

DÉVELOPPEMENT JAVASCRIPT

1.1

JS

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



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

kumquats.fr (<http://kumquats.fr>) ~ [@kumquatsfr](http://twitter.com/kumquatsfr) (<http://twitter.com/kumquatsfr>)

VOUS

Tour de table :

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

AVERTISSEMENT

1.5

J'aime les gifs



The title card for the TV series 'The Walking Dead'. The background is a dark, grainy, and blurry image of a forest at night, with some light reflecting off the ground and trees. The title 'THE WALKING DEAD' is centered in the upper half of the image. 'THE' is in a smaller, yellow, sans-serif font. 'WALKING DEAD' is in a much larger, bold, yellow, sans-serif font with a distressed, weathered texture. The letters have a slight 3D effect with dark outlines.

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 :

```

```

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](#).



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



Flattr

[Home](#) [About...](#) [News](#) [Docs](#) [Help & FAQ](#) [Feedback](#) [Contribute](#)



This service runs the W3C Markup Validator, [v1.3+hg](#).

COPYRIGHT © 1994-2013 W3C® (MIT, ERCIM, KEIO, BEIHANG), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH



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 : EXAMPLE

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

CSS : SÉLECTEURS DE BASE

2.17

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

```
<p>Characters :</p>  
<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>
```

CSS : SÉLECTEURS MULTIPLES

Appliquer un style identique à plusieurs sélecteurs :

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

Inline:

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

Dans une balise style:

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

Dans un fichier externe:

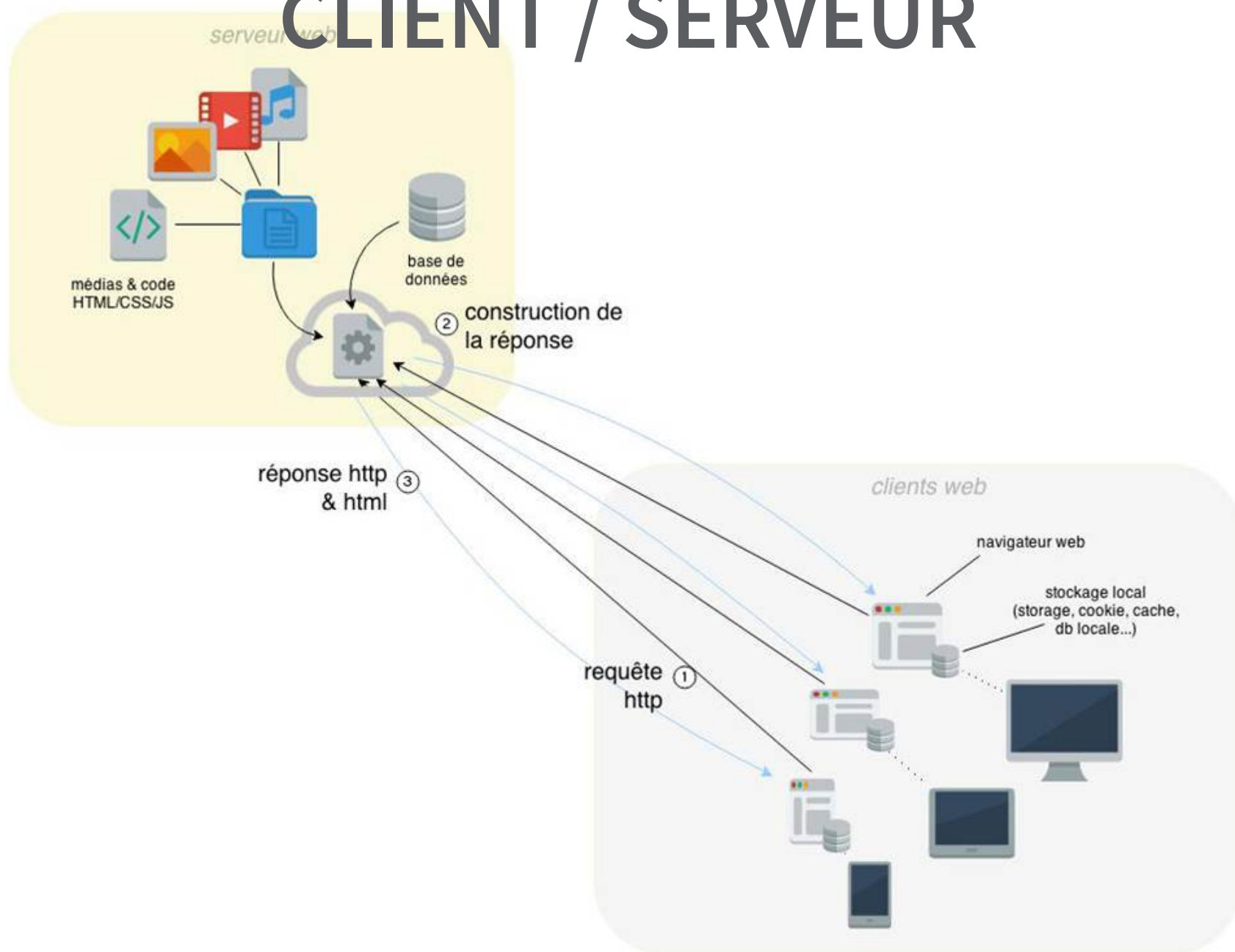
```
<link rel="stylesheet" href="css/styles.css">
```

