



# **PROGRAMMATION WEB ET MOBILE**

LICENCE SPI 3 PARCOURS INFORMATIQUE

2024

PAUL-ANTOINE BISGAMBIGLIA

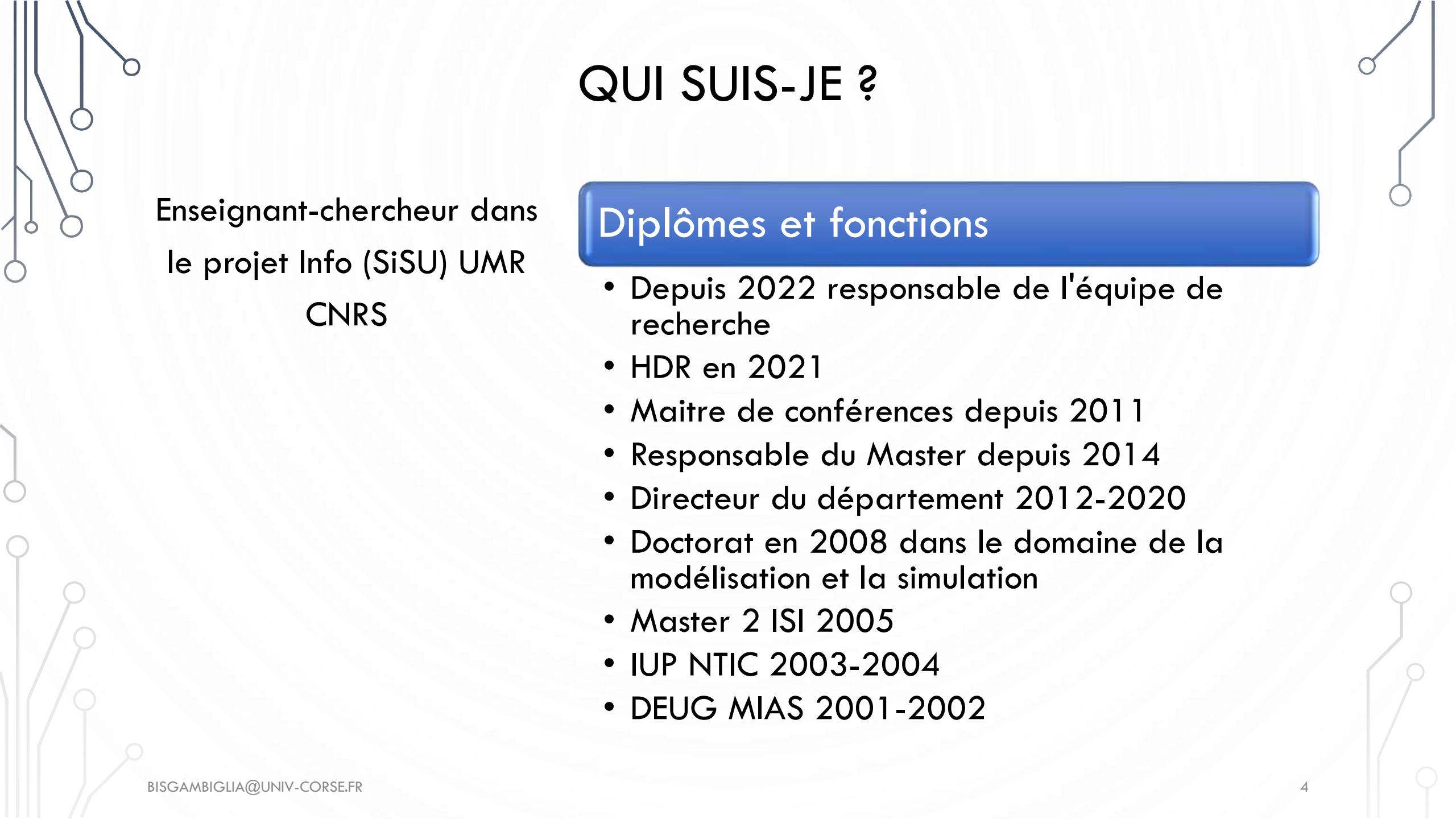
[HTTP://PAUL-ANTOINE-BISGAMBIGLIA.UNIV-CORSE.FR/](http://PAUL-ANTOINE-BISGAMBIGLIA.UNIV-CORSE.FR/)

@PABISGAMBIGLIA

# LICENCE



- Cours en licence libre
- Certaines images ont été récupérées sur google image avec le filtre « réutilisation utilisée sans but commercial »
- Les images sans sources sont sous Licences Creative Commons
- Certains exemples de code sont issus de sites web référencés en fin de chapitre via leurs urls



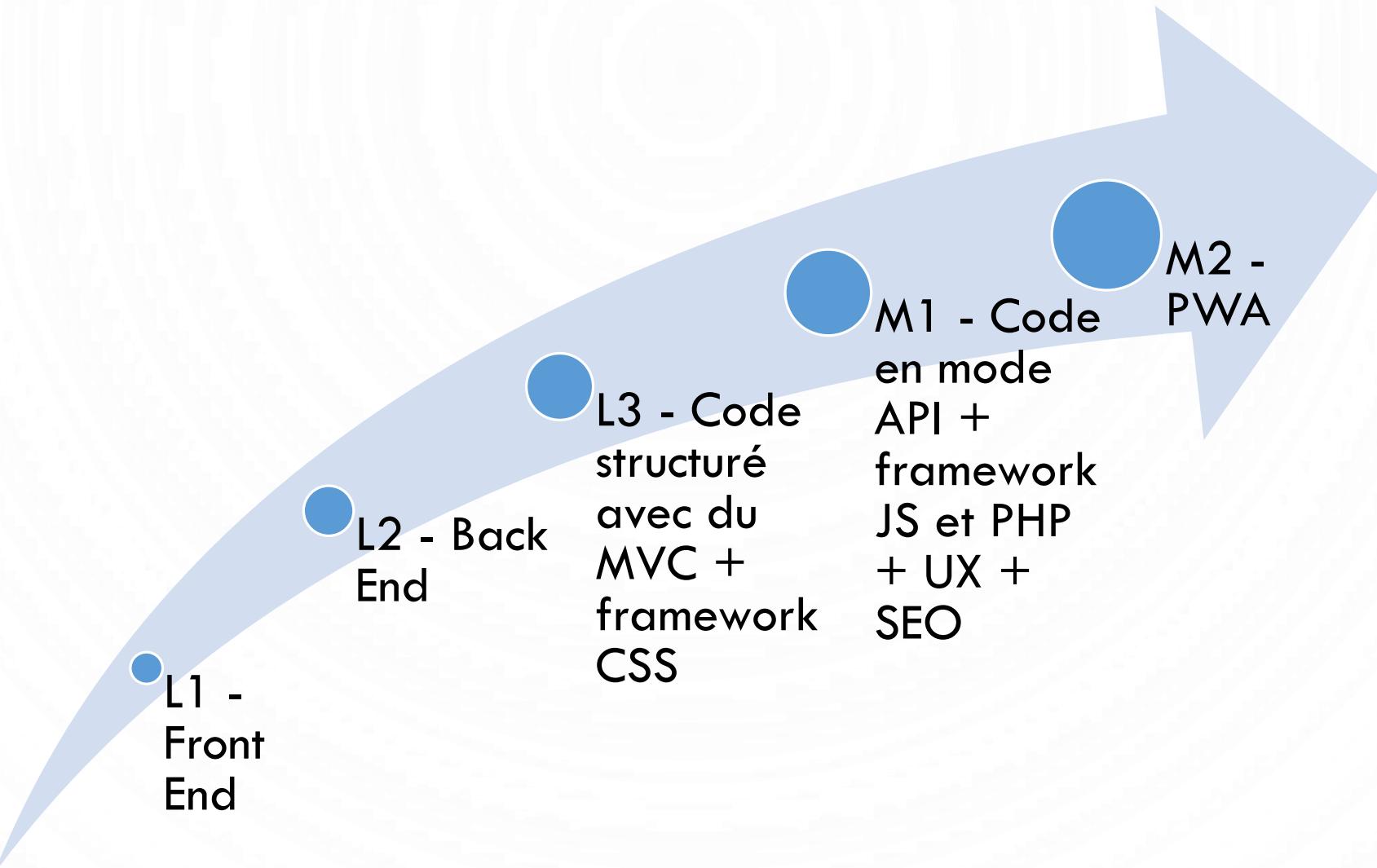
# QUI SUIS-JE ?

Enseignant-chercheur dans  
le projet Info (SiSU) UMR  
CNRS

## Diplômes et fonctions

- Depuis 2022 responsable de l'équipe de recherche
- HDR en 2021
- Maitre de conférences depuis 2011
- Responsable du Master depuis 2014
- Directeur du département 2012-2020
- Doctorat en 2008 dans le domaine de la modélisation et la simulation
- Master 2 ISI 2005
- IUP NTIC 2003-2004
- DEUG MIAS 2001-2002

# LE COURS ANNÉE PAR ANNÉE



# LE COURS

- Programmation logique & **Programmation web**
  - **HTML**
  - **CSS**
  - **JS**
  - **PHP**

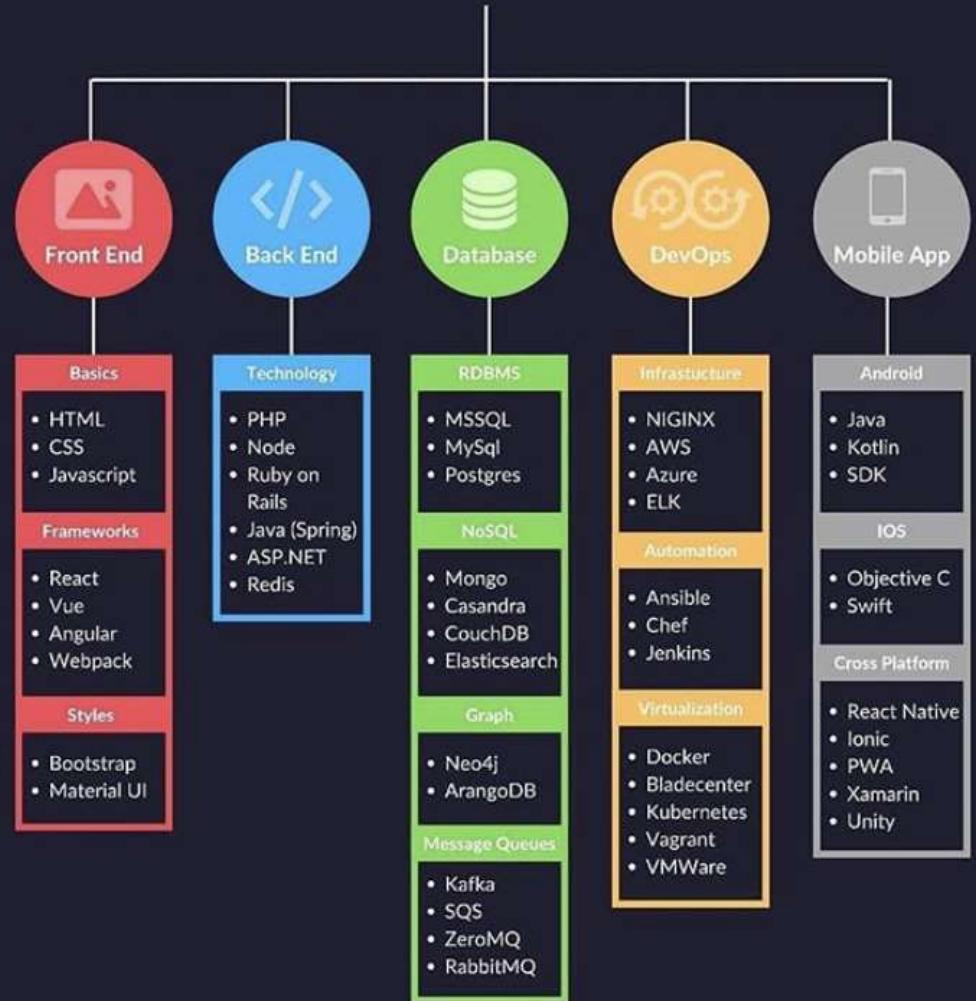
- 1) **Introduction**
  - 1) Le web, et ses techno
- 2) **Les bases**
  - 1) HTML : langage de balise <body> </body>
  - 2) Ergonomie du web
    - 1) CSS 3 pour la mise en forme
    - 2) Framework css (Bootstrap, ect.)
  - 3) JavaScript et TypeScript
- 3) **Dynamisme coté client**
  - 1) jQuery
- 4) **Dynamisme coté serveur**
  - 1) PHP

# EVALUATION



- 1h d'examen en mai en session
  - 1 note de TP / Projet / Contrôle Continu
- 
- Vous aurez à rendre un TP / Projet
  - L'évaluation de ce projet sera faite sous forme de démonstration en live.

# FULL STACK DEVELOPER



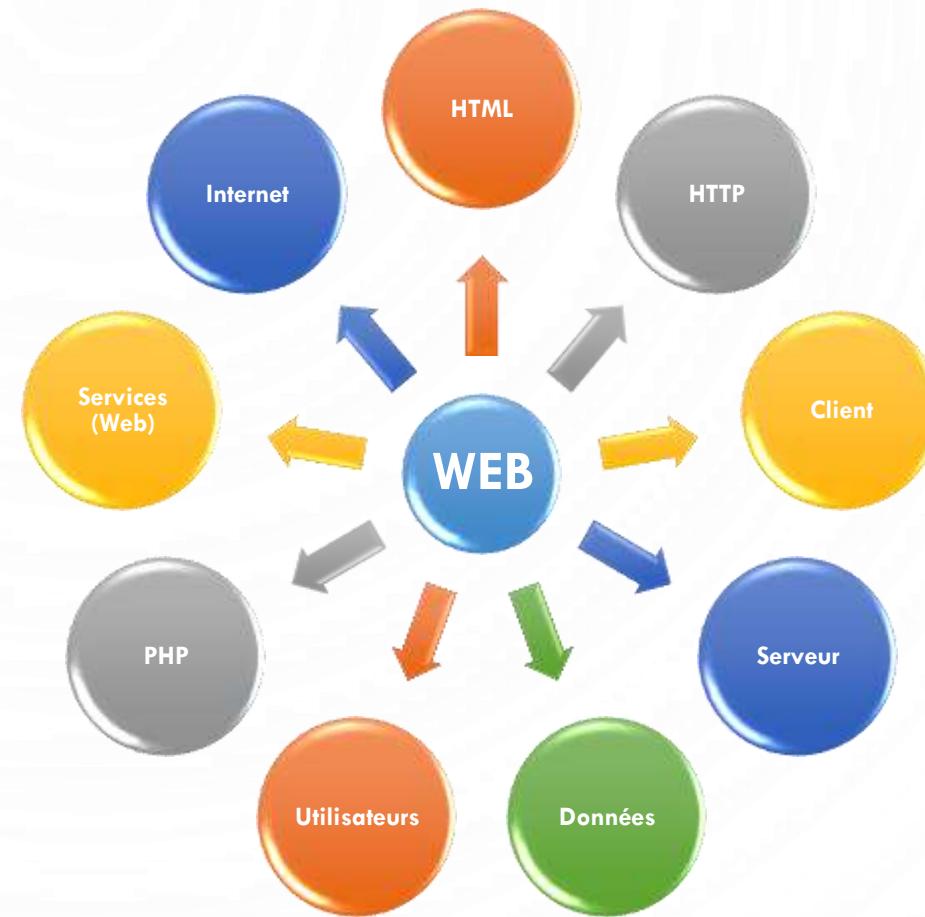
# OBJECTIFS

- Découvrir ou peut-être maîtriser de nouveaux langages de programmation
- Prendre en main les techno Web
- Savoir faire un site



# LE WEB

?



# EVOLUTION

Allez visiter :

- <http://www.evolutionoftheweb.com/?hl=fr>

*On en discute*

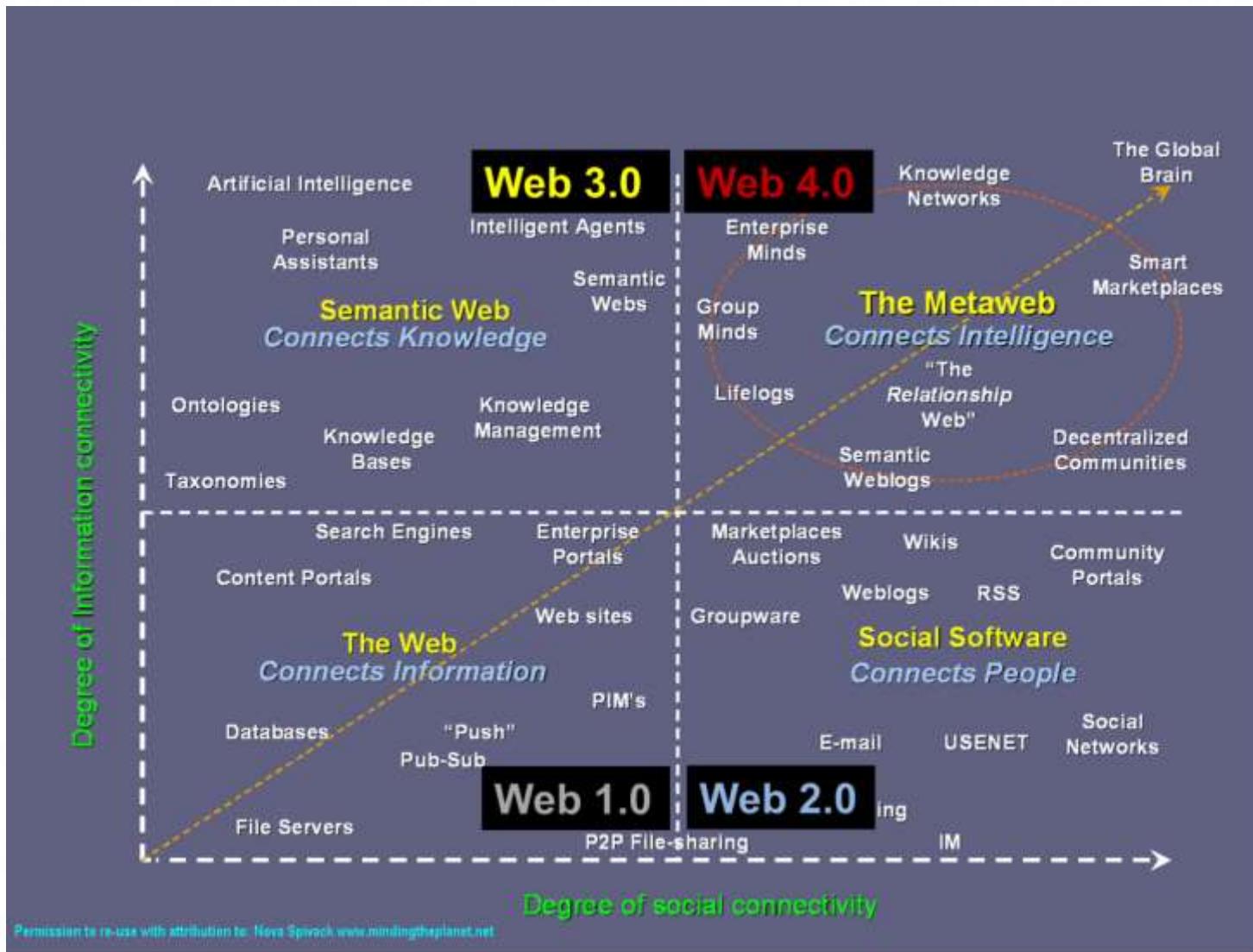
- Le Web 1.0 : Focalisé sur la documentation
  - Le Web 2.0 : Focalisé sur les utilisateurs
  - Le Web<sup>2</sup> : Focalisé sur les données
  - Le Web 3.0 : l'I.A (Intelligence artificielle)
  - Le Web 3.0 : le Meta Verre
  - Le Web3 : basée sur la blockchain
- Source : <http://www.w3b.fr/blog/evolution-d-internet-1990-2020.htm>

# EVOLUTION

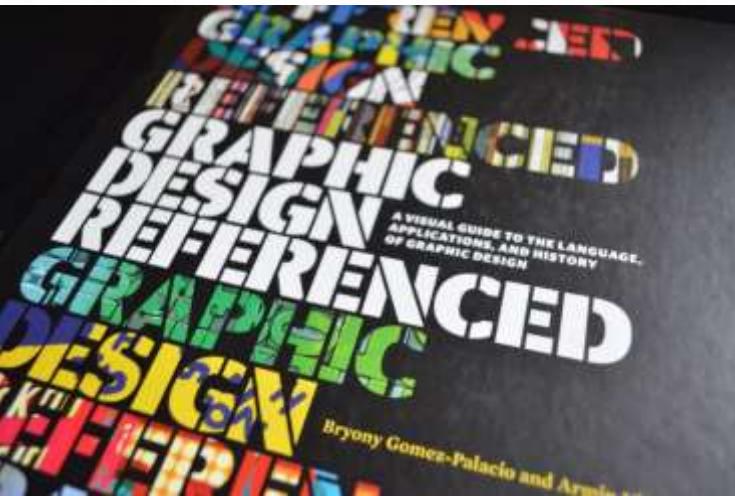
SOURCE : [HTTP://WWW.W3B.FR/BLOG/EVOLUTION-D-INTERNET-1990-2020.HTM](http://WWW.W3B.FR/BLOG/EVOLUTION-D-INTERNET-1990-2020.HTM)



