l’événement load se produit sur l’élément body ;

En javascript :

```
document.getElementById( ‘toggleBodyColorButton’ ).addEventListener(
‘click’, toggleBodyColorButton_clickHandler );
```

toggleBodyColorButton_clickHandler étant une référence à la méthode appelée lors du clic sur le bouton d’id toggleBodyColorButton.

Les objets de type Event sont alors le premier argument des méthodes appelées écouteur d’événement. Nous pourrons donc utiliser ces objets au sein des écouteurs d’événement.

HTML CSS JS Result

EDIT ON CODEPEN

5.14

```
<a href="#" id="letItBurn">Où sont mes dragons ?</a>

<div class="dragon">Drogon</div>
<div class="dragon">Rhaegal</div>
<div class="dragon">Viserion</div>
```

Où sont mes dragons ?

Notes :

http://codepen.io/kumquats/pen/zBZwJm/ (http://codepen.io/kumquats/pen/zBZwJm/)


5.15
WHERE ARE MY DRAGONS?!

FORMULAIRES (1/2)

```
<form>
    <input type="text" name="message">
    <input type="submit" value="Valider">
</form>
```

écouter la modification d'un champ

```
let input = document.querySelector('form input[name=message]');
input.addEventListener('change', event => console.log('input changed')
);
```

modifier la valeur d'un champ

```
input.value = 'We are the walking dead !'
```

FORMULAIRES (2/2)

écouter la soumission du formulaire

```
let form = document.querySelector('form');
form.addEventListener('submit', event => {
    event.preventDefault();
    ...
});
```

TRAVAUX PRATIQUES