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>
    ...
```



Browse

AMC

2.27

sign in



FEAR_{THE} WALKING DEAD

NEW: EPISODE 12 NOW AVAILABLE

4 Full Episodes 20+ Extras

AMC SUNDAYS 9/8c

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

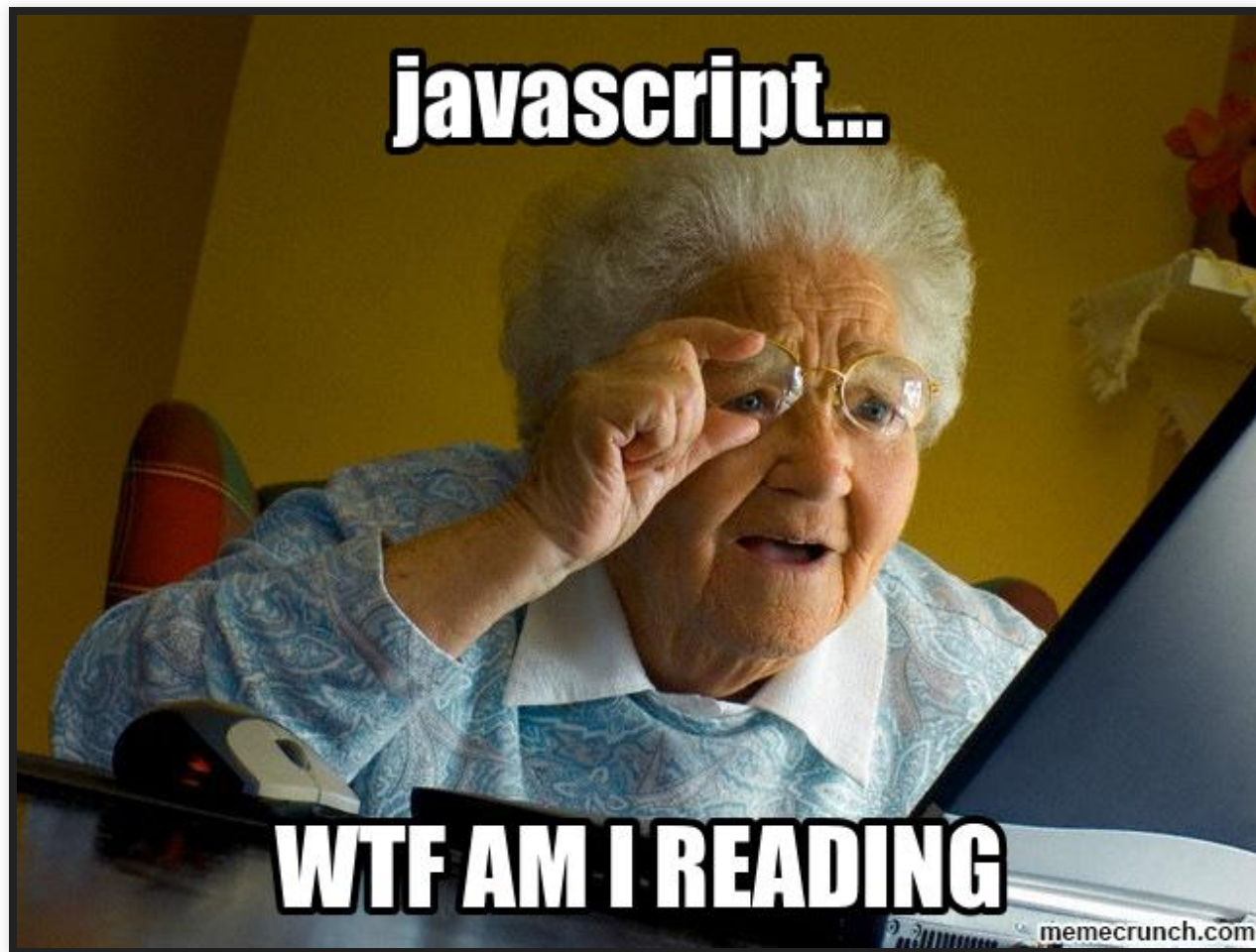
6. jQuery

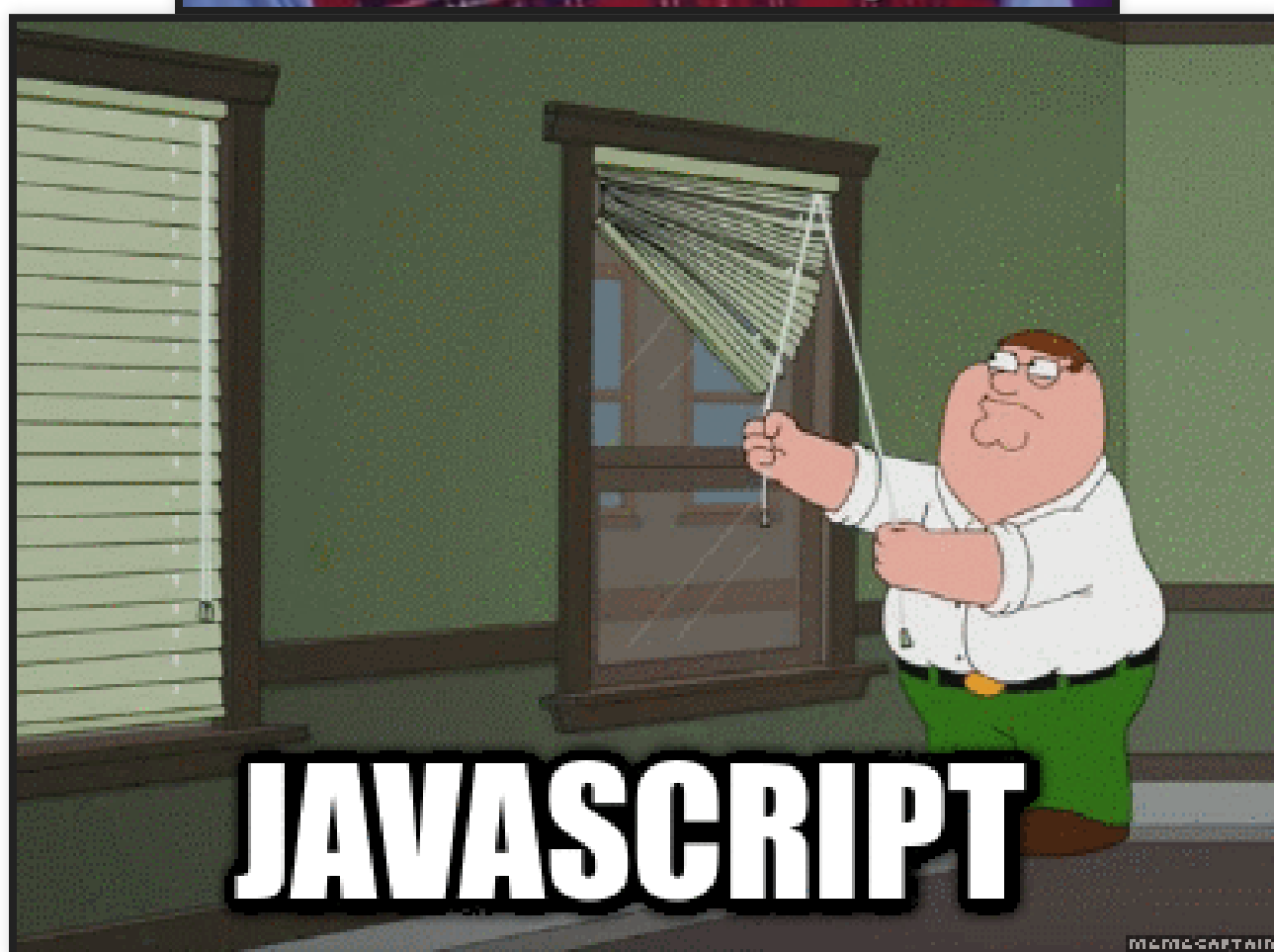
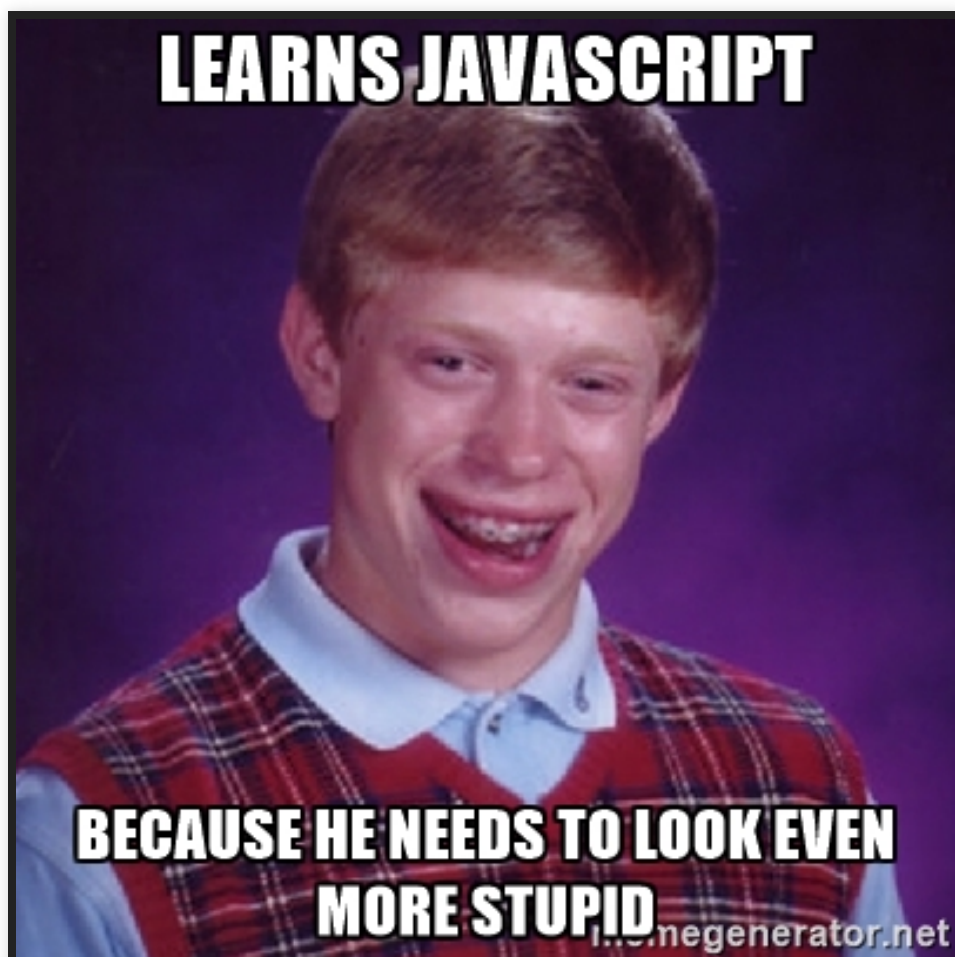
7. La POO en JS

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

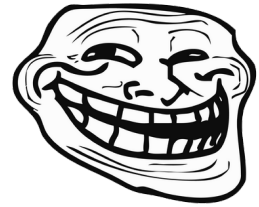
“ *JavaScript ?? C'est pas un langage ça !
LOL !* ”

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 !

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.

- `eval()`

Notes :

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



HTML

CSS

JAVASCRIPT

MORE ▾

REFERENCES ▾

EXAMPLES ▾



Prevent SQL Injection

Learn How to Code Securely & Prevent SQL Injection Bugs in 8 mins. Online Lessons!



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'éléments
- La console
- L'onglet sources

Notes :

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

Game of Thrones

AVAILABLE ON HBO NOW & HBO GO



WATCH FOR FREE

Start at the Beginning With The Game Revealed

How did everything come to life? Get the story behind the story in this 7-part series about the making of Season 7. Watch the first part now before subscribing to HBO.

WATCH GAME OF THRONES

NOW & GO

Available

ON DEMAND

7 episodes available

ALL AVAILABILITY →
FULL HBO SCHEDULE →