# EVOLUTION



# EVOLUTION

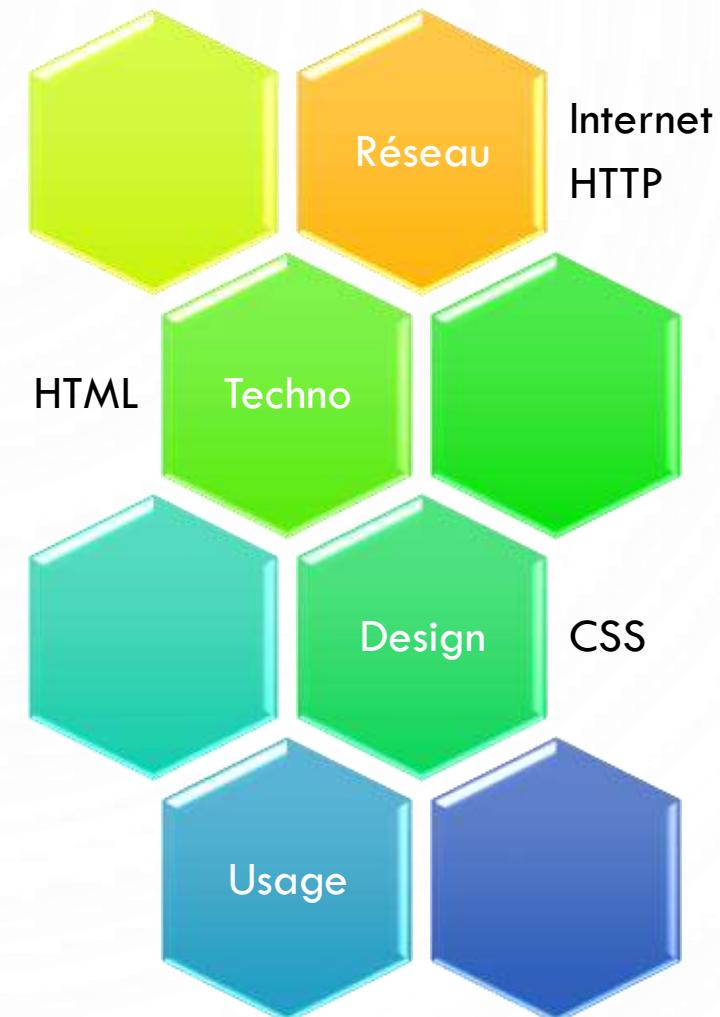
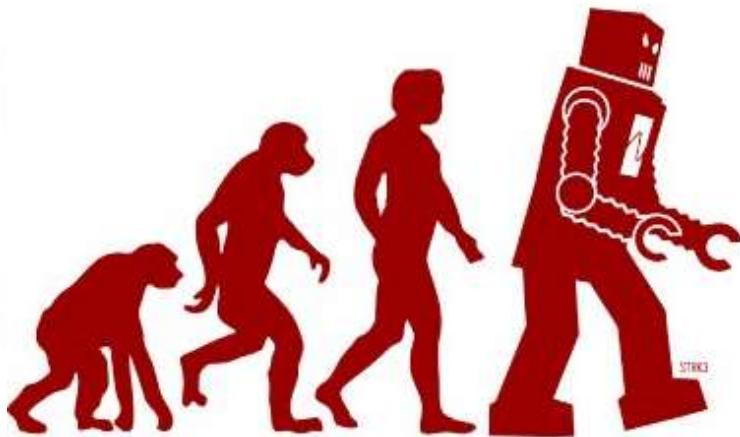


- Allez visiter :

- <https://o.nouvelobs.com/galeries-photos/high-tech/20120925.OBS3438/en-images-facebook-10-ans-d-evolution.html#modal-msg>

On en discute

# EVOLUTION



# INTERNET



## Les protocoles du web :

- IP ... IPs
- TCP
- DNS
- HTTP ... HTTPS
- SSH et FTP

Source : <http://parcoursnumeriques-pum.ca/les-protocoles-d-internet-et-du-web>

# INTERNET



*Pour permettre à deux ordinateurs d'échanger des informations entre eux, il faut un lien physique entre ces deux ordinateurs (éventuellement sans fil). Si on relie non pas deux, mais plusieurs ordinateurs qui vont pouvoir s'échanger des informations, on construit un réseau informatique.*

# INTERNET



*Internet est un ensemble de technologies qui permettent à plusieurs réseaux de s'interconnecter de manière à permettre l'échange d'informations entre ordinateurs connectés non seulement au même réseau, mais aussi sur des réseaux différents.*

*Pour gérer la transmission de données sur ce réseau de réseaux, deux protocoles sont utilisés et constituent le fondement d'Internet : IP, pour Internet Protocol, et TCP, pour Transfert Control Protocol.*

Source : <http://parcoursnumeriques-pum.ca/les-protocoles-d-internet-et-du-web>

# INTERNET



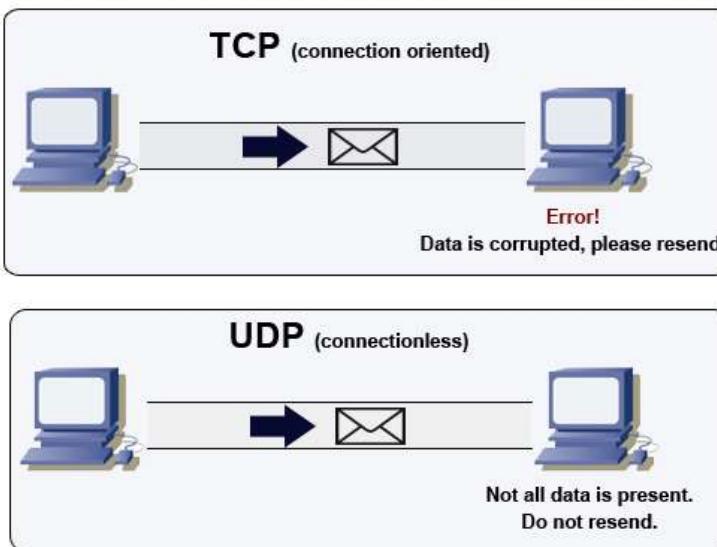
Le protocole **IP** permet d'attribuer une adresse unique à chaque ordinateur (nommée « adresse IP ») et fournit les mécanismes pour acheminer les données à bon port.

les « routeurs », ont pour fonction d'aiguiller les données de l'expéditeur au destinataire

Deux propriétés du protocole IP doivent être soulignées. La première est qu'il est non fiable, la seconde propriété est que la taille des paquets IP est limitée, au mieux à 1 280 octets, soit la taille d'un texte de 200 mots

Source : <http://parcoursnumeriques-pum.ca/les-protocoles-d-internet-et-du-web>

# INTERNET

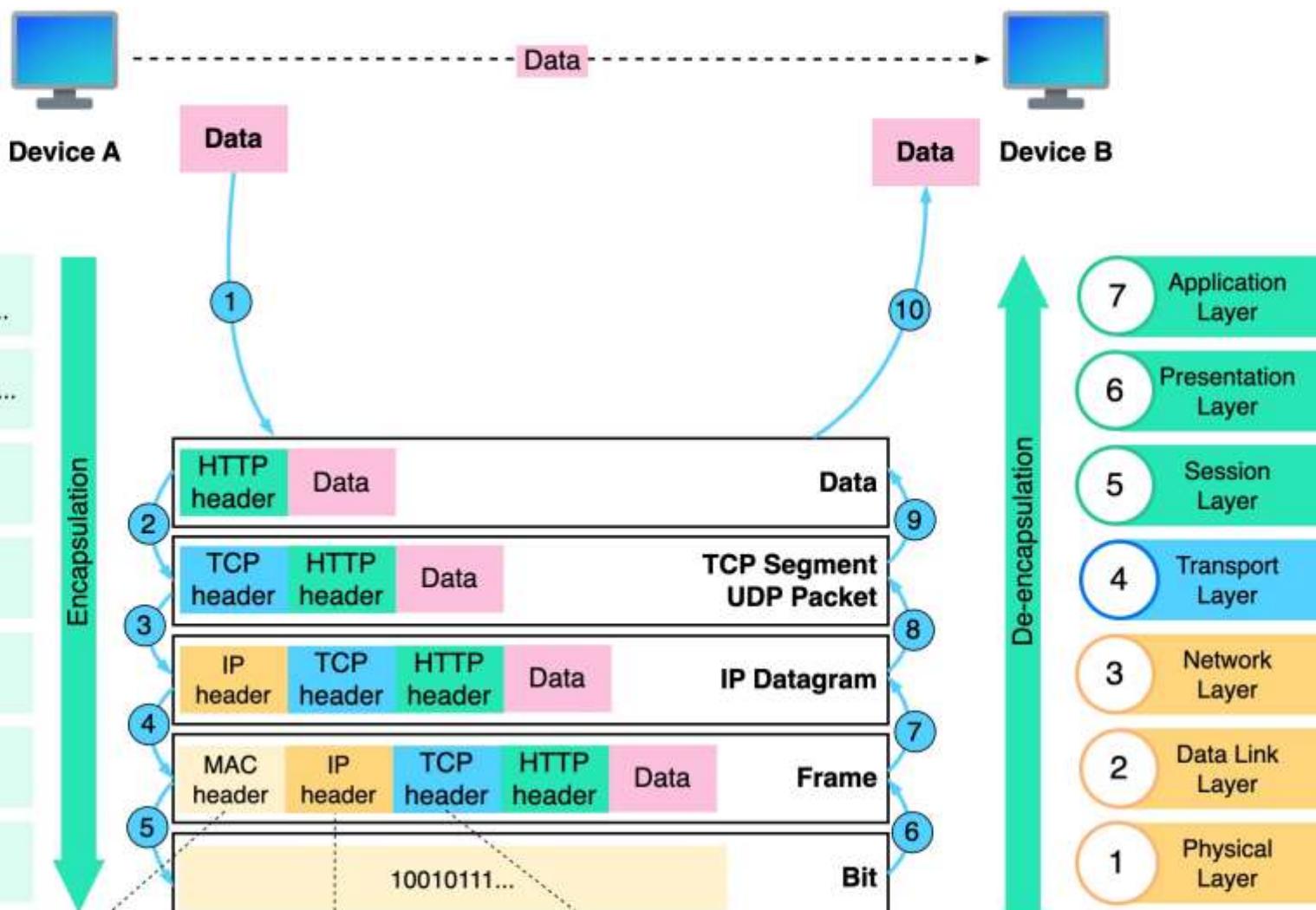
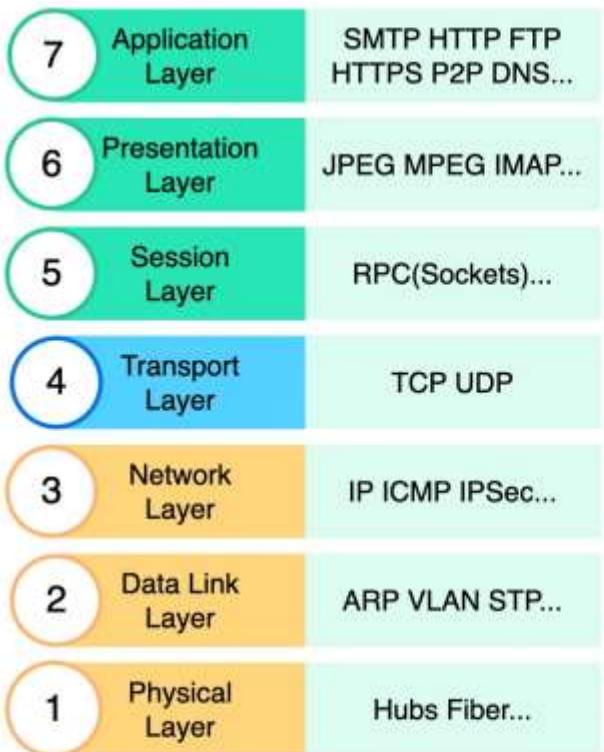


Le rôle du protocole **TCP** est de composer des échanges de paquets IP pour proposer des services plus adaptés aux types d'échanges d'information se déroulant sur Internet.

TCP est une couche de communication construite par-dessus la couche IP, encapsulant et masquant les détails du protocole IP.

# What is OSI model

<https://github.com/ByteByteGoHq/system-design-101#tcpip-encapsulation>



- source MAC address
- destination MAC address
- frame control
- sequence control

- source IP address
- destination IP address
- datagram sequence order

- source port
- destination port
- sequence number

# INTERNET



*Si les adresses IP permettent d'identifier les ordinateurs sur Internet, ayant été conçues à l'usage des machines, elles sont difficiles à mémoriser pour les humains :*

*Les applications construites sur Internet telles que le web ou le courrier électronique sont au contraire conçues pour être utilisées par les humains.*

*Pour pallier ce problème de lisibilité des adresses, le Domain Name System (DNS), a été conçu. Il permet que les machines, en parallèle à leurs adresses IP, soient identifiées par un « nom de domaine », plus facilement mémorisable.*

# INTERNET



Le web, ou plus précisément le World Wide Web, est un système documentaire construit sur Internet dans lequel les documents, nommés **hypertextes** ou pages web, sont reliés les uns aux autres par des **hyperliens**. Ces documents sont affichés dans des navigateurs qui permettent, grâce aux hyperliens, de naviguer d'une page à une autre. Les données qui sont échangées sur le web le sont avec le protocole HTTP, *HyperText Transfert Protocol*.

Source : <http://parcoursnumeriques-pum.ca/les-protocoles-d-internet-et-du-web>

# INTERNET



Lorsque deux machines s'échangent des données sur le web avec le protocole HTTP, leur rôle est asymétrique. L'une d'entre elles, le « serveur », a pour fonction de fournir des ressources ou des services, tandis que l'autre, le « client », utilise les ressources et les services du serveur

# INTERNET



Le protocole HTTP fonctionne selon un principe de requête/réponse : le client formule une requête auprès du serveur, lequel renvoie en retour une réponse au client

La première fois qu'un client envoie une requête HTTP à un serveur, il ouvre une connexion TCP avec le serveur (généralement sur le port 80). Il envoie alors sa requête à travers cette connexion, et le serveur envoie en retour sa réponse par la même connexion. Cette connexion TCP peut être maintenue ouverte pour des requêtes ultérieures.

# INTERNET



Les requêtes HTTP sont, déclenchées lorsqu'un utilisateur clique sur un lien hypertexte ou saisit une adresse dans son navigateur. Dans un cas comme dans l'autre, la requête est construite à partir d'une URL, *Uniform Resource Locator*.

# INTERNET

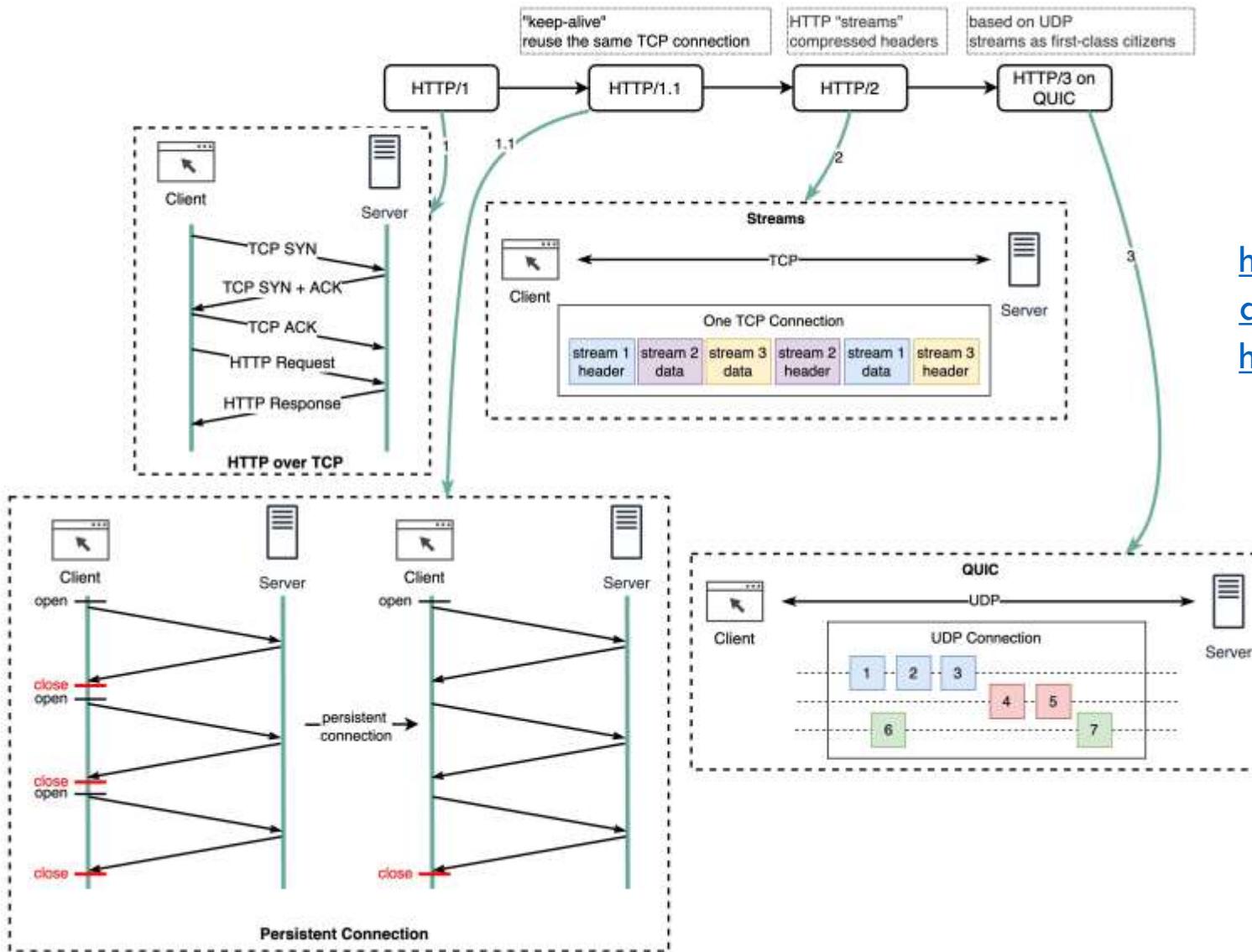


**Exemple**

[http://fr.wikipedia.org:80/w/index.php?title=Uniform Resource Locator&#38;oldid=93045880#URL absolue](http://fr.wikipedia.org:80/w/index.php?title=Uniform_Resource_Locator&oldid=93045880#URL_absolue)

# How did we get to HTTP/3?

ByteByteGo



<https://github.com/ByteByteGoHQ/system-design-101#http-10---http-11---http-20---http-30-quic>

# INTERNET

#ForDevelopers

There Are **5 Types** Of Status Codes

- Informational Responses → 100–199
- Successful Responses → 200–299
- Redirects → 300–399
- Client Errors → 400–499
- Server Errors → 500–599

@SourenaDev

Coding, Career guid, Productivity

BISGAMBIGLIA@UNIV-CORSE.FR

Les requêtes et réponses HTTP sont constituées de deux parties, un en-tête qui contient des informations propres au protocole HTTP, et un corps qui contient les données échangées

En-tête :

- 9 méthodes HTTP dont GET et POST
- cookie
- statut « 200 OK » « 404 Not Found »

Source : <http://parcoursnumeriques-pum.ca/les-protocoles-d-internet-et-du-web>

# INTERNET



**GET** : C'est la méthode la plus courante pour demander une ressource. Une requête GET est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.

**HEAD** : Cette méthode ne demande que des informations sur la ressource, sans demander la ressource elle-même.

# INTERNET



**POST** : *Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource (le plus souvent depuis un formulaire HTML). L'URI fourni est l'URI d'une ressource à laquelle s'appliqueront les données envoyées. Le résultat peut être la création de nouvelles ressources ou la modification de ressources existantes.* À cause de la mauvaise implémentation des méthodes HTTP (pour Ajax) par certains navigateurs (et la norme HTML qui ne supporte que les méthodes GET et POST pour les formulaires), cette méthode est souvent utilisée en remplacement de la requête PUT, qui devrait être utilisée pour la mise à jour de ressources.

Source : [https://fr.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

# INTERNET



**OPTIONS** : *Cette méthode permet d'obtenir les options de communication d'une ressource ou du serveur en général.*

**CONNECT** : *Cette méthode permet d'utiliser un proxy comme un tunnel de communication.*

**TRACE** : *Cette méthode demande au serveur de retourner ce qu'il a reçu, dans le but de tester et effectuer un diagnostic sur la connexion.*

# INTERNET



**PUT** : *Cette méthode permet de remplacer ou d'ajouter une ressource sur le serveur. L'URI fourni est celui de la ressource en question.*

**PATCH** : *Cette méthode permet, contrairement à PUT, de faire une modification partielle d'une ressource.*

**DELETE** : *Cette méthode permet de supprimer une ressource du serveur.*

# INTERNET



*Sur un serveur web, le serveur HTTP est responsable du traitement des requêtes reçues et de leurs réponses.*

*Une fois qu'il a reçu une requête, le serveur HTTP vérifie que l'URL demandée correspond à un fichier existant.*

*Si c'est le cas, le serveur envoie le fichier vers le navigateur du client. Sinon, le serveur d'applications génère le fichier nécessaire.*

*Si le fichier n'existe pas ou que le traitement est impossible, le serveur web renvoie un message d'erreur au navigateur. Le message d'erreur le plus fréquemment rencontré est « 404 Page non trouvée »*

# INTERNET



*FTP : échange de fichiers*

*SSH : connexion à distance en ligne de commande*

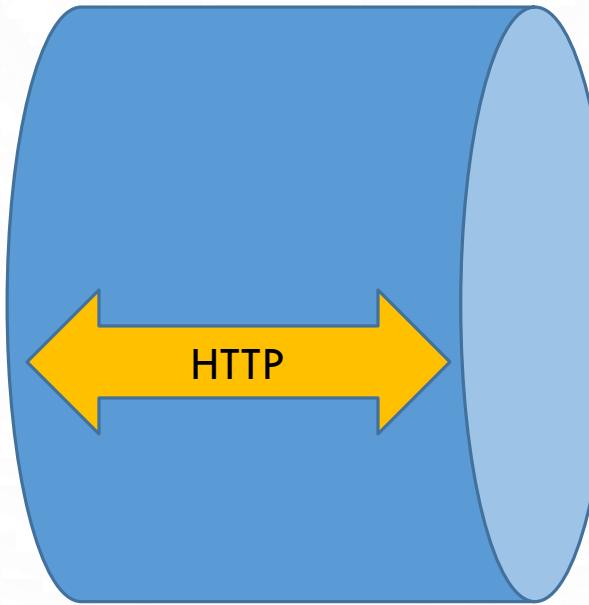
Réf : Magué Jean-Philippe (2014). “Les protocoles d’Internet et du web”, in E. Sinatra Michael, Vitali-Rosati Marcello (édité par), Pratiques de l’édition numérique, collection « Parcours Numériques », Les Presses de l’Université de Montréal, Montréal, p. 129-144, ISBN: [978-2-7606-3202-8](http://978-2-7606-3202-8)

Source : <http://parcoursnumeriques-pum.ca/les-protocoles-d-internet-et-du-web>

# MODÈLE CLIENT SERVEUR



Navigateur (client)



Le serveur est la partie non visible d'un site ou d'un service web. Mais il en est la base !



Ordinateurs (serveur)

# LANGUAGE



- **HTML 5**

- Une API de dessin 2D, grâce à la nouvelle balise canvas
- Une API pour jouer des vidéos et des sons/musiques permis grâce aux balises video et audio;
- Une API utilisée pour les applications hors-ligne;
- Une API d'édition en combinaison grâce à l'attribut contenteditable
- Une API de « glisser-déposer » en combinaison avec l'attribut draggable;
- Une API qui permet l'accès à l'historique et qui donne la possibilité aux pages d'en ajouter pour prévenir les problèmes de bouton retour-en-arrière.
- Une API de géo-localisation
- Une API permettant d'analyser de reproduire une page HTML grâce à la balise inner-HTML

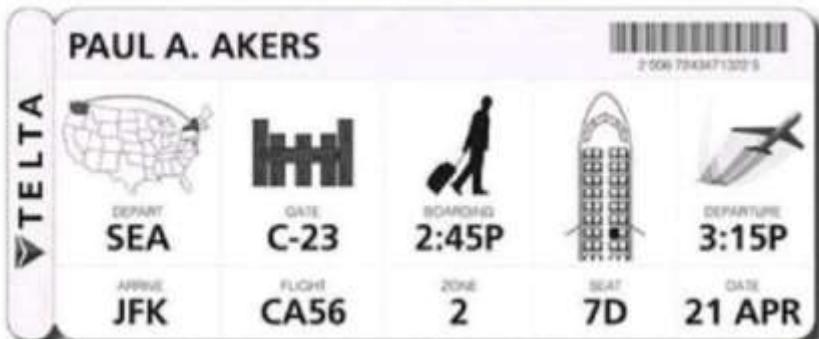
Source : <http://www.commentcamarche.net/faq/30122-html-5-qu'est-ce-qui-change>

# DESIGN & ERGONOMIE

This is a boarding pass



This is a Lean boarding pass



Qu'est-ce que l'ergonomie d'un site internet ?

**Ergonomie  
Design  
IHM**

# DESIGN & ERGONOMIE

## UX : LES 10 PIRES ERREURS EN 2021



Popups et fenêtres modales



Lenteur du temps de réponse



Liens trompeurs



Faible contraste et texte minuscule



Saisie contrainte



Impossibilité de sélectionner  
et de copier



Icônes sans intitulé



Changement de mise en page  
au chargement



Visuel encombrant



Affichage sur grand écran

# ERGONOMIE LOGICIEL

Une petit guide issu du travail de Dominic Scapin et de Christian Bastien :  
Bastien, J.M.C., Scapin, D. (1993) Ergonomic Criteria for the Evaluation of Human-Computer interfaces. Institut National de recherche en informatique et en automatique, France.

**1.Guidage**

**2.Charge de travail**

**3.Contrôle explicite**

**4.Adaptabilité**

**5.Gestion des Erreurs**

**6.Homogénéité/Cohérence**

**7.Signifiance des Codes et**

**Dénominations**

**8.Compatibilité**

# DESIGN & ERGONOMIE



- **l'utilité d'un site** : savoir rendre les contenus d'un site intéressants
- **l'utilisabilité du site** : savoir rendre un site simple d'accès et surtout facile à utiliser
- **le design graphique** : comme on dit "si on peut joindre l'utile à l'agréable". Un beau site restera d'autant plus facilement ancré dans la mémoire, si vous trouvez rapidement ce que vous recherchez ... ce ne sera que du bonheur

Source : <https://www.keacrea.com/blog/les-regles-indispensables-de-l-ergonomie-d-un-bon-site-web>

# DESIGN & ERGONOMIE



- Une page d'accueil claire et précise
- Des textes lisibles
- Pas de gadget
- Un affichage rapide
- Un plan de site lisible
- Des menus accessibles
- Mettre savamment en avant vos contenus
- Un accès en 3 clics
- Un design original et esthétique

Source : <https://www.keacrea.com/blog/les-regles-indispensables-de-l-ergonomie-d-un-bon-site-web>

# DESIGN & ERGONOMIE

A lire

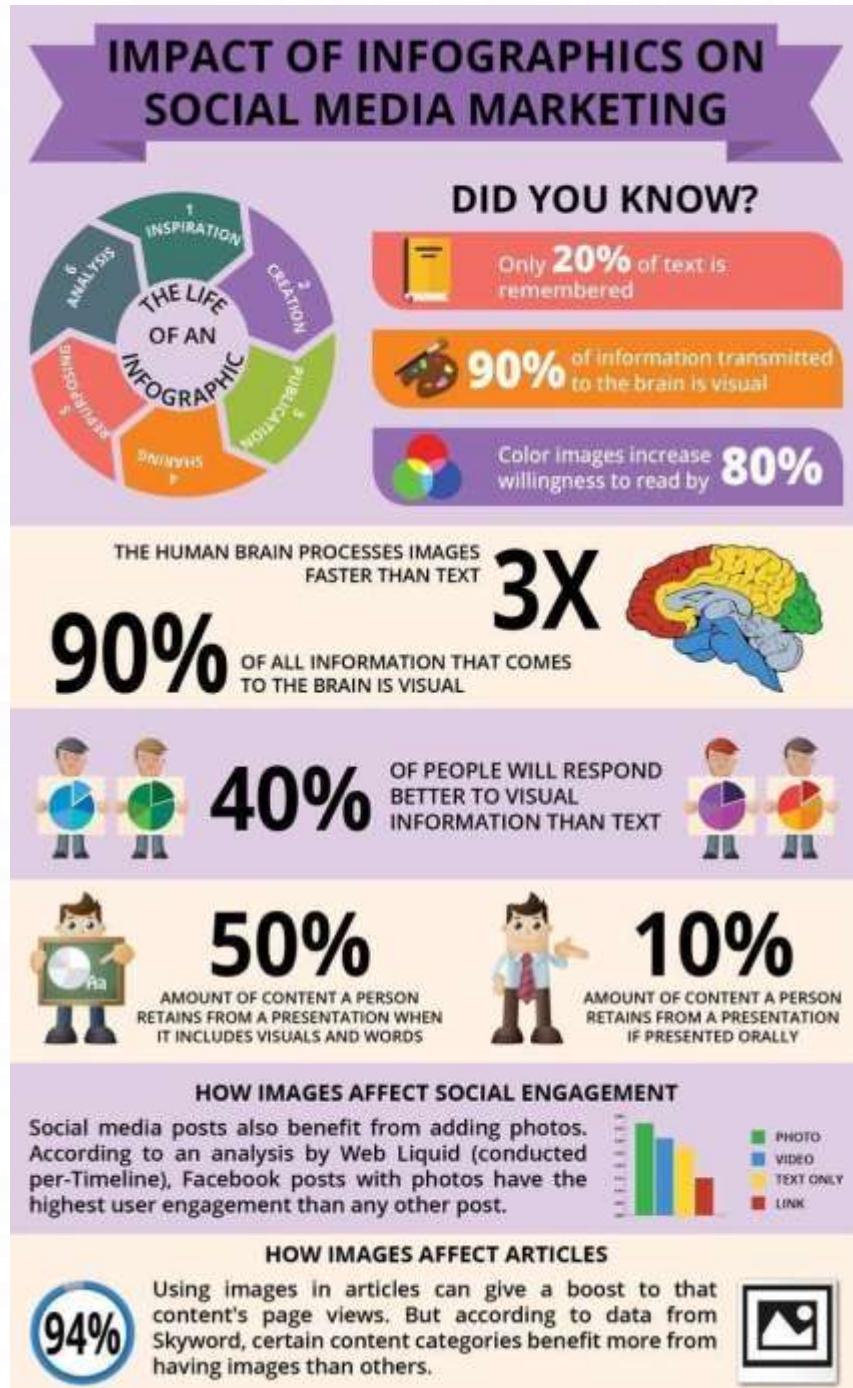
[https://www.keacrea.com/blog/ergonomie-web-  
qu-est-ce-que-c-est](https://www.keacrea.com/blog/ergonomie-web-qu-est-ce-que-c-est)

<http://www.mon-design-web.com/standards.php>

[http://fr.wix.com/blog/2016/01/12/les-  
grandes-tendances-du-web-design-2016/](http://fr.wix.com/blog/2016/01/12/les-grandes-tendances-du-web-design-2016/)

*On en parle !*

Source : <https://www.keacrea.com/blog/les-regles-indispensables-de-l-ergonomie-d-un-bon-site-web>



# SÉCURITÉ



# RESPECT DE LA VIE PRIVÉ

**Privacy matters.**



# USAGE



Focalisé sur la documentation

- Consultation / lecture

Focalisé sur les utilisateurs

- Réseaux sociaux
- Blog

Focalisé sur les données

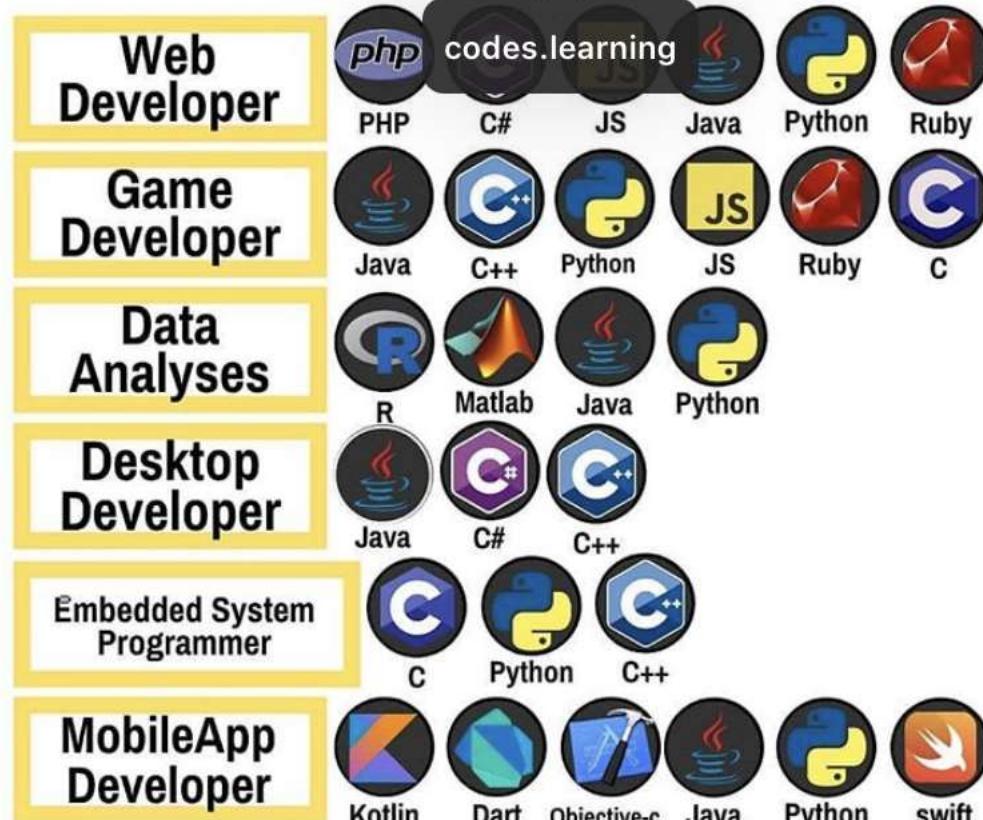
- Big data

# TENDANCES



## PROGRAMMING LANGUAGE

By- @codes.learning



# TENDANCES



## Programming Languages First Appearances

Name & Year	Designed / Written By
C 1972	Dennis Ritchie
C++ 1985	Bjarne Stroustrup
C# 2000	Microsoft Corporation
GoLang 2009	Rob Pike & Ken Thompson
Java 1995	James Gosling
Javascript 1995	Brendan Eich
Julia 2012	Jeff, Alan, Stefan, Viral
Kotlin 2011	Jet Brains
Matlab 1970	Cleve Moler
PHP 1995	Rasmus Lerdorf
Python 1990	Guido van Rossum
R 1993	Ross Ihaka, Robert Gentleman
Ruby 1995	Yukihiro Matsumoto
Swift 2014	Chris Lattner & Apple
.Net 2001	Andres Hejlesberg

WORLDWIDE ENGINEERING

# TENDANCES



## Free Hosting Platforms

### BACKEND -

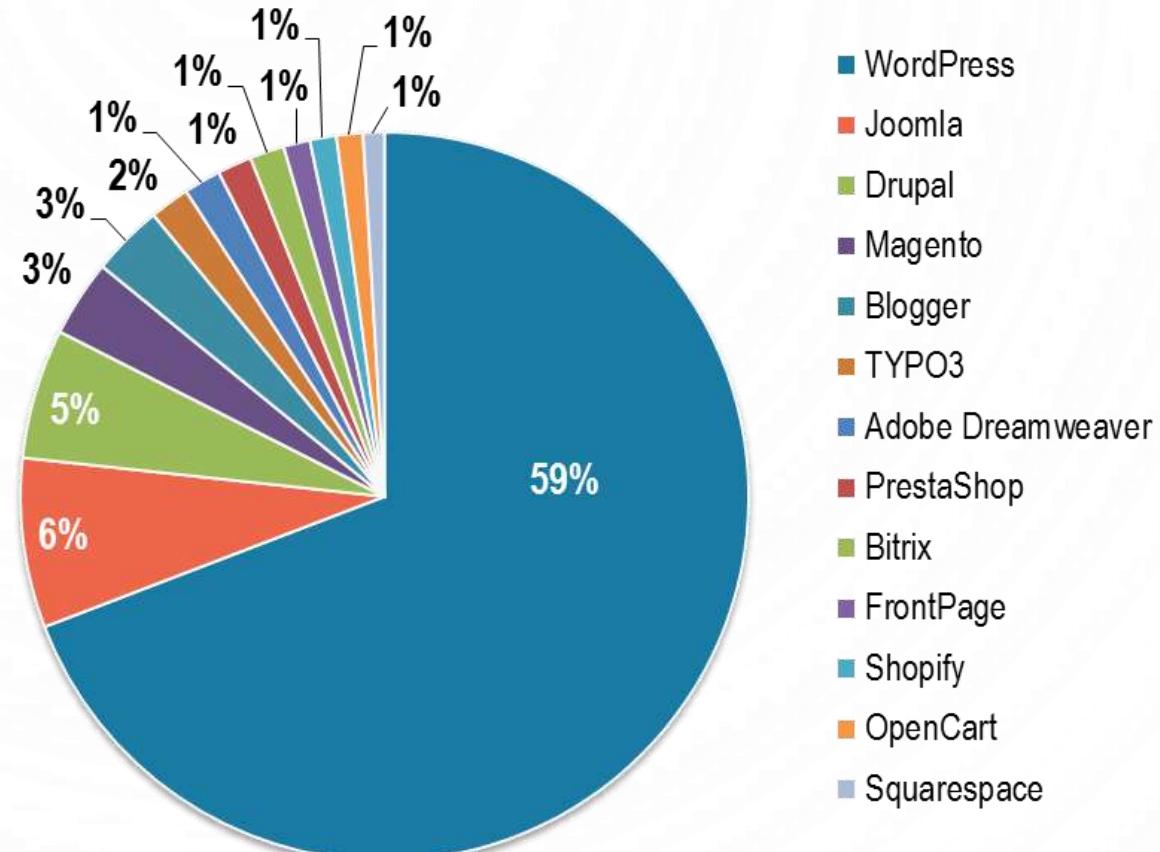
1. Vercel
2. Heroku
3. Google Cloud
4. AWS
5. Digital Ocean
6. Heroku

### FRONTEND -

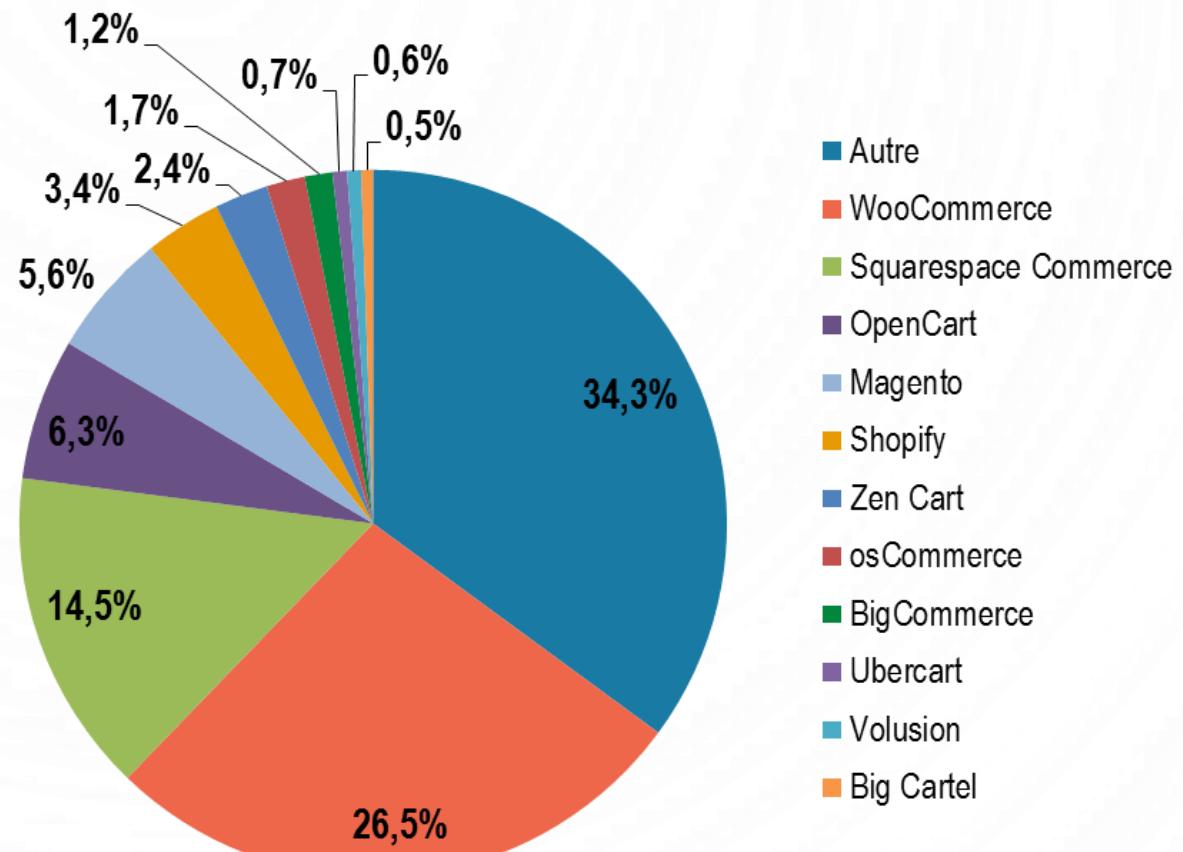
1. GitHub Pages
2. Netlify
3. Vercel
4. Surge
5. Firebase

@Python.Learning

# TENDANCES

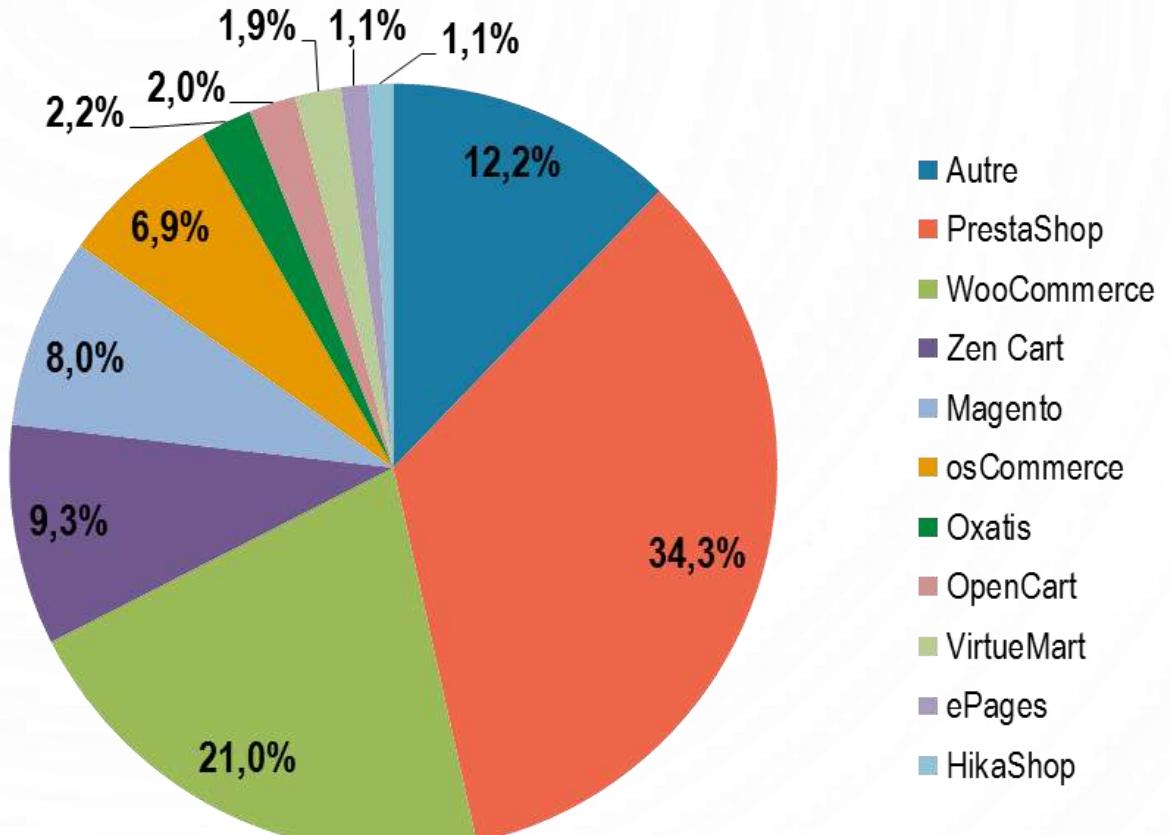


# TENDANCES



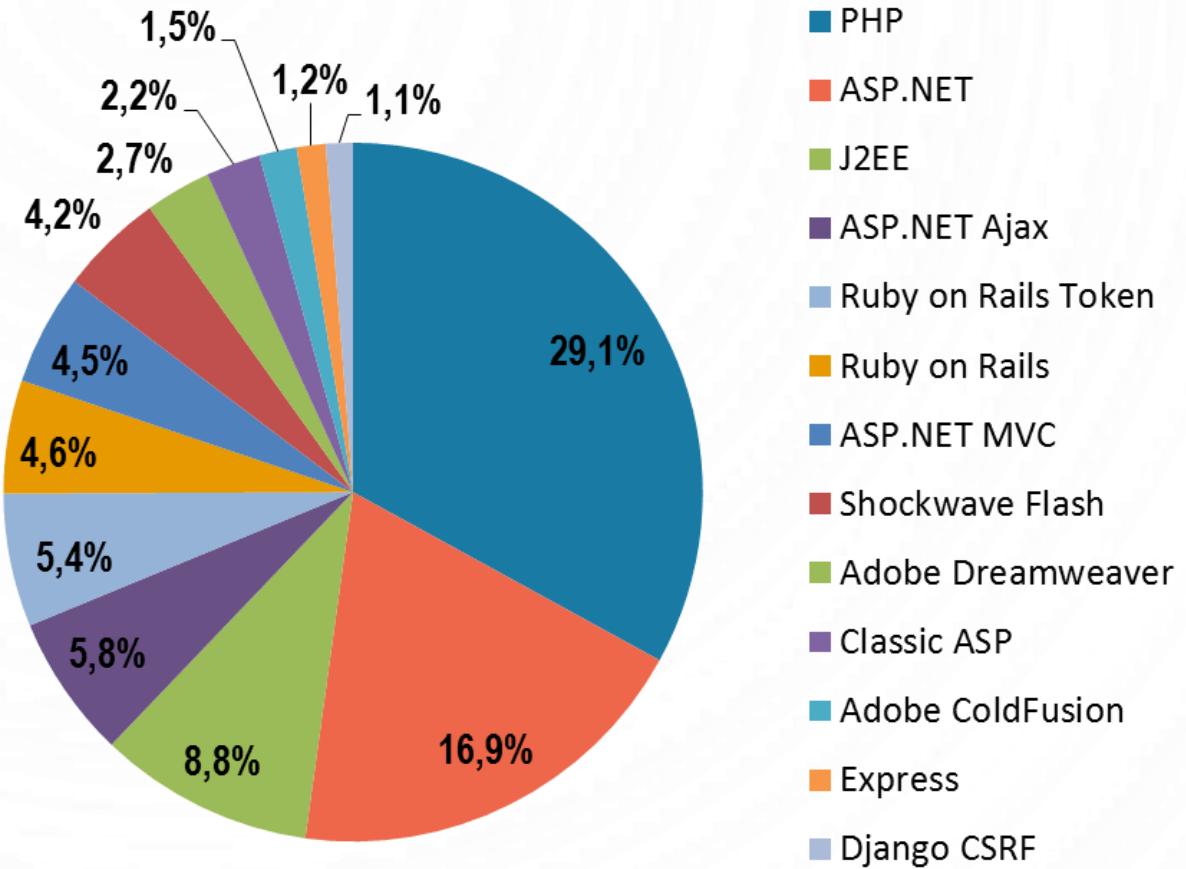
Source : [Builtwith](#)