```
checksum="713815144" wtx-context="A596C0B6-9C2D-4E2D-AEAE-E5B91547C3F0">
  <head data-reactid=".1u1ryj1pqm8.0" class="at-element-marker">...</head>
  <body class="template-commonTemplates-Portal" data-reactid=".1u1ryj1pqm8.1">
    <div class="pageLoading" data-reactid=".1u1ryj1pqm8.1.0"></div>
    <header class="headerMain" data-reactid=".1u1ryj1pqm8.1.1">...</header>
    <div class="app-template" data-reactid=".1u1ryj1pqm8.1.2">
      <main class="wrapperMain" role="main" data-reactid=".1u1ryj1pqm8.1.2.0">
        <div class="sticky--needed" data-reactid=".1u1ryj1pqm8.1.2.0.0" style="null">
          <aside class="navSidebar" data-reactid=".1u1ryj1pqm8.1.2.0.0.0">
            <nav class="navProperty" data-reactid=".1u1ryj1pqm8.1.2.0.0.0.0">
              <h1 class="navProperty-title" data-reactid=".1u1ryj1pqm8.1.2.0.0.0.0.0">...
            </h1> == $0
              <h4 class="navProperty-subtitle" data-reactid=".1u1ryj1pqm8.1.2.0.0.0.0.1">
                AVAILABLE ON HBO NOW & HBO GO</h4>
              <div class="navProperty-toggle" data-reactid=".1u1ryj1pqm8.1.2.0.0.0.0.2">...
            </div>