# TENDANCES



Source : [Builtwith](#)

# TENDANCES



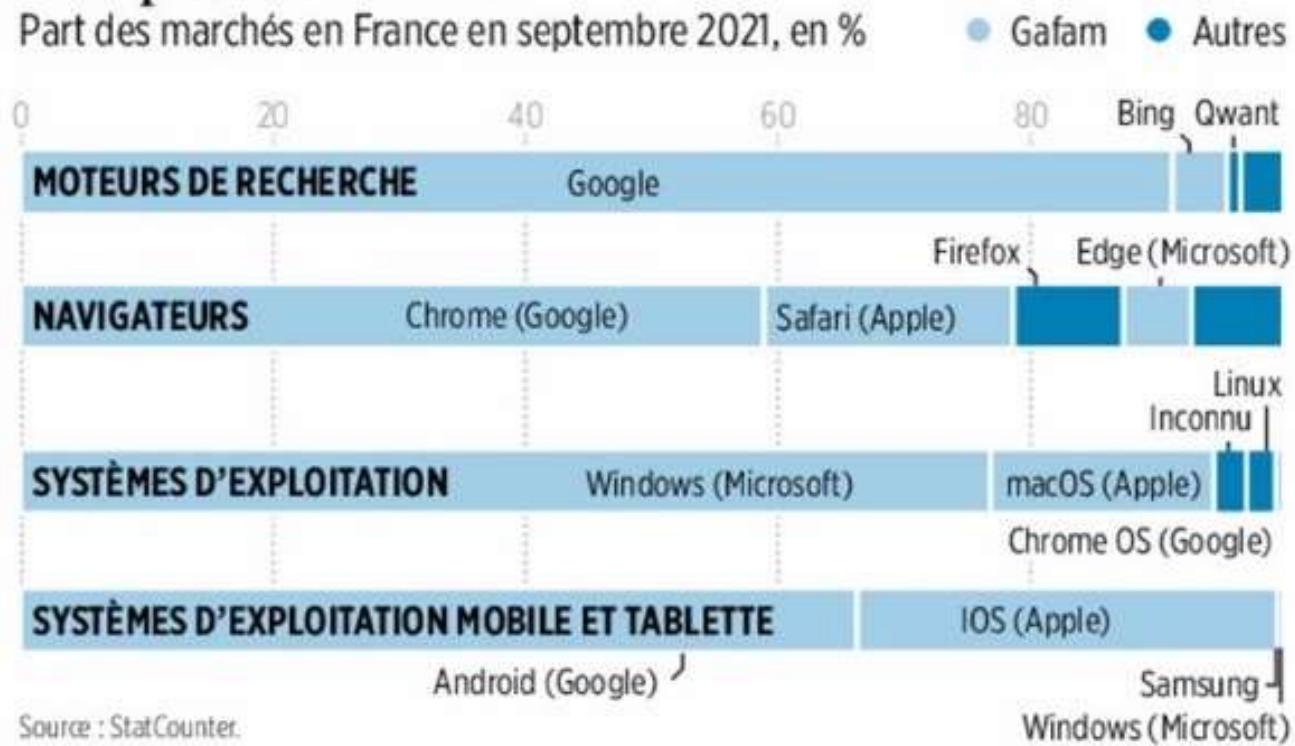
Source : [Builtwith](#)

# TENDANCES



## Omniprésence en France

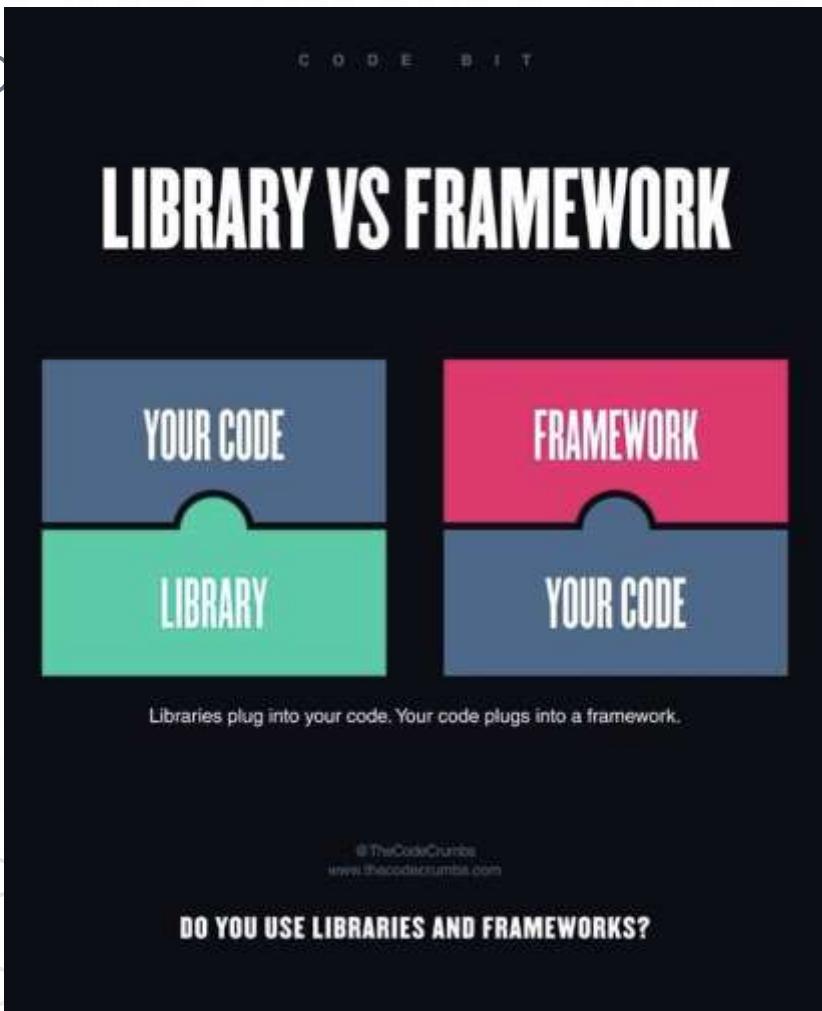
Part des marchés en France en septembre 2021, en %





**Quels sont les avantages à  
utiliser un Framework Javascript**

?



Quels sont les avantages à  
utiliser un Framework Javascript  
?

# TENDANCES



## LIBRAIRIES JS

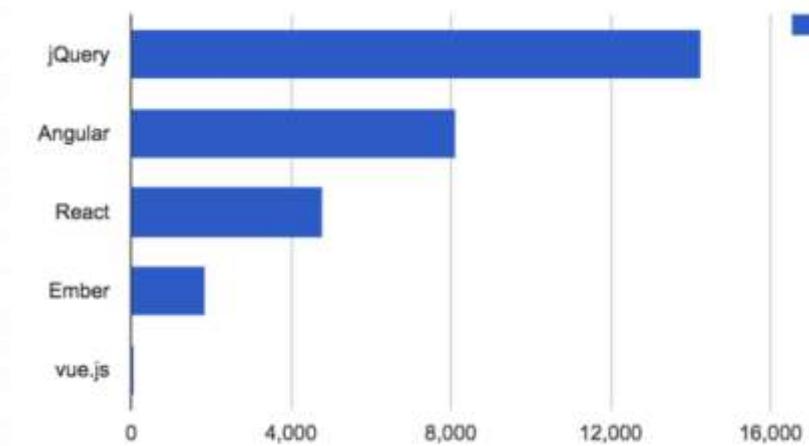
Nom	Tout le web	Top 10k	Top 100k	Top Million
jQuery	34,1%	12,4%	17,1%	20,1%
Google Libraries	6,8%	6,9%	7,8%	5,2%
html5shiv	6,2%	5,9%	6,4%	5,6%
jQuery UI	4,9%	5,1%	6,0%	6,7%
Facebook for Websites	4,6%	7,3%	7,3%	7,0%
Modernizr	3,6%	5,5%	5,4%	3,9%
jQuery Form	3,5%	1,6%	2,0%	2,7%
Facebook SDK	3,3%	6,1%	6,0%	5,7%
jQuery Easing	3,2%	1,5%	2,0%	2,5%
Fancybox	3,0%	2,0%	2,3%	3,0%
jQuery Cycle	2,3%	1,6%	2,1%	2,4%
Lightbox	2,1%	1,3%	1,7%	2,5%
FlexSlider	1,8%	1,2%	1,5%	1,6%
SuperFish	1,7%	0,8%	1,1%	1,6%

Source : [Builtwith](#)

## FRAMEWORKS JS

Nom	Tout le web	Top 10k	Top 100k	Top Million
Angular JS	90,5%	47,3%	47,5%	47,5%
Backbone.js	6,6%	51,4%	50,8%	51,0%
Meteor	1,9%	0,1%	0,3%	0,3%
Ember	0,6%	1,0%	1,1%	1,0%
SailsJS	0,5%	0,3%	0,3%	0,2%

Source : [Builtwith](#)



<https://medium.com/javascript-scene/top-javascript-frameworks-topics-to-learn-in-2017-700a397b711>

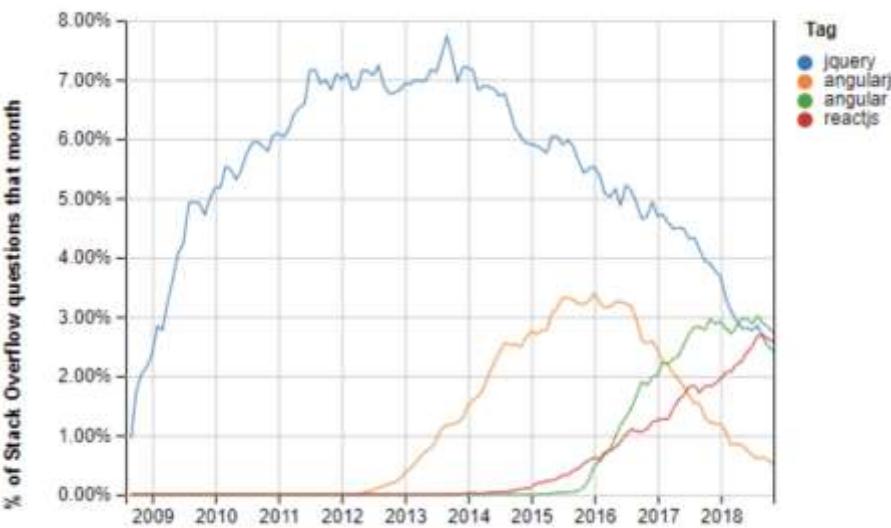
# TENDANCES



Les tendances

<https://trends.google.fr/trends/?geo=FR>

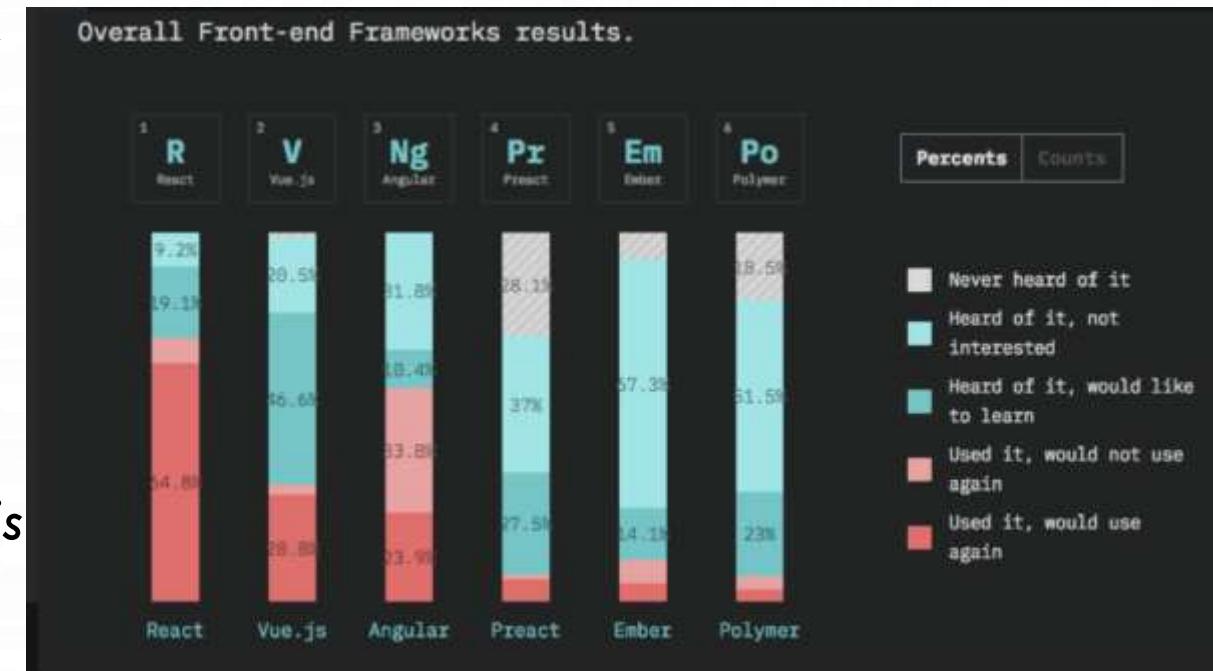
# TENDANCES



<https://divante.com/blog/top-10-popular-javascript-frameworks-2019/>

- TOP 5 des Frameworks JavaScript 2017 d'après <https://www.inteam.fr/blog/protege-top-5-frameworks-javascript-2017/>

1. Angular
2. React
3. Vue.js
4. Ember.js
5. Meteor.js



<https://www.easypartner.fr/blog/framework-javascript-les-tendances-2019/>

# A SUIVRE



Les 30 sites les plus populaires pour réaliser une veille efficace

- <http://www.blogduwebdesign.com/ressources/30-sites-populaires-realiser-veille-efficace/2151>

# LE COURS

- Programmation logique & **Programmation web**
  - **HTML**
  - **CSS**
  - **JS**
  - **PHP**

- 1) **Introduction**
  - 1) Le web, et ses techno
- 2) **Les bases**
  - 1) HTML : langage de balise <body> </body>
  - 2) Ergonomie du web
    - 1) CSS 3 pour la mise en forme
    - 2) Framework comme Bootstrap
  - 3) JavaScript
- 3) **Dynamisme côté client**
  - 1) jQuery
- 4) **Dynamisme côté serveur**
  - 1) PHP



**HTML**

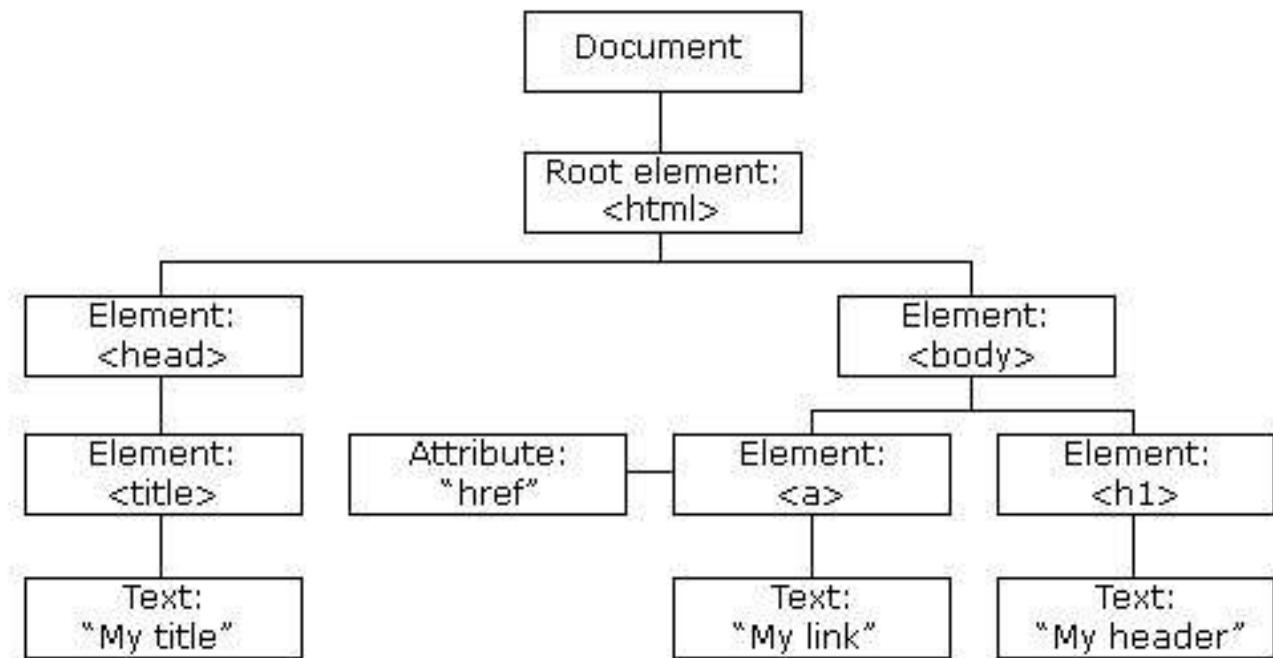
# **HTML**

**Les bases !!!**

# HTML

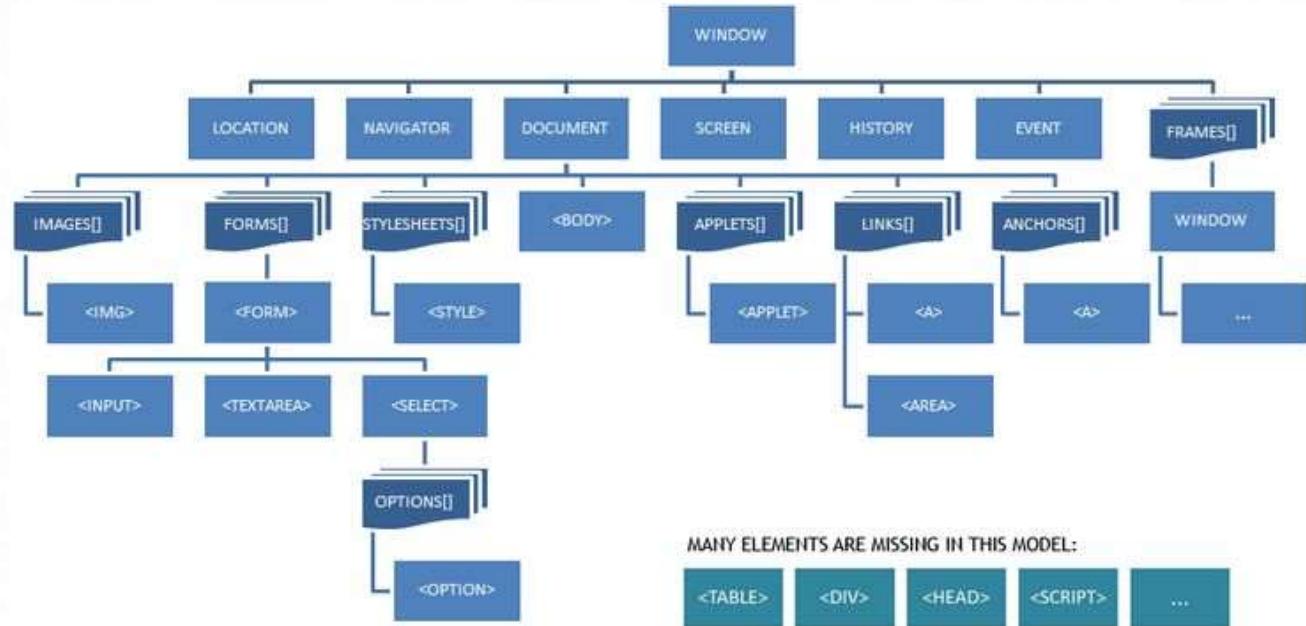
HTML

HTML DOM Tree:



# HTML

# HTML



Src : <https://fr.pinterest.com/pin/87679523969613722/>

# HTML

# HTML

```
<!DOCTYPE html>

<head>

<META NAME="Content-language" CONTENT="french">

<META http-equiv="Content-Type" content="text/html; charset=utf-8">

<META NAME="description" CONTENT="Mon super site">

<META NAME="keywords" CONTENT="cours master programmation S2I web site">

<META NAME="Author" CONTENT="PA BISGAMBIGLIA">

<META HTTP-EQUIV="Reply-to" CONTENT="bisgambiglia@univ-corse.fr">

<META NAME="Rating" CONTENT="General">

<link rel="stylesheet" media="screen" href="include/css/screen.css" type="text/css"
/>

<link rel="stylesheet" media="print" href="include/css/print.css" type="text/css" />

<link rel="stylesheet" media="screen" and (max-width: 810px)
href="include/css/smallScreen.css" type="text/css" />

<title>Mon site pour les cours 2016</title>

<script></script>

</head>
```

# HTML

# HTML

```
<body>
<div>
<header>
<div></div>
<nav id="menuNav">
    <ul>
        <li><a href="#">home</a></li>
        <li><a href="html/indexHTML.html">html</a></li>
        <li><a href="html/css/indexCSS.html">css</a></li>
        <li><a href="html/js/indexJS.html">js</a></li>
        <li><a href="php/indexPHP.php">php</a></li>
        <li><a href="html/contact.html">contact</a></li>
        <li><a href="html/ressources.html">liens</a></li>
    </ul>
</nav>
<div id="rubrique">Accueil</div>
</header>
```