              <ul class="navProperty-list" data-reactid=".1u1ryj1pqm8.1.2.0.0.0.0.3">...
            </ul>
          </nav>
        </div>
      </main>
    </div>
  </body>
</html>
```

html body div main div.sticky--needed aside.navSidebar nav.navProperty **h1.navProperty-title**

Styles Event Listeners DOM Breakpoints Properties

Filter

element.style {

}

.navProperty-title {

margin: 0;

padding-right: 36px;

font-family: 'Gotham 4r';

font-size: 20px;

line-height: 24px;

}

.navGlobal-category, .navProperty-subtitle, .navProperty-title {

font-weight: 400;

font-style: normal;

}

styles.css:1

styles.css:1

margin

border

padding

313 × 24

36

Filter

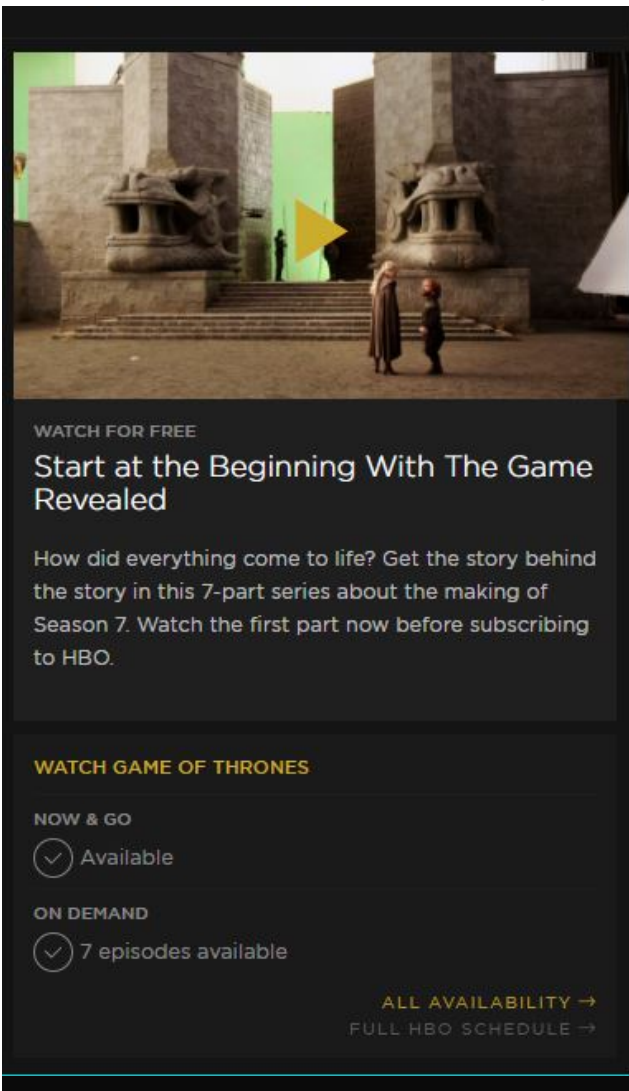
Show all

box-sizing

border-...

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.



WATCH FOR FREE

Start at the Beginning With The Game Revealed

How did everything come to life? Get the story behind the story in this 7-part series about the making of Season 7. Watch the first part now before subscribing to HBO.

WATCH GAME OF THRONES

NOW & GO

Available

ON DEMAND

7 episodes available

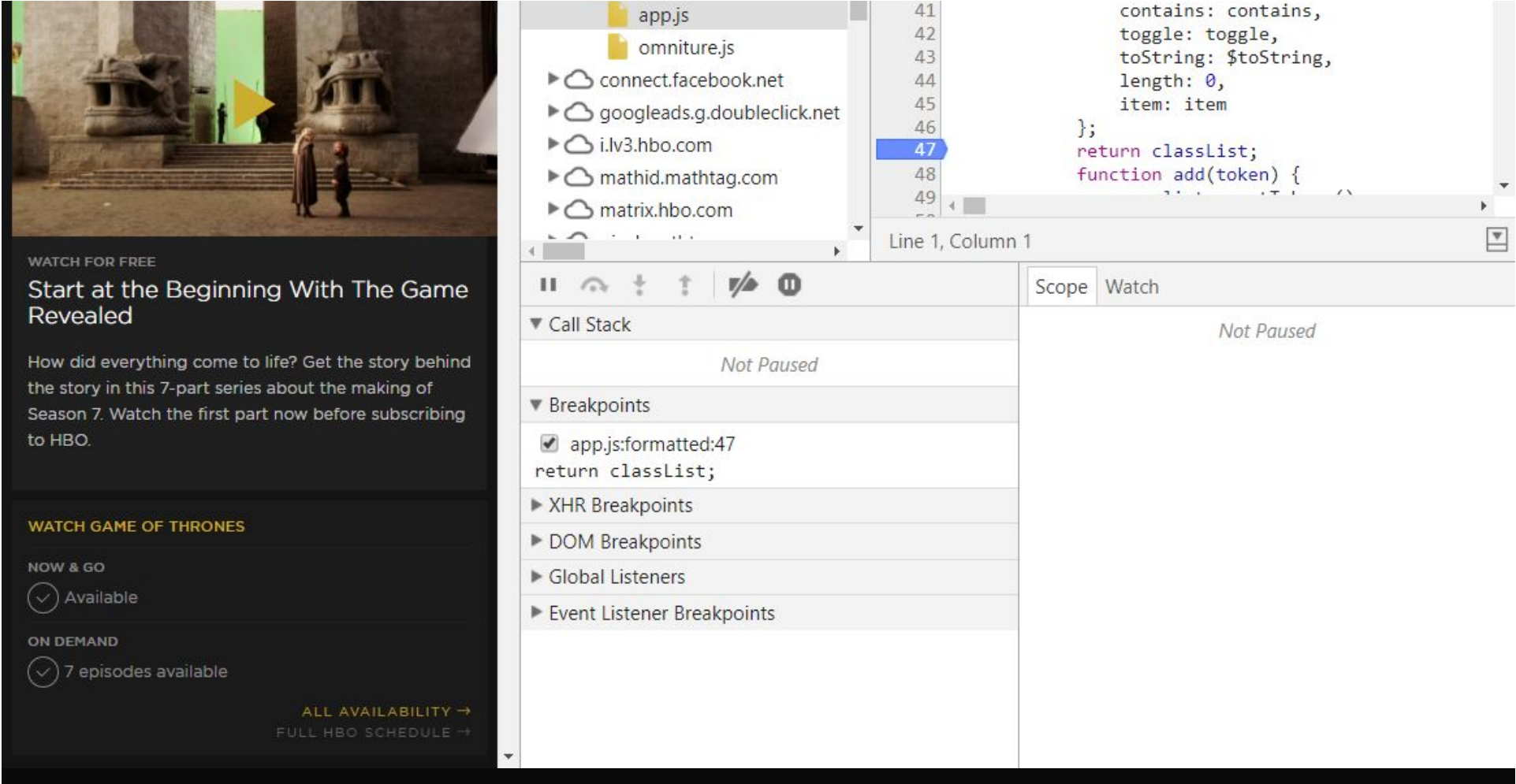
ALL AVAILABILITY →
FULL HBO SCHEDULE →