# HTML

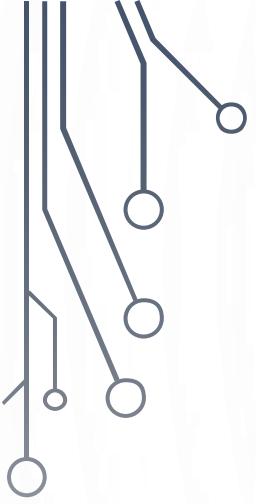
# HTML

```
<div id="corps">
<h1 id="titre01">Mon titre principal</h1>
    <article class="newspaper">
        <h2><strong>Mon sous-titre</strong></h2>
            <p>mon paragraphe de texte avec des
<strong>mots importants</strong> pour améliorer le référencement
            </p>
            <p>et on met la suite de mon texte</p>
            <p> ... </p>
            <h2><strong>Mon
                sous-titre
                2</strong></h2>
                <p> ...
                </p>
            </article>
    </div>
    <footer id="piedPage">
        <div id="mLegales"><p> Site Paul-Antoine Bisgambiglia 2016-
        2017 </p>
        </div>
    </footer>
</body>
</html>
```

# LE COURS

- Programmation logique & **Programmation web**
  - **HTML**
  - **CSS**
  - **JS**
  - **PHP**

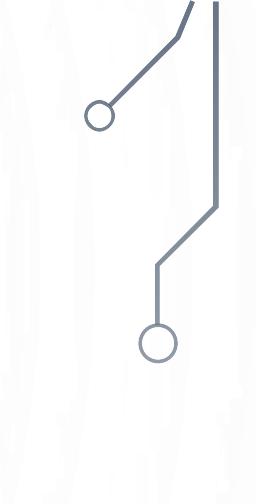
- 1) **Introduction**
  - 1) Le web, et ses techno
- 2) **Les bases**
  - 1) HTML : langage de balise <body> </body>
  - 2) Ergonomie du web
    - 1) CSS 3 pour la mise en forme
    - 2) Bootstrap
  - 3) JavaScript
- 3) **Dynamisme coté client**
  - 1) jQuery
- 4) **Dynamisme coté serveur**
  - 1) PHP



**CSS**

# **CSS**

**Les bases !!!**





CSS



CSS

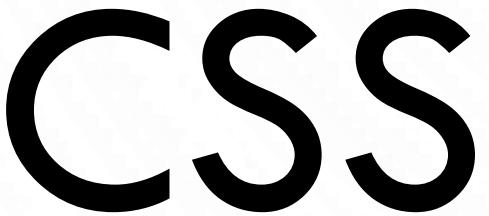
print.css

```
#menuNav, #footer, #rubrique, #piedPage,  
aside{  
    display:none;  
}
```

```
body {  
    font-size:120%;  
    color:black;  
}
```



CSS



CSS



## screen.css

```
p {  
    font-size: 9vm;  
    font-family: trebuchet, helvetica, sans-serif;  
}  
/*Titre principal*/  
#titre01{  
    text-align: center;  
    margin-top: 5.4vh;  
    color: #2C75FF;  
    font-family: trebuchet, helvetica, sans-serif;  
    font-size: 10vmin;  
    font-size: 10vmaw;  
}
```



# CSS

## screen.css

```
input[type=button], input[type=submit], input[type=reset] {  
    background-color: #3A8EBA;  
    border: none;  
    color: white;  
    padding: 16px 32px;  
    text-decoration: none;  
    margin: 4px 2px;  
    cursor: pointer;  
    border-radius: 6px;  
}
```



# CSS et variable

var( <custom-property-name> , <declaration-value>? )

```
:root { --main-bg-color: brown; }

.un { color: white; background-
color: var(--main-bg-color);
margin: 10px; width: 50px;
height: 50px; display: inline-
block; }
```

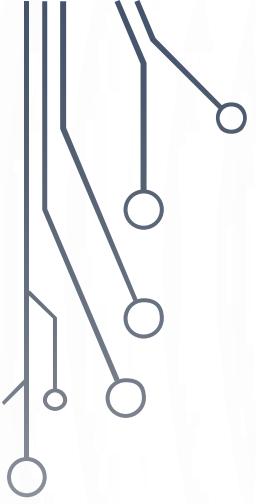
[https://developer.mozilla.org/fr/docs/Web/CSS/Using\\_CSS\\_custom\\_properties](https://developer.mozilla.org/fr/docs/Web/CSS/Using_CSS_custom_properties)

# work CSS

# Frame

- Bootstrap
- Foundation
- Bulma
- W3school

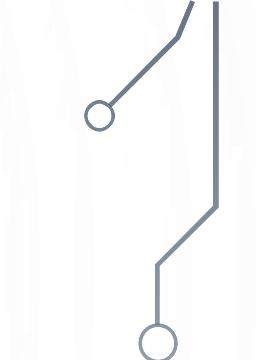
<https://www.codeur.com/blog/front-end-framework/>



JAVASCRIPT



JS



Les bases !!!

# JAVASCRIPT

## Historique

- Crée par Netscape (LiveScript) ;
- En 1995, il devient JavaScript (Firme Sun) ;
- Apparaît d'abord dans la version Netscape 2.0 ;
- Il est défini par la norme ECMA-262 ou ECMAScript ;
- Différentes versions : 1.0 à 1.7 et 2.0.

# JAVASCRIPT



JavaScript

- C'est un langage de scripts qui est incorporé aux balises Html et qui permet d'agrémenter la présentation et l'interactivité des pages web ;
- Langage de programmation basé sur les objets
- JavaScript est inspiré de Java

# JAVASCRIPT

JavaScript	Java
Code intégré dans la page HTML	Applet distinct de la page HTML
Code interprété par le navigateur au moment de l'exécution	Code source compilé avant son exécution
Script (ligne de code)	Applet (petites applications)
Confidentialité des codes nulle (code source visible et accessible)	Sécurité avec des codes compilés est donc illisibles
Codes de programmation simples mais pour applications limitées	Langage de programmation plus complexe mais plus performant
Permet l'accès aux objets du navigateur	N'accède pas aux objets du navigateur

# JAVASCRIPT



- Comment intégrer du code JavaScript dans une page HTML ?
- Avec la balise Script
- Avec un fichier externe
- Avec les événements

# JAVASCRIPT



```
<HTML>
<HEAD>
<TITLE> titre de la page </TITLE>
<SCRIPT language="JavaScript">
<!--
//-->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT type="text/JavaScript">
<!--
//-->
</SCRIPT>
</BODY>
</HTML>
```

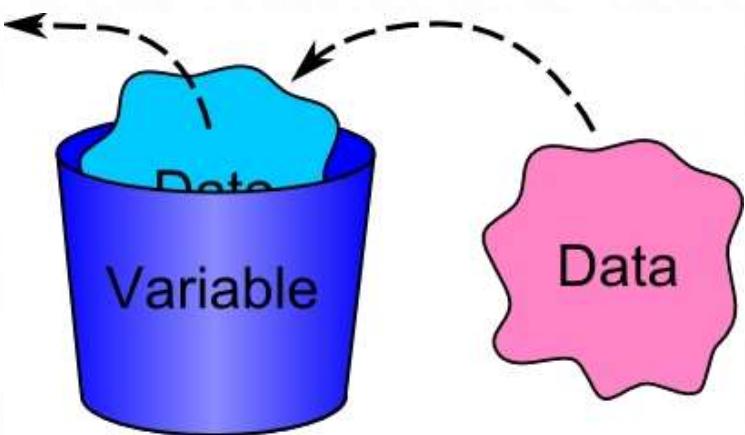
# JAVASCRIPT

- Avec un fichier externe
- Les codes sont en annexe dans un fichier ;
- on insère juste :

```
<SCRIPT  
LANGUAGE=JavaScript  
SRC="url/fichier.js ">  
defer> </SCRIPT>
```

- Avec les événements
- Gestionnaires d'événements :  
`onEvenement="Action_JavaScript_ou_Fonction()  
;"`
- Exemple du : <FORM>  
`<INPUT TYPE="button" VALUE="Cliquez ici"  
onClick="alert('Vous avez bien cliqué ici')">  
</FORM>`

# JAVASCRIPT



## Variables : var

- Déclaration : `var maVariable = 12;`
- JavaScript utilise l'instruction `var` pour la déclaration. Toute nouvelle variable doit être initialisée. Le type est dynamique :
- `var prenom_visiteur="Marcel";`
- `var nom_visiteur="Dupond";`
- `var age_visiteur=29;`
- Utilisation :
- `var accueil="Bonjour " + prenom_visiteur + " " + nom_visiteur;`

# JAVASCRIPT

- À lire :
- <https://developer.mozilla.org/fr/docs/Outils/Débogueur>
- <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/débugger>
- <https://developer.mozilla.org/fr/docs/Web/API/Console>

## Débogage javascript

- La console du navigateur
- L' instruction **debugger** permet de faire appel à un outil de débogage
- `console.log(" ")`
- `console.error(" ")`
- `console.warn(" ")`
- // Script entier en mode strict
- `"use strict";`

# JAVASCRIPT

## Chaînes de caractères :

- Pour déclarer une chaîne de caractères, vous pouvez utiliser les guillemets ("") ou l'apostrophe (').
- Concaténation +
- Une chaîne de caractères en javascript est un objet string sur lequel s'appliquent des propriétés et des méthodes.
- La méthode charAt(n) récupère le caractère n.
- La méthode substring extrait une sous-chaîne, elle attend 2 paramètres :
  - l'indice du premier caractère (inclus),
  - l'indice du dernier caractère (exclus).

# JAVASCRIPT

## Chaînes de caractères :

- Quelques exemples utiles :
- Vérifier qu'un mail est valide, il suffit de tester la présence de @ et du point.

```
function verifiermail(mail) {  
    if ((mail.indexOf("@")>=0)&&(mail.indexOf(".")>=0)) {  
        return true  
    } else {  
        alert("Mail invalide !");  
        return false  
    }  
}
```

# JAVASCRIPT

## Chaînes de caractères :

- Quelques exemples utiles :
- JavaScript offre deux méthodes pour transformer les lettres (et uniquement les lettres) d'un mot en majuscules ou en minuscules.
  - var chaine="Ceci est un texte";
  - var maj=chaine.toUpperCase();
  - var min=chaine.toLowerCase();
- fonction qui ajoute une majuscule à la première lettre d'un mot.
- Par exemple pierre, Pierre ou PIERRE deviennent Pierre.

```
function nompropre(mot) {  
    var m=mot.charAt(0).toUpperCase() ;  
    m = m + mot.substring(1).toLowerCase();  
    return m;  
}
```

# JAVASCRIPT



## Objets :

- La déclaration se fait toujours avec var. Pour créer un objet, il faut utiliser le mot-clé new suivi du type d'objet Date. ATTENTION, le respect des majuscules/minuscule est indispensable et source de nombreuses erreurs
- `var date_jour=new Date();`
- Cette ligne crée un objet Date contenant la date du jour.
- `var une_date=new Date(annee,mois-1,jour,heure,min)`
- Cette ligne crée un objet date avec une date paramétrable.

# JAVASCRIPT



## L'objet Math :

- En JavaScript, la plupart des fonctions mathématiques sont des méthodes de l'objet Math.
- Voici sûrement la fonction la plus utilisée, la génération d'un nombre aléatoire.
- `var nb=Math.random();`
- La méthode `random()` est appliquée à l'objet Math et retourne un nombre compris entre 0 et 1.
- Générer un nombre aléatoire entier entre 1 et N

```
function aleatoire(N) {  
    return (Math.floor((N)*Math.random())+1));  
}
```

# LES BASES DU LANGAGE

## const

- La **déclaration const** permet de créer une constante nommée accessible uniquement en lecture.
- Cela ne signifie pas que la valeur contenue est **immutable**, uniquement que l'identifiant ne peut pas être réaffecté.
- Autrement dit la valeur d'une constante ne peut pas être modifiée par des réaffectations ultérieures. Une constante ne peut pas être déclarée à nouveau.
- Cette déclaration permet de créer une constante qui peut être globale ou locale pour la fonction dans laquelle elle a été déclarée. Les constantes font partie de la portée du bloc (comme les variables définies avec let)

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/const>

# LES BASES DU LANGAGE

## **const**

- Il est nécessaire d'initialiser une constante lors de sa déclaration.
- Attention, la déclaration **const** crée une référence en lecture seule vers une valeur.
- Cela ne signifie pas que la valeur référencée ne peut pas être modifiée ! Ainsi, si le contenu de la constante est un objet, l'objet lui-même pourra toujours être modifié.

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/const>

# JAVASCRIPT

## **Tableau** : les tableaux sont des objets

- `var un_tableau=new Array(10)`
- En JavaScript, le premier élément est indexé à 0. Il est possible de déclarer un tableau sans dimension fixée. La taille du tableau s'adapte en fonction du contenu :
- `var un_autre_tableau=new Array;`
- Utilisation :
- Pour accéder aux éléments du tableau, on utilise les crochets "[" et "]"
- `un_tableau[0]=10;`
- `un_tableau[9]=5;`
- `var dimension=un_tableau.length; => méthode`

# LES TABLEAUX

```
let sequence = [1, 1, 2, 3,      • .split(',');
5, 8, 13];                      • .length
                                 • .join(',');
let size = sequence.length;      • .toString();
for (var i = 0; i < size;        • .push('elemt');
i++)                           • .pop();
{                                • .unshift('elemt');
    console.log(sequence[i]);   • .shift();
}

```

# LES TABLEAUX

```
let sequence = [1, 1, 2, 3,  
5, 8, 13];  
  
for (var i = 0; i <  
sequence.length; i++)  
{  
    console.log(sequence[i]);  
}
```

**Exercices, réalisez une fonction pour :**

1. Doubler chaque élément du tableau
2. Renvoyer un tableau avec les éléments paires
3. Renvoyer un tableau avec les éléments impairs
4. Faire la sommes des éléments du tableau
5. Indiquez si un élément est paire

# LES TABLEAUX

```
let sequence = [1, 1, 2, 3, 5,  
8, 13];  
  
for (var i = 0; i <  
sequence.length; i++)  
{  
    console.log(sequence[i]);  
}
```

Doubler chaque élément du tableau

```
let sequence = [1,1,2,3,5,8,13]  
const soustableau = sequence.map(e =>  
e * 2)
```

- soustableau
- Array(7) [ 2, 2, 4, 6, 10, 16, 26 ]

# LES TABLEAUX

```
let sequence = [1, 1, 2, 3, 5, 8, 13];
for (var i = 0; i < sequence.length; i++)
{
    console.log(sequence[i]);
}
```

Renvoyer un tableau avec les éléments paires / impaires

```
const soustableauinaire = sequence.filter(e => e % 2 ==0)
```

- soustableauinaire
- Array [ 2, 8 ]

```
const soustableauimpaire = sequence.filter(e => e !% 2 ==0)
```

# LES TABLEAUX

```
let sequence = [1, 1, 2, 3, 5, 8, 13];
for (var i = 0; i < sequence.length;
i++)
{
    console.log(sequence[i]);
}
```

Faire la somme des éléments du tableau

```
const sommetableau =
sequence.reduce((sum,e)=>sum+e);
```

- sommetableau
- 33

Ici sum est un accumulateur

```
reduce(function (accumulator, currentValue,
currentIndex) { /* ... */ })
```

```
reduce(function (accumulator, currentValue) {
/* ... */, initialValue)
```

# LES TABLEAUX

```
let sequence = [1, 1, 2, 3, 5, 8, 13];
for (var i = 0; i < sequence.length;
i++)
{
    console.log(sequence[i]);
}
```

Indiquez si un élément est paire

```
const isPaire = sequence.filter(e => e % 2 != 0).length != 0;
```

- isPaire
- true

```
const words = ['spray', 'limit', 'elite',
'exuberant', 'destruction', 'present'];
```

```
const result = words.filter(word =>
word.length > 6);
```

```
console.log(result);
```

```
// expected output: Array ["exuberant",
"destruction", "present"]
```

# BILAN

- **map** : mon tableau de sortie est différent mais de même taille
- **filter** : mon tableau de sortie est plus petit, il est filtré
- **reduce** : je renvoie une valeur

# BILAN

## reduce

- `const getMax = (a, b) => Math.max(a, b);`
- `[1, 100].reduce(getMax, 50); // 100`
- `[50].reduce(getMax, 10); // 50`

# EXERCICE

- Sois le tableau :

```
const football = [  
  {club: ''SCB'', joueur: ''SANTELLI'', but:2},  
  {club: ''SCB'', joueur: 'MAGRI', but:1},  
  {club: '' HAC'', joueur: ''KITALA'', but:3},  
  {club: ''SCB'', joueur: 'ROBIC', but:3},  
  {club: ''BORDEAUX'', joueur: ''MAJA'', but:4},  
]
```

**Combien de but ont marqué (somme)  
les joueurs du SCB après avoir ajouté 1  
but à tous les joueurs du tableau !**

# EXERCICE

- Sois le tableau :

```
const football = [  
  {club: "SCB",  
   joueur: "SANTELLI", but: 2},  
  {club: "SCB", joueur: "MAGRI",  
   but: 1}, {club: "HAC",  
   joueur: "KITALA", but: 3},  
  {club: "SCB", joueur: "ROBIC",  
   but: 3}, {club: "BORDEAUX",  
   joueur: "MAJA", but: 4}  
]
```

```
const somme = football  
  .filter(line => line.club === "SCB")  
  .map(line => line.but += 1)  
  .reduce((somme, sumGoal) => somme + sumGoal);
```

# JAVASCRIPT

## Fonction :

- Les fonctions et instructions sont déclarées et codées dans l'entête de la page et peuvent être appelées ensuite à n'importe quel endroit de la page. Pour déclarer une fonction ou une instruction, on utilise le mot-clé "function", suivi de son nom et des éventuels paramètres.

```
function ma_fonction(param1, param2)  
{  
    ....  
}
```

# JAVASCRIPT

- A réaliser en JS
- Puis chercher le pattern
- H, O, S

## T Pattern

```
for row in range(7):
    for col in range(5):
        if (row in {0,6}) or (col == 2):
            print("*", end=" ")
        else:
            print(" ",end=" ")
    print()
```

Output:

```
*****
 *
 *
 *
 *
 *
*****

```



# JAVASCRIPT

A vous :

*Réaliser une page html simple qui utilise cette fonction*

## Fonction :

- Comme en C, les instructions sont regroupées par les accolades { et }. Dans un script, il doit y avoir autant d'accolades ouvertes que d'accolades fermées.

```
function somme(a,b) {  
    var sum=a+b;  
    return sum;  
}
```

# JAVASCRIPT

```
<html>
<head>
<title>Titre</title>
<script type='text/javascript'>
    function somme(a,b)
    {
        var sum=Number(a)+Number(b);
        document.form1.textfield3.value = sum;
        alert(sum);
    }
</script>
```

# JAVASCRIPT

```
</head>
<body>
<p>Une page HTML tout simple</p>
<form name="form1" method="" action="">
    <input type="text" name="textfield1" value=0>
    <input type="text" name="textfield2" value=0>
    <input type="submit" name="Submit" value="Envoyer"
onClick="somme(document.form1.textfield1.value,document.form1.textfield2.value)">
    <input type="text" name="textfield3" value="">
</form>
</body></html>
```

# JAVASCRIPT

## Conditions If :

- L'instruction if permet de diriger l'exécution du script suivant le résultat d'un test. Il y a 2 moyens d'utiliser if :
- Si l'action à réaliser tient en une instruction :
- `if (age<18) alert("Vous devez être majeur");`
- Ici, si l'âge du visiteur est inférieur à 18 ans, un message est affiché.
- Si l'action à réaliser tient en plusieurs instructions :

```
if (age<18) {  
  
    alert("Vous devez être majeur");  
  
    window.location="mineur.php3";}
```

- Ici, si l'âge du visiteur est inférieur à 18 ans, un message est affiché et le visiteur est redirigé vers la page mineur.php3.

# JAVASCRIPT

## Conditions If - Else:

- Reprenons notre dernier exemple :

```
if (age<18) {  
  
    alert("Vous devez être majeur");  
  
    window.location="mineur.php3";  
  
} else {  
  
    window.location="majeur.php3";  
  
}
```

- Ici, si le visiteur est mineur, il est redirigé vers mineur.php3, sinon il est redirigé vers la page majeur.php3..

# JAVASCRIPT

## Boucles : For

- Une boucle itérative exécute un groupe d'instructions tant que le compteur d'itérations n'a pas atteint une valeur donnée. Voici la syntaxe générale :

```
for(initialisation;condition;opération) {  
    // Vos instructions }
```

- Boucle pour i variant de 0 à 100 inclus par pas de 1

```
for (var i=0;i<=100;i=i+1)
```

- Boucle pour i variant de 10 à 0 inclus par pas de -1

```
for (var i=10;i>0;i=i-1)
```

- A vous : sur le modèle du dernier exemple faites une autre fonction qui calcule la somme des nombres de 1 à N, N étant entré par l'utilisateur.

# JAVASCRIPT

```
<html>
<head>
<title>Somme N</title>
<script type='text/javascript'>
function somme(N) {
    var sum=0;
    for (var i=1;i<=N;i=i+1) {
        sum=sum+i;
    }
    alert("Somme de 1 à "+N+" = "+sum);
}
```

# JAVASCRIPT

```
        } </script>
    </head>
<body>
    <p>Une page HTML tout simple</p>
    <form name="f" method="" action="">
        <input type="text" name="v" value=0>
        <input type="submit" name="Submit" value="Envoyer"
            onclick="somme(document.f.v.value);">
    </form>
</body>
</html>
```

# JAVASCRIPT

## Boucles While :

- Les boucles conditionnelles. Il peut être utile de faire une boucle tant qu'une condition est vraie.

- La boucle "tant que" est déclarée par while :

```
while (condition) {  
    // Les instructions de la boucle  
}
```

- Ici, tant que condition est vraie, la boucle est exécutée.

# JAVASCRIPT



## Conversion :

- En JavaScript, les variables ne sont pas typées, mais il est utile de savoir transformer une chaîne en un entier ou un réel (nombre à virgule).
- Imaginons ce script :

```
var chaine="3.14";  
  
var entier=parseInt(chaine);  
  
var reel=parseFloat(chaine);
```
- A l'issue de ce script, entier contient 3 et reel contient 3.14

# LE MODÈLE OBJET

BISGAMBIGLIA@UNIV-CORSE.FR

115

# JAVASCRIPT



## L'objet *this*

- Il est fastidieux d'accéder aux éléments de formulaire par toute la chaîne `document.forms.elements`. Un objet JavaScript `this` permet de raccourcir ce chemin d'accès. `this` représente l'objet javascript en cours.

<FORM>

```
<INPUT type="text" name="zonedetexte" value="Valeur initiale">
```

```
<INPUT type="button" value="Changer le contenu"
```

```
onClick='this.form.zonedetexte.value="NOUVEAU" '>
```

</FORM>

- Grâce à `this.form`, on peut accéder au formulaire de l'élément en cours. Le chemin pour accéder à `zonedetexte` est ensuite classique.

# JAVASCRIPT

```
<script type="text/javascript">
function muestraMas(){
    var var1=document.getElementById('masElementos');
    var var2=document.getElementById('masElementos');
    var var3=document.getElementById('comactual');
    var var4=document.getElementById('numpoliza');

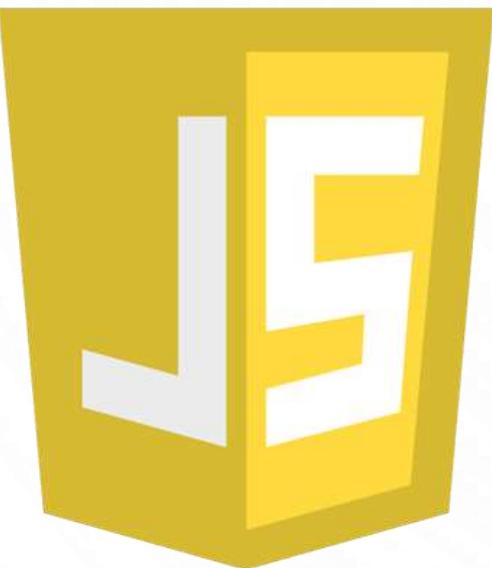
    if(var1=="SI"){
        var2.style.display="block";
        //document.getElementById('comactual').className="documentacion";
        document.getElementById('comactual').className="requiere";
    }
    else{
        var2.style.display="none";
    }
}

```

# JAVASCRIPT

- Javascript est basé objet (se n'est pas un langage objet !)
- On utilise la notation pointée pour se référer à des méthodes et des attributs d'un objet:  
objet.nom\_méthode(argument)
- Exemple : objRequete.open('get','lander.php?ou=' + i,true);
- Création explicite avec constructeur:
- var [objet] = new [déclaration d'objet]
- ([paramètres optionnels])
- ex. : objRequete = new ActiveXObject("MSXML2.XMLHTTP");
- objRequete = new XMLHttpRequest();
- objDate = new Date();

# JAVASCRIPT

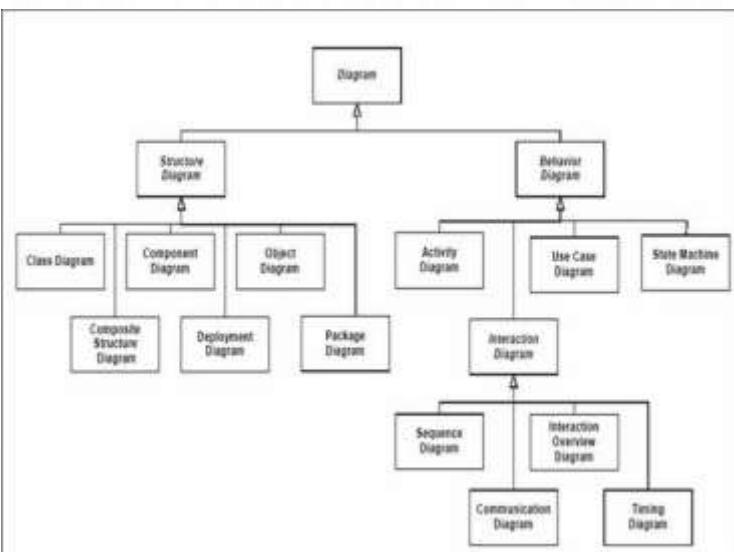


## Mot clé **this**

- Il permet de désigner l'objet courant.
- Lorsque vous déclarez une variable de la forme `this.[variable]` dans une méthode constructeur, cette notation représente un accès à une propriété de l'objet créé par la méthode :

```
function creeXMLHttpRequestObjet() {  
    this.objRequete = new  
    ActiveXObject("Microsoft.XMLHTTP");  
}
```

# JAVASCRIPT



## Création d'objet

- L'exemple suivant inclut dans la déclaration d'objet
- CreeObjetAjax() une référence à la fonction
- creeXMLHttpRequestObject() au moyen de this :

```
function CreeObjetAjax(){  
    this.creeXMLHttpRequestObject=  
        creeXMLHttpRequestObect;  
}
```

- Création d'un objet à partir d'une déclaration d'objet personnalisé et appel d'une méthode:

```
o = new CreeObjetAjax();  
objRequete = o.creeXMLHttpRequestObject();
```

# JAVASCRIPT



- Fonction universelle de création d'un objet asynchrone : **XMLHttpRequest**
- La gestion des exceptions est essentiel pour pouvoir écrire une fonction de création d'un objet XMLHttpRequest universelle est portable.
- La logique est la suivante:
- Soit la création de l'objet XMLHttpRequest réussit
- Soit elle échoue (navigateur ou analyseur XML incompatible)
- Auquel cas cela produit une exception que nous interceptions.

# LES CLASSES

```
class Rectangle {  
    constructor(hauteur,  
    largeur) {  
        this.hauteur = hauteur;  
        this.largeur = largeur;  
    }  
}
```

- <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes>  
// anonyme  

```
let Rectangle = class { constructor(hauteur,  
largeur) { this.hauteur = hauteur; this.largeur =  
largeur; } };
```

  
// nommée  

```
let Rectangle = class Rectangle {  
    constructor(hauteur, largeur) { this.hauteur =  
hauteur; this.largeur = largeur; } };
```

# LES CLASSES

```
class Point {  
    constructor(x, y) {  
        this.x = x; this.y = y;  
    }  
    static distance(a, b) {  
        const dx = a.x - b.x;  
        const dy = a.y - b.y;  
        return Math.hypot(dx, dy);  
    }  
}  
const p1 = new Point(5, 5);  
const p2 = new Point(10, 10);  
console.log(Point.distance(p1, p2));
```

# LES CLASSES

```
class Rectangle {  
    #hauteur = 0;  
    #largeur;  
    constructor(hauteur, largeur) {  
        this.#hauteur = hauteur;  
        this.#largeur = largeur;  
    }  
}  
}
```

# LES CLASSES

```
class Animal {  
    constructor(nom) {  
        this.nom = nom;  
    }  
  
    parle() {  
        console.log(`${this.nom} fait du bruit.`);  
    }  
}  
  
class Chien extends Animal {  
    constructor(nom) {  
        super(nom); // appelle le constructeur parent avec le paramètre  
    }  
    parle() {  
        console.log(`${this.nom} aboie.`);  
    }  
}
```

# JAVASCRIPT

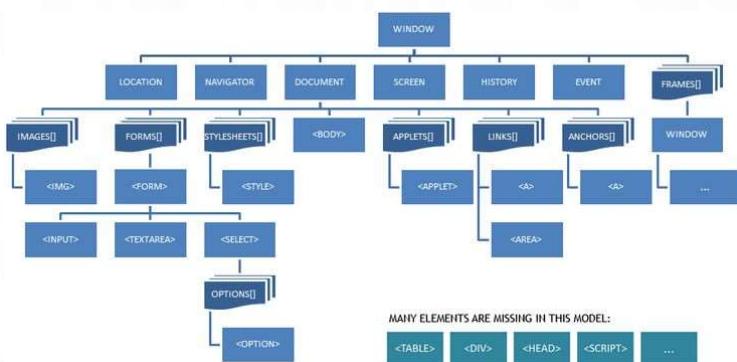
# DOM

- La plupart des objets que vous employer forme une bibliothèque d'objets n'appartenant pas à JavaScript mais au modèle DOM.
- Lorsqu'un navigateur compatible avec DOM charge une page, il indexe tous les éléments connus et identifiables d'après leur type, leurs propriétés, et leur position dans le document.
- Ces éléments (ou objet) sont ensuite accessibles à des scripts par le biais de tableaux spéciaux appelés collections ou par leur nom s'ils possède un attribut name.
- Exemple : Accès au troisième formulaires d'une page Web: `window.document.forms[2]` ;

# JAVASCRIPT

Quelques objets disponibles dans JavaScript :

- document: représente une page Web.
- event: généré par des événements cotés client.
- form: contient une référence à un objet de type formulaire.
- frame: contient une référence à une frame.
- history: contient des informations sur les URL visitées.
- image: contient une référence à une image.
- link: contient une référence à un hyperlien.
- location: permet d'accéder à la barre d'adresse.
- window: contient les informations d'état de la fenêtre.
- navigator: contient des informations sur le navigateur.



# JAVASCRIPT

# DOM

Collections disponibles dans JavaScript

- anchors: contient des références aux ancrés d'une page.
- applets: contient des références aux applets Java.
- elements: contient des références aux éléments d'un formulaire:
- Ces éléments sont représenté par:
- Button, CheckBox, FileUpload, Hidden, Password, Radio, Reset, Select, Submit, Text, Textarea.
- forms: contient des références aux formulaires.
- images: contient des références aux images.
- options: contient des références aux options d'un élément de formulaire de type Select.
- plugins: contient des références aux plugins installés.

# JAVASCRIPT

- Pour accéder à un objet, il faudra donner le chemin complet de l'objet (du plus grand au plus petit)
- Exemple : `(window).document.form.radio[0]`
- Ils ont ensuite des propriétés personnalisées. Pour accéder aux propriétés on utilise :
- `nom_de_l'objet.nom_de_la_propriété`

# JAVASCRIPT

- Supposons le formulaire suivant :

```
<FORM name="general"><INPUT type="text" name="champ1" value="Valeur  
initiale"></FORM>
```

- Le formulaire est un élément de l'objet document. Pour accéder au formulaire **general**, il faut écrire :

```
document.forms["general"] ou document.forms[0] ou document.general
```

- Pour accéder maintenant à la zone de texte, on écrit :

```
document.forms["general"].elements["champ1"]
```

```
ou document.forms["general"].elements[0]
```

```
ou document.forms["general"].champ1
```

# JAVASCRIPT

- Manipuler les propriétés d'un élément
  - Une fois que l'élément est atteint, il est possible de manipuler ces propriétés.
  - Par exemple, pour placer dans la zone de texte le mot "NOUVEAU", il faut juste écrire :  
`document.forms["general"].elements["champ1"].value="NOUVEAU"`
- Appeler une méthode sur un élément
  - Pour donner le focus au champ texte du haut de cette page, il faut appeler la méthode `focus()` sur cet élément.  
`document.forms["general"].elements["champ1"].focus()`

# JAVASCRIPT

- Selecteur

- `element = document.querySelector(selecteurs);`
  - Avec selecteurs = "#nomID" l'id d'une balise HTML nommé alors nomID
  - Avec selecteur = ".maclasse" la classe d'une balise HTML
- `const matches = document.querySelectorAll("p");`

<https://developer.mozilla.org/fr/docs/Web/API/Document/querySelector>

# JAVASCRIPT



- Invocation explicite d'un gestionnaire d'événements
- L'objet event
- L'objet event contient des informations sur un événement qui s'est produit quelque part dans la page.
- Les conséquences de ce mécanisme sont les suivantes:
- La surveillance d'événement est programmée directement dans un script (lien avec HTML rompu).
- Il est possible d'implémenter une gestion d'événement globale.
- Exemple: Un utilisateur clique dans une zone de la page. L'objet event produit alors les informations:
  - La touche de la souris activée;
  - La touche éventuellement pressée: Ctrl, Alt, Alt Gr, Maj
  - Les coordonnées du pointeur de la souris

# JAVASCRIPT



- **EventTarget.addEventListener()**

```
addEventListener(type, listener)
```

```
addEventListener(type, listener, options)
```

```
addEventListener(type, listener, useCapture)
```

```
// Function to change the content of t2 function
```

```
modifyText() { const t2 =  
document.getElementById("t2"); const  
isNodeThree = t2.firstChild.nodeValue ===  
"three"; t2.firstChild.nodeValue = isNodeThree ?  
"two" : "three"; } // Add event listener to table  
const el = document.getElementById("outside");  
el.addEventListener("click", modifyText, false);
```

# JAVASCRIPT



- Invocation explicite d'un gestionnaire d'événements
- Réaction à un objet event
- Lorsqu'un objet event est créé, il est répercuté tout au long de la hiérarchie d'objets en amont du document.
- Exemple : Si vous cliquez sur "Cliquez": une fenêtre 'span' s'affiche, puis une fenêtre 'h1'... jusqu'à la fenêtre 'body'.

```
<html><body onclick="alert('body')">
  <div onclick="alert('div')">
    <h1 onclick="alert('h1')">
      <span onclick="alert('span')">Cliquez</span>
    </h1></div>
  </body></html>
```
- Si vous cliquez dans une autre zone de la page, vous ne recevez que la notification pour body

# JAVASCRIPT



- Le chargement (`onLoad`) et l'abandon (`OnUnLoad`);
- La soumission (`onSubmit`) et la réinitialisation (`onReset`) des données d'un formulaire;
- Le survol d'une zone d'une page avec le curseur (`onMouseOver` à l'arrivée dans la zone et `onMouseOut` en quittant la zone) ou plus généralement le simple déplacement de la souris, que l'utilisateur clique ou non (`onMouseMove`);
- Le clic de l'utilisateur sur une référence, un bouton, ou un autre élément (`onClick`);
- Le changement (`onChange`), l'activation (`onFocus`), ou l'abandon (`onBlur`) d'un élément.

# JAVASCRIPT



- Les gestionnaires d'événements relatifs au clavier sont utilisés en liaison avec le champs de saisie libre d'un formulaire:
- L'activation complète d'une touche (`onKeyPress`)
- La pression d'une touche (`onKeyDown`)
- Le relâchement d'une touche (`onKeyUp`)
- Pour réagir à la sélection du texte (`onSelect`)
- Pour réagir à l'interruption du chargement d'une image (`onAbort` et `onError`)

# JAVASCRIPT



RÉCIT préscolaire

Pour appeler une fonction ou méthode Javascript à partir d'une page Web, vous pouvez utiliser:

- un gestionnaire d'événements,
- Un gestionnaire d'événements propre à JavaScript,
- Un Script.
- Une fonction ou procédure n'est pas exécutée lors de son chargement mais uniquement lorsqu'elle est appelée.
- Fonctions personnalisées

```
function [nom] ([liste de paramètres]){
    ... instructions
}
```

# JAVASCRIPT



## Exception sous forme d'objets

- Une exception se manifeste dans un script sous la forme d'une erreur d'exécution pouvant être décrite de manière assez standardisée.
- ECMAScript Edition 3 (i.e JavaScript 1.5), dispose d'un objet erreur nommée Error.
- Des instances de ces objets sont créées en tant qu'exceptions lorsqu'une erreur connue survient:
  - EvalError,
  - RangeError,
  - ReferenceError,
  - TypeError,
  - SyntaxError....

# JAVASCRIPT



## JavaScript

- Une instruction JavaScript peut occuper plusieurs lignes dans l'éditeur et se termine toujours par un point-virgule.
- Sortes d'instructions:
- Les blocs sont délimité par les accolades et contiennent plusieurs instructions
- Les expressions: `poids=98+3;`
- Les commentaires possède la même syntaxe qu'en C.
- Les instructions conditionnelles: `if`, `if-else` et `switch-case`.
- Les boucles: `while`, `do-while` et `for`
- Les instructions de saut: `break` et `continue`

# FONCTIONS JAVASCRIPT DE CODAGE DE CARACTÈRES



- **charCodeAt**

- var code = machaine.charCodeAt(5);

- **fromCharCode**

- var ch = String.fromCharCode(125);

- var machaine ="azertyuiop";

- var code = machaine.charCodeAt(5);

- var ch = String.fromCharCode(121);

A VOUS

- Coder ce ‘mini’ jeu en JS



```

<!doctype html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <main>
        <input type='number' placeholder='value' name='code'
id='code'>
        <input type='submit' onclick="myTest()">
        <div id='laReponse'></div>
        <details>
            <summary> Indice 1</summary>
            <p>682 un des numéros est correct et bien placé</p>
        </details>
        <details>
            <summary> Indice 2</summary>
            <p>614 un des numéros est correct mais mal placé</p>
        </details>
        <details>
            <summary> Indice 3</summary>
            <p>206 deux numéros sont corrects mais mal placés</p>
        </details>
        <details>
            <summary> Indice 4</summary>
            <p>738 aucun des numéros n'est</p>
        </details>
        <details>
            <summary> Indice 5</summary>
            <p>780 un des numéros est correct mais mal placé</p>
        </details>
    </main>
</body>
<script>
    var solution = String.fromCharCode(48, 52, 50);
    function myTest()
    {
        var message = "ce n'est pas la bonne solution";
        var reponse =
document.getElementById('code').value;
        if (reponse === solution)
        {
            message = "bravo";
        }
        document.getElementById('laReponse').innerHTML =
message;
    }
</script>
</html>

```

# FONCTIONS FLÉCHÉES

Deux facteurs sont à l'origine de la conception des fonctions fléchées : une syntaxe plus courte et l'absence de this spécifique à la fonction. Sur ce dernier point, cela signifie qu'une fonction fléchée ne lie pas son propre [this](#) au sein de la fonction

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Functions/Arrow\\_functions](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Functions/Arrow_functions)

```
var a = [
  "We're up all night 'til the sun",
  "We're up all night to get some",
  "We're up all night for good fun",
  "We're up all night to get lucky",
];
// Sans la syntaxe des fonctions fléchées
var a2 = a.map(function (s) {
  return s.length;
});
// [31, 30, 31, 31]
// Avec, on a quelque chose de plus concis
var a3 = a.map((s) => s.length);
// [31, 30, 31, 31]
```

# CALLBACK

Une fonction de rappel (aussi appelée *callback* en anglais) est une fonction passée dans une autre fonction en tant qu'argument, qui est ensuite invoquée à l'intérieur de la fonction externe pour accomplir une sorte de routine ou d'action.

[https://developer.mozilla.org/fr/docs/Glossary/Callback\\_function](https://developer.mozilla.org/fr/docs/Glossary/Callback_function)

```
function salutation(name) {  
    alert("Bonjour " + name);  
}
```

```
function processUserInput(callback) {  
    var name = prompt("Entrez votre nom.");  
    callback(name);  
}
```

```
processUserInput(salutation);
```

# PROMISE

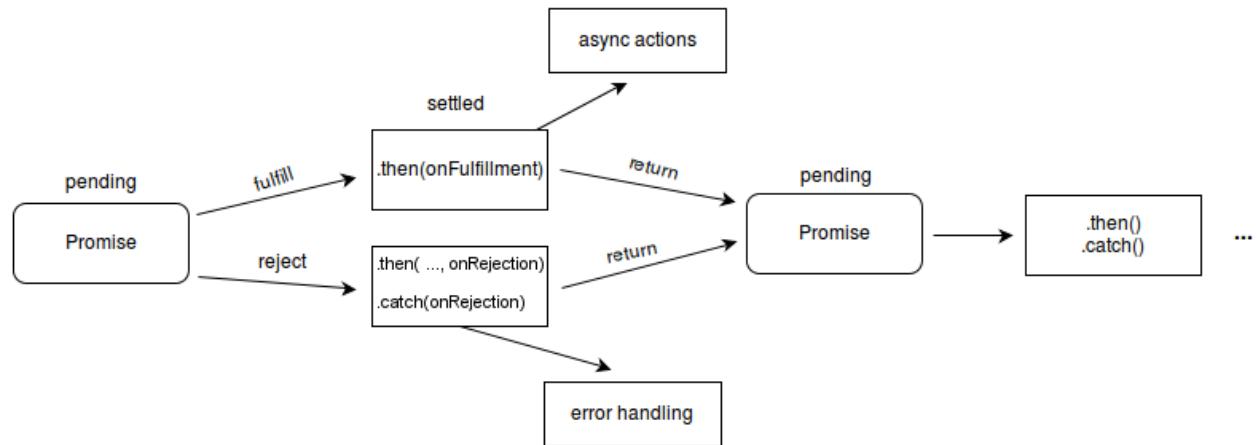
L'objet **Promise** (pour « promesse ») est utilisé pour réaliser des traitements de façon asynchrone. Une promesse représente une valeur qui peut être disponible maintenant, dans le futur voire jamais.

L'interface **Promise** représente un intermédiaire (proxy) vers une valeur qui n'est pas nécessairement connue au moment de la création de la promesse.

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Promise)

- Une Promise est dans un de ces états :

- *pending* (*en attente*) : état initial, la promesse n'est ni tenue, ni rompue ;
- *fulfilled* (*tenue*) : l'opération a réussi ;
- *rejected* (*rompue*) : l'opération a échoué.



# PROMISE

L'objet **Promise** (pour « promesse ») est utilisé pour réaliser des traitements de façon asynchrone. Une promesse représente une valeur qui peut être disponible maintenant, dans le futur voire jamais.

L'interface **Promise** représente un intermédiaire (proxy) vers une valeur qui n'est pas nécessairement connue au moment de la création de la promesse.

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Promise)

```
const maPromesse = new Promise((resolve, reject)  
=> {  
    setTimeout(() => {  
        resolve("toto");  
    }, 300);  
});
```

**maPromesse**

```
.then(gestionnaireSuccesA, gestionnaireEchecA)  
.then(gestionnaireSuccesB, gestionnaireEchecB)  
.then(gestionnaireSuccesC, gestionnaireEchecC);
```

# ASYNC

Une fonction asynchrone est une fonction précédée par le mot-clé `async`, et qui peut contenir le mot-clé `await`. `async` et `await` permettent un comportement asynchrone, basé sur une promesse ([Promise](#)), écrite de façon simple, et évitant de configurer explicitement les chaînes de promesse.

[https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/async\\_function](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/async_function)

```
function resolveAfter2Seconds() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000);
  });
}

async function asyncCall() {
  console.log('calling');
  const result = await resolveAfter2Seconds();
  console.log(result);
  // Expected output: "resolved"
}

asyncCall();
```

# API FETCH

L'API Fetch fournit une interface pour la récupération de ressources (e.g., à travers le réseau.) Elle paraîtra familière à tout utilisateur de [XMLHttpRequest](#), mais cette nouvelle API propose néanmoins un ensemble de fonctionnalités plus souples et plus puissantes.

Fetch fournit une définition générique des objets [Request](#) et [Response](#) (et d'autres choses impliquées par les requêtes réseau).