```
at e (app.js:1)
s99 franchise: Game of Thrones satelliteLib-a894a2d...js:12
s99 pageName: series>game of thrones>home satelliteLib-a894a2d...js:12
2 Failed to load resource: the server responded with a status of 403 manifest.mpd (Forbidden)
XMLHttpRequest cannot load http://dash.pro35.lv3.cdn.hbo.com/av/ game-of-thrones:1 videos/series/game-of-thrones/season-07/game-revealed-61-18927101_PRO35/manifest.mpd. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://www.hbo.com' is therefore not allowed access. The response had HTTP status code 403.
2 Failed to load resource: the server responded with a status of 403 manifest.mpd (Forbidden)
XMLHttpRequest cannot load http://dash.pro35.lv3.cdn.hbo.com/av/ game-of-thrones:1 videos/series/game-of-thrones/season-07/game-revealed-61-18927101_PRO35/manifest.mpd. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://www.hbo.com' is therefore not allowed access. The response had HTTP status code 403.
2 Failed to load resource: the server responded with a status of 403 manifest.mpd (Forbidden)
XMLHttpRequest cannot load http://dash.pro35.lv3.cdn.hbo.com/av/ game-of-thrones:1 videos/series/game-of-thrones/season-07/game-revealed-61-18927101_PRO35/manifest.mpd. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://www.hbo.com' is therefore not allowed access. The response had HTTP status code 403.
2 Failed to load resource: the server responded with a status of 403 manifest.mpd (Forbidden)
XMLHttpRequest cannot load http://dash.pro35.lv3.cdn.hbo.com/av/ game-of-thrones:1 videos/series/game-of-thrones/season-07/game-revealed-61-18927101_PRO35/manifest.mpd. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://www.hbo.com' is therefore not allowed access. The response had HTTP status code 403.
```


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 3.16

- 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

3.17

- 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 3.19

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

LES TYPES DE DONNÉES : STRING

3.20

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 3.21

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 chaîne)

    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 chaîne)

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

LES TYPES DE DONNÉES : NUMBER

3.22

Déclaration

```
// Nombre entier  
var age = 9;  
  
// Nombre à virgule flottante  
var price = 12.5;
```

LES TYPES DE DONNÉES : NUMBER

3.23

+ : addition de nombres / concaténation de chaînes.

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 3.24

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

3.25

déclaration

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

LES TYPES DE DONNÉES : BOOLEAN 3.26

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

3.29

- 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

3.30

constructeur Object

```
var o = new Object();
```

```
o.propriete = valeur;
```

```
o[string] = 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 3.32

- 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
);
```

IF

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

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 ?

 JavaScript : Portée des variables 

Save

Fork

Settings

Change View

3.40

HTML

CSS

 JS

1

2

3

4

5

6

7

8

9

10

11

12

function a() {

var i = 2 ;

debugger;

function b(i) {

var j = i;

return j - k;

}

var k = 42;

console.log(b(i*3));

return j;

}

a();

Console

Clear



Collections

▼

Console

Assets

Comments

Delete

Shortcuts

Last saved about 1 year ago



Share

Export

Embed

Notes :

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