[https://developer.mozilla.org/fr/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/fr/docs/Web/API/Fetch_API)

```
// Exemple d'implémentation pour une requête POST
async function postData(url = "", données = {}) {
    // Les options par défaut sont indiquées par *
    const response = await fetch(url, {
        method: "POST", // *GET, POST, PUT, DELETE, etc.
        mode: "cors", // no-cors, *cors, same-origin
        cache: "no-cache", // *default, no-cache, reload, force-cache, only-if-cached
        credentials: "same-origin", // include, *same-origin, omit
        headers: {
            "Content-Type": "application/json",
            // 'Content-Type': 'application/x-www-form-urlencoded',
        },
        redirect: "follow", // manual, *follow, error
        referrerPolicy: "no-referrer", // no-referrer, *no-referrer-when-downgrade,
        origin, origin-when-cross-origin, same-origin, strict-origin, strict-origin-when-cross-origin, unsafe-url
        body: JSON.stringify(données), // le type utilisé pour le corps doit correspondre
        à l'en-tête "Content-Type"
    });
    return response.json(); // transforme la réponse JSON reçue en objet JavaScript
    natif
}

postData("https://example.com/solution", { solution: 42 }).then((données) => {
    console.log(données); // Les données JSON analysées par l'appel `donnees.json()`
});
```

# ECMASCRIPT 6 : LE JAVASCRIPT DE DEMAIN

<https://www.wanadev.fr/21-introduction-a-ecmascript-6-le-javascript-de-demain/>

- Le mot clé let permet de déclarer une variable limitée à la portée d'un bloc
- Template String
  - let me = "Yannick"; let mAge = 29; let result = `Je suis \${me} et j'ai \${mAge} ans`;
- **Boucle for - of**
  - for (let language of languages)
- **Classe et héritage**
  - class Person { constructor(name, age) { this.name = name; this.age = age; } toString() { return `Hi I'm \${this.name} and I'm \${this.age} years old!`; } }
  - class Developer extends Person { constructor(name, age, language) { super(name, age); this.language = language; } toString() { return super.toString() + ` :: I'm a Developer who likes \${this.language}`; } }

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes/constructor>

# ECMASCRIPT 6 - ECMASCIPT 2015

- [https://www.w3schools.com/js/js\\_es6.asp](https://www.w3schools.com/js/js_es6.asp)
- <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes>
- News :
  - JavaScript let
  - JavaScript const
  - JavaScript Arrow Functions
  - JavaScript Classes
  - Default parameter
  - values Array.find()
  - Array.findIndex()
  - Exponentiation (\*\*)

# ECMASCRIPT 6 - ECMASCIPT 2015

- [https://www.w3schools.com/js/js\\_es6.asp](https://www.w3schools.com/js/js_es6.asp)
- <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes>

```
class Rectangle {  
    constructor(hauteur, largeur) {  
        this.hauteur = hauteur;  
        this.largeur = largeur;  
    }  
    get area() {  
        return this.calcArea();  
    }  
    calcArea() {  
        return this.largeur * this.hauteur;  
    }  
}
```

```
const carré = new Rectangle(10, 10);  
  
console.log(carré.area);
```

# ECMASCRIPT 6 - ECMASCIPT 2015

- [https://www.w3schools.com/js/js\\_es6.asp](https://www.w3schools.com/js/js_es6.asp)
- <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes>

```
class Point {  
    constructor(x, y) {  
        this.x = x;  
        this.y = y;  
    }  
    static distance(a, b) {  
        const dx = a.x - b.x;  
        const dy = a.y - b.y;  
        return Math.hypot(dx, dy);  
    }  
}  
  
const p1 = new Point(5, 5);  
const p2 = new Point(10, 10);  
console.log(Point.distance(p1, p2));
```

# ECMASCRIPT 6 - ECMASCIPT 2015

- [https://www.w3schools.com/js/js\\_es6.asp](https://www.w3schools.com/js/js_es6.asp)
- <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes>

```
class Chat {  
    constructor(nom) {  
        this.nom = nom;  
    }  
  
    parler() {  
        console.log(` ${this.nom} fait du bruit.`);  
    }  
}  
  
class Lion extends Chat {  
    parler() {  
        super.parler();  
        console.log(` ${this.nom} rugit.`);  
    }  
}
```

# JSX

- <https://fr.reactjs.org/docs/introducing-jsx.html>

```
const element =  
<h1>Bonjour, monde  
!</h1>;
```

```
function formatName(user) {  
  return user.firstName + ' ' + user.lastName;  
}  
  
const user = {  
  firstName: 'Kylian',  
  lastName: 'Mbappé'  
};
```

```
const element = (  
  <h1>  
    Bonjour, {formatName(user)} !  
  </h1>  
)
```

```
ReactDOM.render(  
  element,  
  document.getElementById('root')  
)
```

# TYPE SCRIPT

[HTTPS://WWW.TYPESCRIPTLANG.ORG/](https://www.typescriptlang.org/)

[HTTPS://LEARNXINYMINUTES.COM/DOCS/FR-FR/TYPESCRIPT-FR/](https://learnxinyminutes.com/docs/fr-fr/typescript-fr/)

# TYPE SCRIPT

TypeScript

- Installation
  - `npm install -g typescript`
- **Compilation tsc monfichier.ts**
  - `> monfichier.js`

# TYPE SCRIPT / EXEMPLE

```
class Student { fullName: string; constructor(public firstName: string, public middleInitial: string, public lastName: string) { this.fullName = firstName + " " + middleInitial + " " + lastName; } }

interface Person { firstName: string; lastName: string; }

function greeter(person : Person) { return "Hello, " + person.firstName + " " + person.lastName; }

let user = new Student("Jane", "M.", "User");

document.body.innerHTML = greeter(user);
```

# TYPE SCRIPT / EXEMPLE

```
<!DOCTYPE html>
<html>
<head>
<title>TypeScript Greeter</title>
</head>
<body>
<script src="greeter.js"></script>
</body>
</html>
```

# TYPE SCRIPT

- Type
- <https://www.typescriptlang.org/docs/handbook/basic-types.html>

- let isDone: boolean = false;
- let decimal: number = 6;
- let hex: number = 0xf00d;
- let binary: number = 0b1010;
- let octal: number = 0o744;
- let color: string = "blue";
- let list: number[] = [1, 2, 3];
- let list: Array<number> = [1, 2, 3];
- let x: [string, number];
- enum Color {Red, Green, Blue}
- let c: Color = Color.Green;
- let notSure: any = 4;

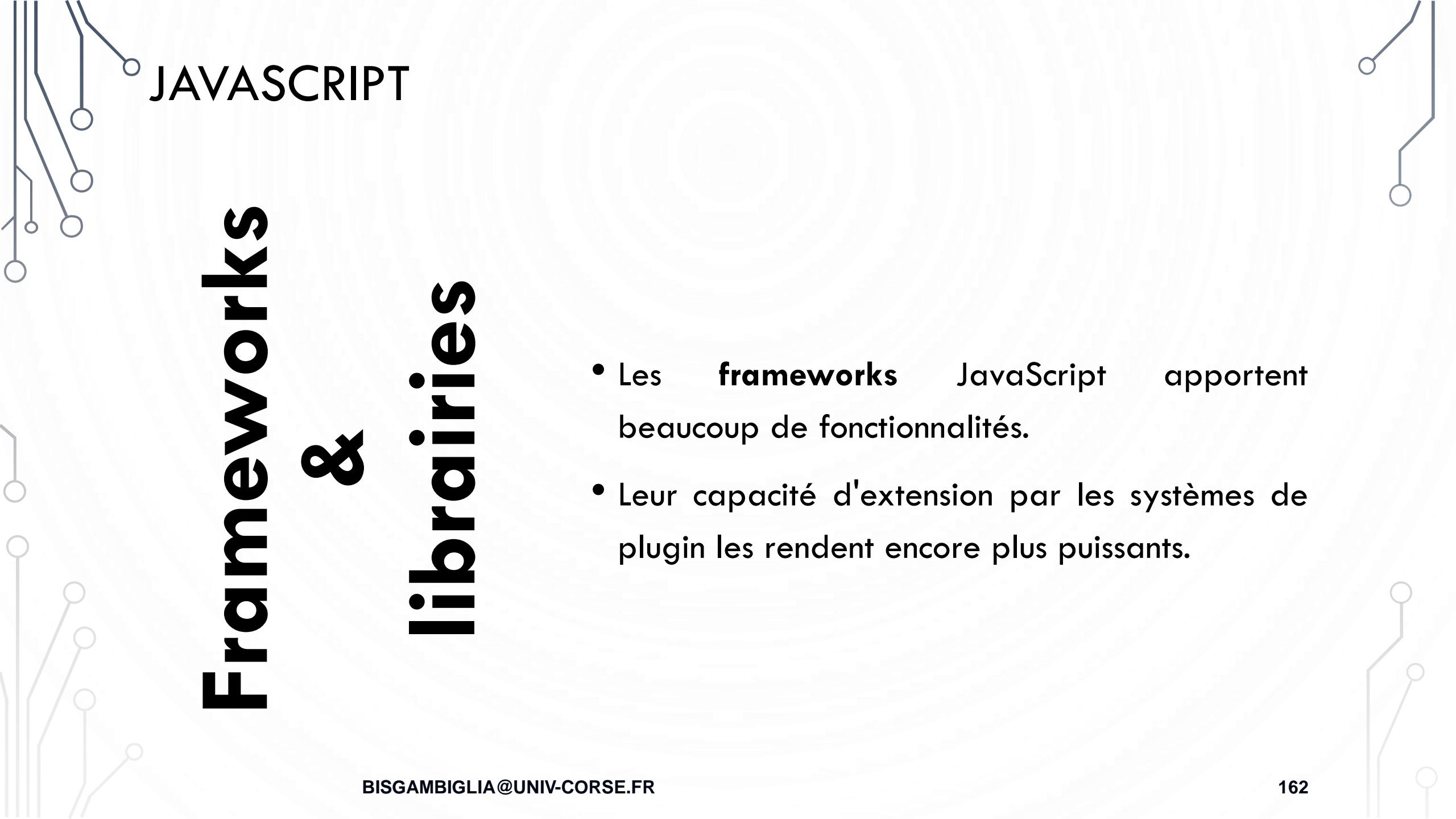
# TYPE SCRIPT

- class
- <https://www.typescriptlang.org/docs/handbook/classes.html>

```
class Animal {  
    move(distanceInMeters: number = 0)  
{ console.log(`Animal moved  
${distanceInMeters}m.`); }  
}
```

```
class Dog extends Animal {  
    bark() { console.log('Woof!  
Woof!'); } }
```

```
const dog = new Dog();  
dog.bark();  
dog.move(10);  
dog.bark();
```



# JAVASCRIPT

## Frameworks & libraries

- Les **frameworks** JavaScript apportent beaucoup de fonctionnalités.
- Leur capacité d'extension par les systèmes de plugin les rendent encore plus puissants.

# JAVASCRIPT



## Animation

- **JSTweener** : JSTweener est une librairie de gestion des transitions (tween en anglais) basé sur la classe Tweener utilisée dans le code ActionScript de Flash.
- **\$fx()** – JavaScript Animation Library : \$fx() est une librairie JavaScript légère (moins de 4Ko) d'animation d'éléments HTML. Elle vous permet de modifier n'importe qu'elle propriété CSS progressivement avec un paramétrage simple. Vous pouvez aussi combiner les effets en les enchainant ou en les synchronisant. Enfin, de nombreux callbacks vous offrent beaucoup de liberté dans la gestion de vos effets.
- **Facebook Animation** : Cette librairie vous offre beaucoup de possibilités pour améliorer leur page Facebook avec juste une ou deux lignes de code. Toutes les animations sont basées sur les CSS, donc une bonne connaissance de CSS est utile. A noter qu'il existe une version open-source fonctionnant en dehors de Facebook.

# JAVASCRIPT



## Audio / Vidéo

- SoundManager : SoundManager importe et améliore l'API Sound de Flash et la rend disponible en JavaScript. La portion Flash est cachée et donc invisible, que ce soit pour le développeur ou l'utilisateur.
- Flowplayer JavaScript API : L'API JavaScript de Flowplayer vous permet de contrôler facilement et efficacement une ou plusieurs instances de Flowplayer dans une page HTML. Flowplayer se compose de deux parties : l'objet SWF Flowplayer, inclue dans un objet Flash et une librairie JavaScript qui transforme les instructions simples de l'API en interaction plus complexe avec l'objet SWF (qui contrôle l'objet Flash).



## Cookies

- **Cookies** : Cette librairie vous permet de récupérer et de manipuler les cookies HTTP dans le navigateur. Vous pouvez récupérer un cookie ou une liste, en créer de nouveaux, en supprimer, tester si le navigateur les accepte... Le framework jQuery n'est pas nécessaire pour utiliser la librairie, mais sa présence ajoute des fonctionnalités, comme créer un cookie à chaque remplissage d'un champ de formulaire ou le remplissage automatique d'un formulaire avec les valeurs des cookies.
- **EasyCookie** : Un script simple et facile d'utilisation permettant la gestion des cookies.

# JAVASCRIPT



## Cryptographie

- JavaScript MD5 : Une implémentation en JavaScript de l'algorithme MD5.

## Bases de données

- Taffy DB : Taffy DB est une librairie AJAX libre. Compatible avec les principales librairies AJAX, ses principales fonctionnalités sont : une interface CRUD (Create, Read, Update, Delete), le tri, les boucles, la gestion de requêtes avancées, etc.
- ActiveRecord.js : ActiveRecord.js est un outil de mapping objet-relationnel compatible avec tous les navigateurs. Son vocabulaire est proche de celle d'ActiveRecord pour Ruby, mais en utilisant la syntaxe et les bonnes pratiques JavaScript. Il peut être utilisé à partir d'un mappage chargé en mémoire ou avec un environnement SQL sur une plateforme Jaxer (SQLite et MySQL), AIR d'Adobe (SQLite) ou Google Gears (SQLite).

# JAVASCRIPT



## Date / Heure

- Date.js : Cette librairie open-source vous permet de manipuler les dates et les heures facilement. Elle supporte plus de 150 formats et propose entre autres de fixer une date, de la parser, de la modifier, de faire des comparaisons, etc.

## Débogage / Traçage

- Firebug Lite : Firebug est probablement l'extension de débogage la plus populaire de Firefox. Firebug Lite est l'alternative pour tester ses pages sur Internet Explorer, Opera et Safari. Il s'agit d'un script JavaScript à inclure pour obtenir certaines fonctionnalités de Firebug.
- Blackbird : Utilitaire de traçage open-source. Il vous permet, à travers une console assez réussie, d'afficher différents messages de débogage.
- NitobiBug : NitobiBug est un outil d'accès aux objets JavaScript basé sur le navigateur (similaire à Firebug). Il peut être utilisé dans différents navigateurs (IE6+, Safari, Opera, Firefox) et offre un outil stable et puissant pour développer des applications AJAX riches.

# JAVASCRIPT



## Police / Texte / Typographie

- **strokeText.js** : strokeText.js est une librairie non intrusive fonctionnant avec les principaux navigateurs (Firefox 1.5 +, IE6 +, Opera 9 +, et Safari). La librairie propose une API d'écriture de texte pour Canvas et VML. La police intégrée sans serif est adaptée au SVG pour un rendu identique.
- **typeface.js** : Cette librairie vous permet d'intégrer vos polices afin d'éviter d'avoir à les afficher sous forme d'images. Au lieu d'utiliser une image ou du Flash pour afficher vos polices, utilisez typeface.js et écrivez votre code HTML / CSS comme si vos visiteurs avaient la police installée.
- **Cufón** : Remplacement rapide de texte à l'aide de Canvas et VML. Pas besoin de Flash ni d'images.
- **Hyphenator.js** : Coupe automatiquement les mots en fin de ligne sur votre site, ou sur tous les sites si vous l'utilisez en plugin.
- Autres librairies : sIFR, Facelift Image Replacement, FontJazz.

# JAVASCRIPT



## Validation de formulaire

- **LiveValidation** : Cette librairie open-source légère vous permet d'effectuer des validations de champs de formulaire facilement et efficacement. Deux versions sont disponibles : une basée sur le framework Prototype et une autonome.
- **wForms** : Librairie non intrusive qui offre toutes les validations usuelles de formulaires sans avoir besoin de connaissances en programmation.
- **Validanguage** : Validanguage est une librairie open-source, non intrusive à gestion d'héritage, développée pour devenir le framework de validation de formulaires de référence.
- **Autres librairies** : Yav, qForms JavaScript API.

# JAVASCRIPT



D'autres exemples à cette adresse :

- <http://javascript.developpez.com/cours/librairies-javascript-vraiment-utiles/>
- Graphiques / Diagrammes
- Tables HTML
- Gestion des images / Visualisation / Dessin
- Clavier
- Cartes
- Maths
- Expressions régulières
- URL

# JAVASCRIPT



## LIBRAIRIES JS

Nom	Tout le web	Top 10k	Top 100k	Top Million
jQuery	34,1%	12,4%	17,1%	20,1%
Google Libraries	6,8%	6,9%	7,8%	5,2%
html5shiv	6,2%	5,9%	6,4%	5,6%
jQuery UI	4,9%	5,1%	6,0%	6,7%
Facebook for Websites	4,6%	7,3%	7,3%	7,0%
Modernizr	3,6%	5,5%	5,4%	3,9%
jQuery Form	3,5%	1,6%	2,0%	2,7%
Facebook SDK	3,3%	6,1%	6,0%	5,7%
jQuery Easing	3,2%	1,5%	2,0%	2,5%
Fancybox	3,0%	2,0%	2,3%	3,0%
jQuery Cycle	2,3%	1,6%	2,1%	2,4%
Lightbox	2,1%	1,3%	1,7%	2,5%

AngularJS  
Sujet

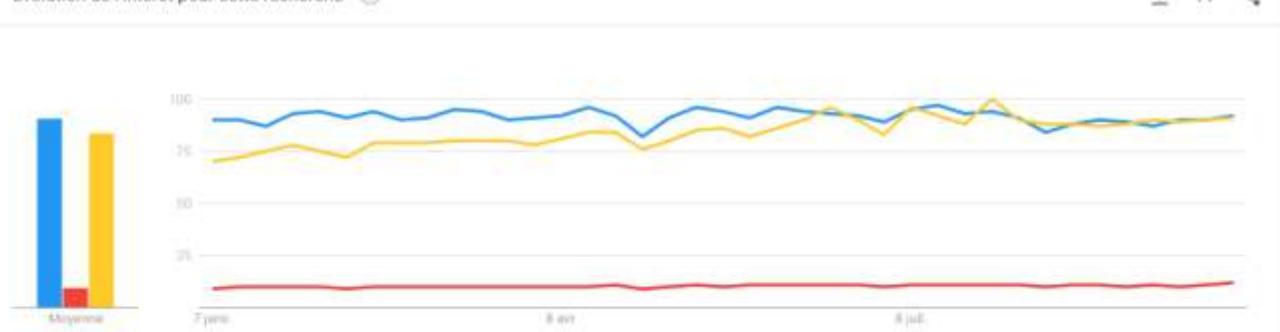
Vue.js  
Sujet

React  
JavaScript

+ Ajouter une comparaison

Dans tous les pays ▾ 01/01/2018 – 08/10/2018 ▾ Toutes catégories ▾ Recherche sur le Web ▾

Évolution de l'intérêt pour cette recherche ⓘ



## FRAMEWORKS JS

Nom	Tout le web	Top 10k	Top 100k	Top Million
Angular JS	90,5%	47,3%	47,5%	47,5%
Backbone.js	6,6%	51,4%	50,8%	51,0%
Meteor	1,9%	0,1%	0,3%	0,3%
Ember	0,6%	1,0%	1,1%	1,0%
SailsJS	0,5%	0,3%	0,3%	0,2%

Source : [Builtwith](#)

# JAVASCRIPT



- Sites

- <https://jquery.com/>
- [jquery.com/download/](https://jquery.com/download/)
- <https://developers.google.com/speed/libraries/#jquery>
- <https://www.asp.net/ajax/cdn#jQuery Releases on the CDN>

- *Include en ligne :*

- ```
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/  
3.1.1/jquery.min.js"></script>
```

# JAVASCRIPT



```
<!DOCTYPE html>
<html><head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
</script> </head>
<body>
<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>
</body> </html>
```

# JAVASCRIPT



- The jQuery library contains the following features:
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX
  - Utilities

# JAVASCRIPT



The **jQuery** syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

- Basic syntax is: **`$(selector).action()`**
  - A \$ sign to define/access jQuery
  - A `(selector)` to "query (or find)" HTML elements
  - A jQuery `action()` to be performed on the element(s)
- Examples:
  - `$(this).hide()` - hides the current element.
  - `$("p").hide()` - hides all `<p>` elements.
  - `$(".test").hide()` - hides all elements with class="test".
  - `$("#test").hide()` - hides the element with id="test".

# JAVASCRIPT



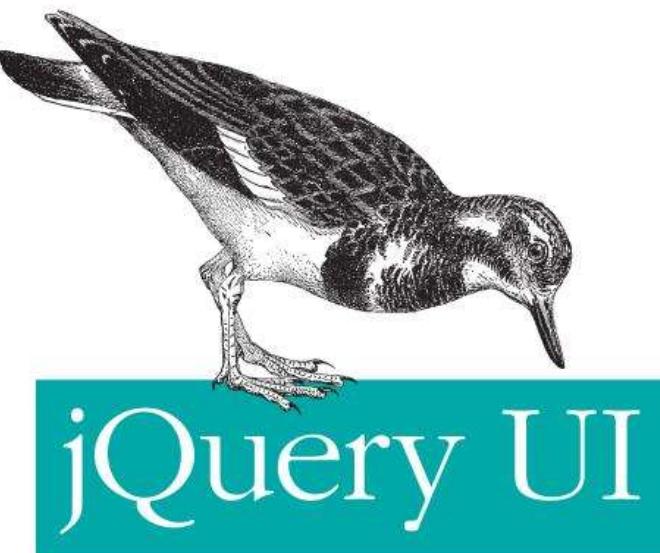
<code>\$('*')</code>	Selects all elements
<code>\$("p.intro")</code>	Selects all <p> elements with class="intro"
<code>\$("p:first")</code>	Selects the first <p> element
<code>\$("ul li:first")</code>	Selects the first <li> element of the first <ul>
<code>\$("ul li:first-child")</code>	Selects the first <li> element of every <ul>
<code>\$("[href]")</code>	Selects all elements with an href attribute
<code>\$("a[target='_blank']")</code>	Selects all <a> elements with a target attribute value equal to "_blank"
<code>\$("a[target!='_blank']")</code>	Selects all <a> elements with a target attribute value NOT equal to "_blank"
<code>\$(":button")</code>	Selects all <button> elements and <input> elements of type="button"
<code>\$("tr:even")</code>	Selects all even <tr> elements
<code>\$("tr:odd")</code>	Selects all odd <tr> elements

# JAVASCRIPT

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

# JAVASCRIPT

*A Code-Centered Approach to User Interface Design*



- **Sites**

- <https://jqueryui.com/>

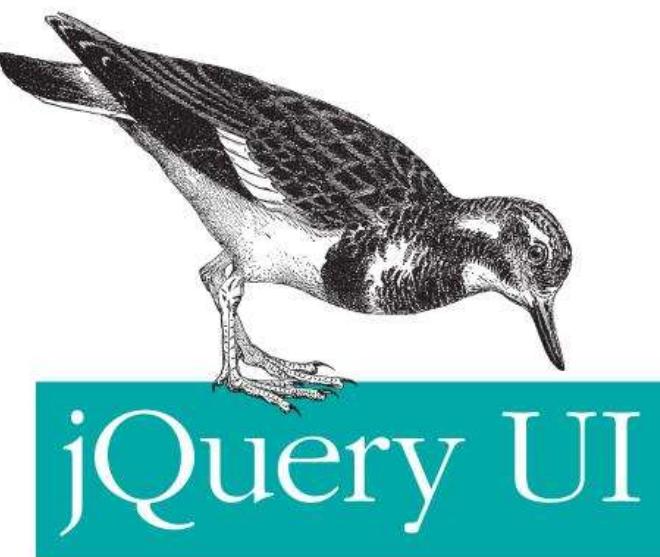
- jQuery UI is a widget and interaction library built on top of the jQuery JavaScript Library that you can use to build highly interactive web applications.

- **Testez**

- <http://jqueryui.com/demos/>

# JAVASCRIPT

*A Code-Centered Approach to User Interface Design*



O'REILLY®

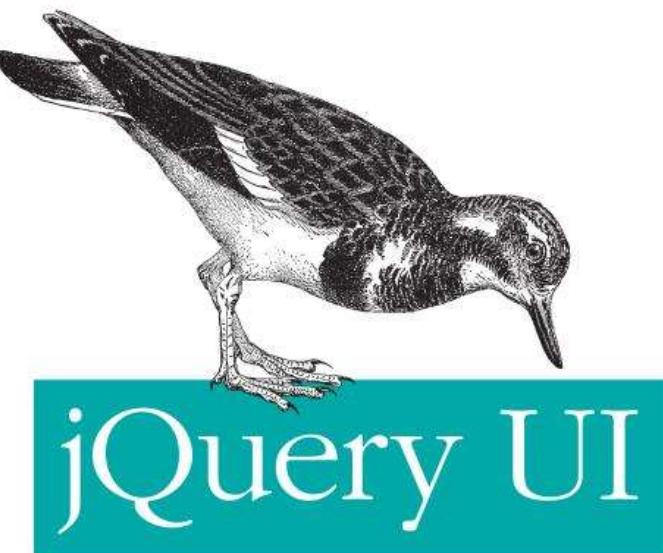
*Eric Sarrion*

- **Interactions**

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable

# JAVASCRIPT

*A Code-Centered Approach to User Interface Design*



O'REILLY®

*Eric Sarrion*

- **Widgets**

- Accordion
- Autocomplete
- Button
- Checkboxradio
- Controlgroup
- Datepicker
- Dialog
- Menu
- Progressbar
- Selectmenu
- Slider
- Spinner
- Tabs
- Tooltip

# JAVASCRIPT > JQUERY > EXEMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
</script>
</head>
<body>

<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
</html>
```

# JAVASCRIPT > JQUERY > EXEMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#hide").click(function(){
    $("p").hide();
  });
  $("#show").click(function(){
    $("p").show();
  });
});
</script>
</head>
<body>

<p>If you click on the "Hide" button, I will disappear.</p>

<button id="hide">Hide</button>
<button id="show">Show</button>

</body>
</html>
```

# JAVASCRIPT > JQUERY > EXEMPLE

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({
      left: '250px',
      opacity: '0.5',
      height: '150px',
      width: '150px'
    });
  });
});
</script>
</head>
<body>

<button>Start Animation</button>

<p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>
</html>
```

# FRAMEWORKS

BISGAMBIGLIA@UNIV-CORSE.FR

184

# FRAMEWORKS JAVASCRIPT



- Vue.js
- Angular.js (**Google**)
  - most used JavaScript framework for developing Single Page Web Applications
  - This framework gives super powers to HTML
- React.js (**Facebook**)
  - it comes to building large scale applications of extreme dynamic nature
- Ember.js
  - competes with the likes of Angular and React

# FRAMEWORKS JAVASCRIPT



VUE The Progressive  
JavaScript Framework

```
<!DOCTYPE html>

<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/vue">
      </script></head>
  <body>
    <div id="app">{{ message }}</div>
    <script> const data = { message: "Hello World !" };
      new Vue({ el: "#app", data });
    </script> </body> </html>
```

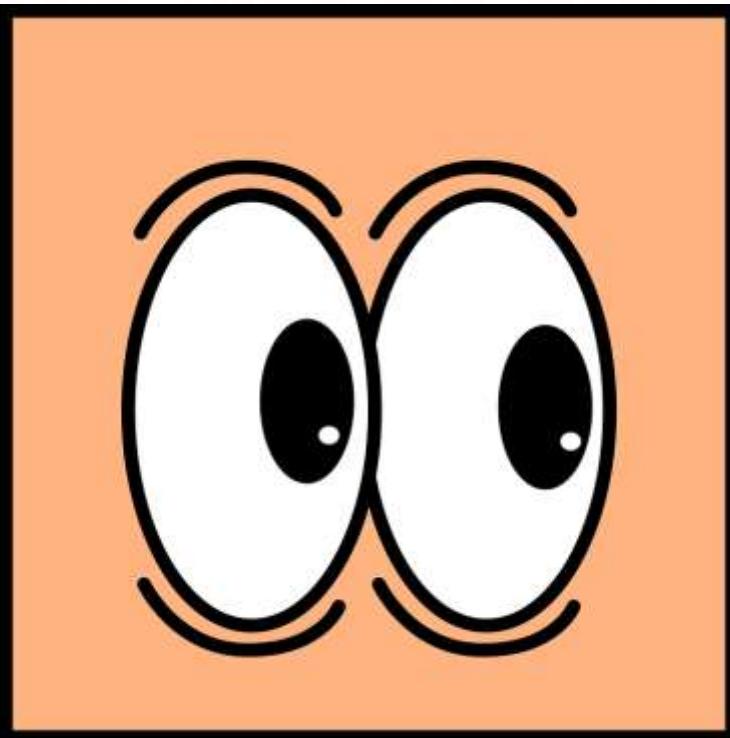
# JAVASCRIPT



A surveiller !

- Aurelia.js
  - tools and frameworks to support next generation of web development
- Meteor.js
  - building end to end mobile and web applications completely in JavaScript

# FRAMEWORK JS



- Cliquez !!!
- [https://makina-  
corpus.com/blog/societe/2017/retour-sur-le-  
petit-dejeuner-quel-framework-js-pour-2017](https://makina-corpus.com/blog/societe/2017/retour-sur-le-petit-dejeuner-quel-framework-js-pour-2017)

# JAVASCRIPT



A lire :

<http://maxlab.fr/veille/developpement-web-tendances-et-perspectives-pour-2016/>

<http://juliendubreuil.fr/blog/developpement/vison-sur-le-futur-du-developpement-web/>

# JAVASCRIPT



- **Comment bien coder**

- Une bonne page web doit être consultable et entièrement fonctionnelle sans Javascript
- Commentez vos scripts
- Bien nommer vos variables (conseil => Notation de type hongroise)
  - Exemple : aMonTableau
  - Débute par une ou deux lettres minuscules qui définissent le type suivies de mots attachés commençant par une majuscule
- Désolidarisez vos scripts de l'interface utilisateur

# JAVASCRIPT



- Ne redirigez et soumettez pas automatiquement les formulaires
- Testez vos méthodes
- Évitez les variables globales
  - Toute variable déclarée en dehors d'une fonction est globale, c'est-à-dire accessible et modifiable depuis n'importe quelle partie du script

# JAVASCRIPT



- Utiliser le Modèle objet

- D'un point de vue général, on assimile un objet à une collection de données avec lesquelles on peut avoir diverses interactions. Un objet peut être pourvu de propriétés (des variables spécifiques à l'objet) et de méthodes (des fonctions que seul l'objet peut invoquer)
- L'objet `window` est appelé objet global puisqu'il regroupe tout ce qui se trouve dans votre fenêtre

# JAVASCRIPT



- **Codez modulaire**
  - Si certaines de vos méthodes comportent des codes lourds et redondants, faites-en de nouvelles méthodes
- **Utilisez le mode strict**
  - "use strict";
- **Séparez comportement, présentation et structure (et données)**

# JAVASCRIPT



## Techniques JavaScript importantes

- Accès à un formulaire
- L'interaction se déroule au travers d'un formulaire obtenu par:
- `window.document.forms[numéro formulaire]`
- ou
- `window.document.forms[name_formulaire]`
- ou
- `this.form`
- 1<sup>er</sup> forme avantageuse au milieu d'une boucle ou lorsque le formulaire ne possède pas de paramètre name.
- 2<sup>ième</sup> forme plus lisible et tolérante aux changements de structure de la page.
- Attention: Vous ne pouvez pas accéder à un formulaire au moyen de JavaScript tant que le navigateur n'a pas traité la structure HTML et crée les objets (position des instructions JS)

# JAVASCRIPT



## Soumission et réinitialisation d'un formulaire

- Submit(): fonction permettant à un objet formulaire d'envoyer ces informations au serveur.
- Elle est appelée lorsque l'utilisateur valide le formulaire par:
- Un lien hypertexte, un bouton, autre gestionnaire d'événement...
- Exemple:

```
<html><body>

<form action= " " method=" get " >

<input name="a"><br>

<input type="button" value="OK"
       onClick="window.document.forms[0].submit()">

</form>

</body></html>
```

## Conclusion

- JavaScript sert à contrôler les données saisies dans des formulaires HTML, ou à interagir avec le document HTML via l'interface DOM ;
- Il est aussi utilisé pour réaliser des services dynamiques ;
- Il permet la réalisation rapide de scripts et de traitements ;
- Des exemples : gestion des dates et heures ; gestion des cookies ; gestion de la navigation ; animations graphiques ; etc.
- Il est sensible à la casse ;
- JavaScript est conçu pour limiter les risques pour le visiteur.

## Limites

- Compatibilité limitée entre navigateurs ;
- JavaScript ne permet pas d'écrire ou de lire un fichier sur le disque dur du visiteur ou sur le serveur ;
- JavaScript n'échange pas non plus avec d'autres machines connectées → pages chat impossible (sauf avec nodeJS)
- **Alternatives** : D'autres technologies existent comme, Java, .NET, etc.

# API HTML 5

[HTTPS://WWW.W3SCHOOLS.COM/HTML/HTML5\\_GEOLOCATION.ASP](https://www.w3schools.com/html/html5_geolocation.asp)

# API HTML 5

- <https://www.programmation-facile.com/api-javascript-html5-utiliser-sites-web/>
- **Geolocation**
- **Drag and Drop**
- **Web Storage**
- **Web Workers**
- **Formulaires**
- <https://developer.mozilla.org/fr/docs/Web/Guide/HTML/HML5>

- *Sémantique* : permet de décrire plus précisément votre contenu.
- *Connectivité* : permet de communiquer avec le serveur d'une façon nouvelle et innovante.
- *Hors-connexion & stockage* : permet aux pages web de stocker des données en local, côté client, et de fonctionner plus efficacement hors-connexion.
- *Multimédia* : rendre la vidéo et l'audio des citoyens de premier plan sur l'Open Web
- *Rendu 2D/3D et effets* : permet des options de présentation bien plus variées.
- *Performance & intégration* : offre une puissance bien plus grande et une meilleure utilisation du matériel de l'ordinateur.
- *Accès aux périphériques* : permet un usage de périphériques d'entrée et de sortie variés.
- *Style* : permet aux auteurs d'écrire des thèmes plus sophistiqués.

# API HTML 5 > FORMULAIRES

- <https://apprendre-la-programmation.net/securiser-formulaires/>

# API HTML 5



## SÉMANTIQUE

### Introduction à HTML5

Cet article vous présente la manière d'utiliser HTML5 dans vos conceptions de sites et d'applications web.

### Sections et structure des documents avec HTML5

Un aperçu de la nouvelle organisation des documents avec les nouveaux éléments de section en HTML5 : `<section>`, `<article>`, `<nav>`, `<header>`, `<footer>`, `<aside>` et `<hgroup>`.

### Utilisation de l'audio et de la video en HTML5

Les éléments `<audio>` et `<video>` en HTML5 permettent d'insérer et de manipuler dans vos pages de nouveaux contenus multimédias.

### Formulaires en HTML5

Un aperçu des améliorations apportées aux formulaires en HTML5 : l'API de validation, de nouveaux attributs, de nouvelles valeurs pour l'attribut `type` de `<input>` et le nouvel élément `<output>`.

### Nouveaux éléments d'HTML5

En plus des éléments de sections, de média et de formulaires, HTML5 ajoute de nombreux nouveaux éléments, comme `<mark>`, `<figure>`, `<figcaption>`, `<data>`, `<time>`, `<output>`, `<progress>`, ou encore `<meter>`.

### Amélioration de `<iframe>`

En utilisant les attributs `sandbox`, `seamless`, et `srcdoc`, les auteurs peuvent contrôler plus finement la sécurité et l'affichage de l'élément `<iframe>`.

### MathML

Permet d'insérer des formules mathématiques directement dans les pages web.

### Parseur conforme à HTML5



## PERFORMANCE ET INTÉGRATION

### Web Workers

Permet de déléguer l'interprétation du JavaScript sur des fils en arrière plan, et évite ainsi le ralentissement des événements d'interaction.

### XMLHttpRequest Niveau 2

Permet de récupérer de manière asynchrone certaines parties d'une page. Ceci permet d'afficher du contenu dynamique, s'adaptant en fonction du temps et des actions de l'utilisateur. Il s'agit de la technologie derrière l'AJAX.

### Moteurs JavaScript à compilation JIT (juste à temps)

La nouvelle génération de moteurs JavaScript est bien plus puissante et permet de meilleures performances dans l'interprétation du langage.

### API Historique

Permet la manipulation de l'historique du navigateur. Ceci est particulièrement utile sur les pages qui chargent du contenu dynamiquement.

### L'attribut contentEditable : transformer un site web en un wiki !

HTML5 a standardisé l'attribut `contentEditable` : apprenez-en plus sur cette fonctionnalité.

### Glisser et déposer

L'API de glisser et déposer HTML5 (drag and drop) permet de faire glisser des éléments au sein d'une page web. C'est aussi une API plus simple utilisable par les extensions et applications basées sur Mozilla.

### Gestion du focus en HTML

Les nouveaux attributs HTML5 `activeElement` et `hasFocus` sont gérés.

### Gestionnaires de protocoles web

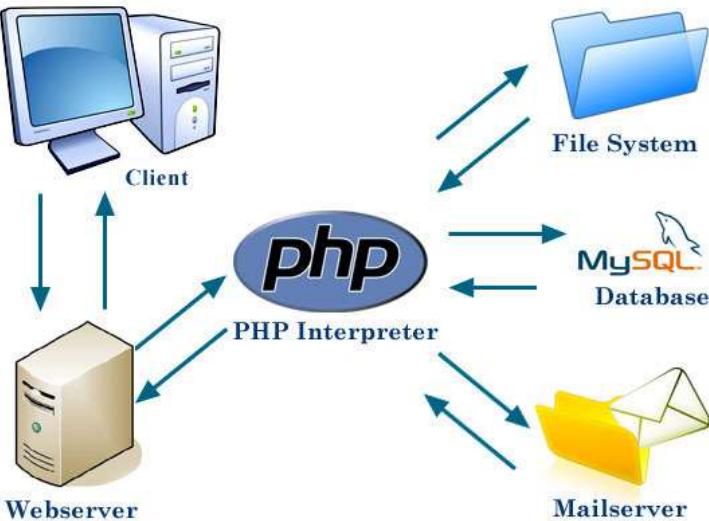
Il est à présent possible d'enregistrer des applications web en tant que gestionnaires de protocoles à l'aide de la méthode `navigator.registerProtocolHandler()`.

PHP



- Langage de script côté serveur
- Version [5.6.26](#) à regarder version 8
- Concurrents ASP, JEE, PYTHON
- Version stable 7.4.12

# PHP



- A intégrer dans du HTML mais interprété côté serveur
- Balise `<?php ... ?>`
- Avec PHP on :
  - Génère des pages dynamiques
  - Manipule des fichiers sur serveur
  - Manipule les cookies
  - Gère les données
  - Gère des utilisateurs (backoffice)

# PHP



- Installation

- Installer un serveur web, puis php
- Ou une solution tout en un, comme :
  - Wamp <http://www.wampserver.com>
  - Easyphp [www.easypht.org/](http://www.easypht.org/)
  - Xamp <https://www.apachefriends.org/fr>

- Utilisation

- Fichier .php
- À lancer depuis localhost

# PHP

Un exemple de  
[http://www.w3schools.com/php/php\\_syntax.asp](http://www.w3schools.com/php/php_syntax.asp)

Les mots clés en PHP ne sont pas sensibles à la casse echo  
= ECHO

Commentaires :

- // et # une ligne
- /\* ... \*/ multilignes

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

# PHP

Un autre exemple de  
[http://www.w3schools.com/php  
/php\\_syntax.asp](http://www.w3schools.com/php/php_syntax.asp)

Les variables doivent commencer par un \$

Les noms de variables sont sensibles à la casse \$toto ≠ \$Toto

Les “...” interprétent les variables echo "My car is " .  
\$color . "<br>"; =  
echo "My car is \$color <br>";

```
<!DOCTYPE html>
<html>
<body>
<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
</body>
</html>
```

# PHP



C'est un langage dynamiquement type

Les variables peuvent être définies avec trois types de portées différentes :

- local, portée limitée
- global, portée générale
- Static, portée partagée

# PHP

4 exemples de :

[http://www.w3schools.com  
/php/php\\_variables.asp](http://www.w3schools.com/php/php_variables.asp)

1

```
<?php
$x = 5; // global scope
function myTest() {
    echo "<p>Variable x inside function is:
$x</p>";
}
myTest();
echo "<p>Variable x outside function is:
$x</p>";
?>
```

# PHP

4 exemples de :

[http://www.w3schools.com  
/php/php\\_variables.asp](http://www.w3schools.com/php/php_variables.asp)

2

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is:
$x</p>";
}
myTest();
echo "<p>Variable x outside function is:
$x</p>";
?>
```

# PHP

4 exemples de :

[http://www.w3schools.com/php/php\\_variables.asp](http://www.w3schools.com/php/php_variables.asp)

3

```
<?php  
$x = 5;  
$y = 10;  
function myTest() {  
    global $x, $y;  
    $y = $x + $y;  
    // eq $GLOBALS['y'] = $GLOBALS['x'] +  
    $GLOBALS['y'];  
}  
myTest();  
echo $y;  
?>
```

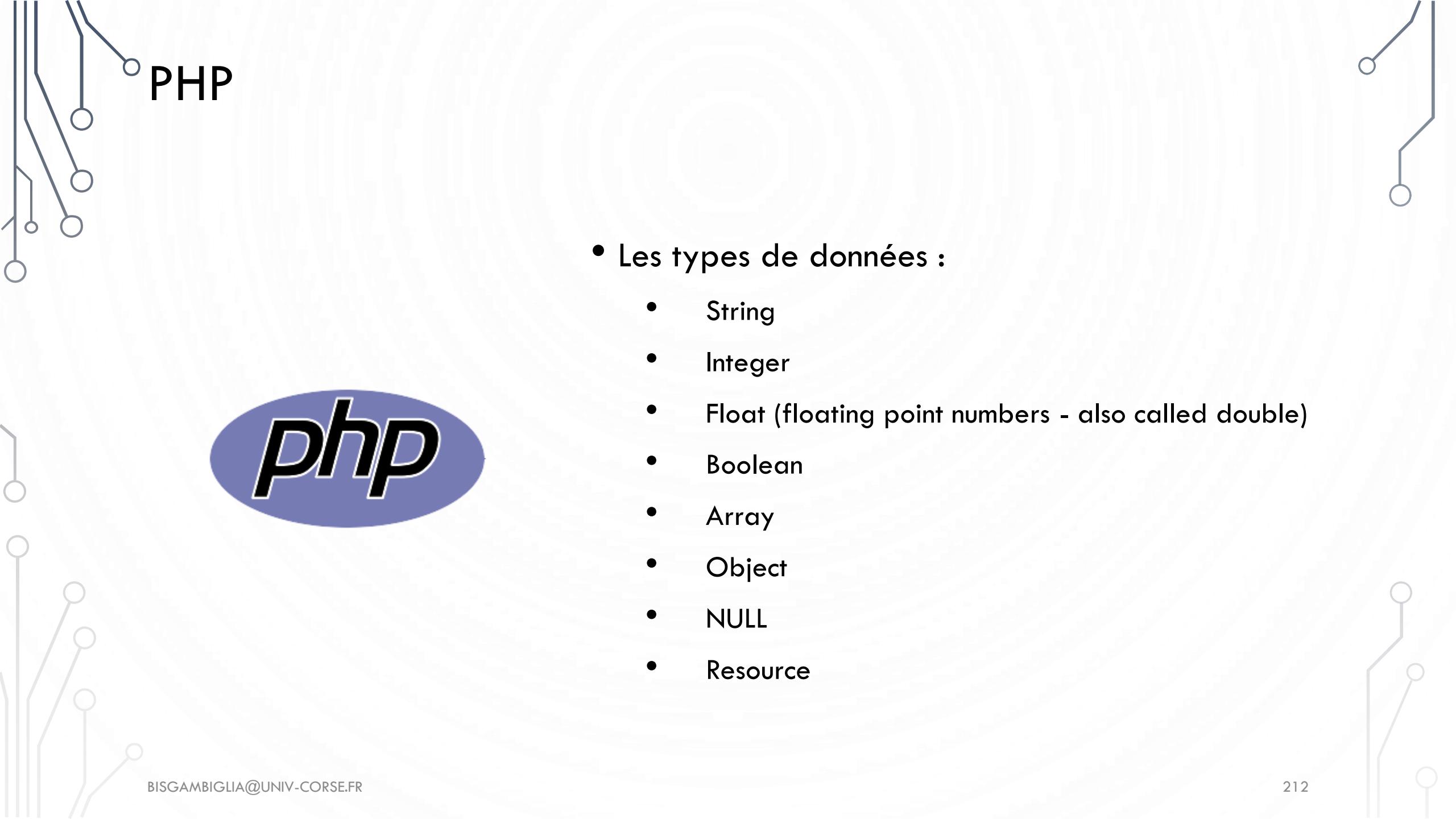
# PHP

4 exemples de :

[http://www.w3schools.com  
/php/php\\_variables.asp](http://www.w3schools.com/php/php_variables.asp)

# 4

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}
myTest();
myTest();
myTest();
?>
```



# PHP



- Les types de données :

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

# PHP



- Affichage : `echo, echo() ou print, print()`
- Débogage : `var_dump()`
- Constante : `define(name, value, case-insensitive)`
- Opérateurs : `+ =, ++, ==, ===, !=, <>, .`
- Condition `if else elseif`
- Boucle `while for`
- Fonction : `function name(...) {...}`
  - Argument par défaut `function foo($a = 5)`

# PHP

- Les tableaux un exemple du  
[http://www.w3schools.com  
/php/php\\_arrays.asp](http://www.w3schools.com/php/php_arrays.asp)
- Fonctions de trie :
  - sort()
  - rsort()

```
$cars = array("Volvo", "BMW", "Toyota");
```

Eq.

```
$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";
```

```
echo count($cars);
```

```
$arrlength = count($cars);  
for($x = 0; $x < $arrlength; $x++) {  
    echo $cars[$x];  
    echo "<br>";  
}
```

# PHP

- Les tableaux associatifs ou (Dict) un exemple du [http://www.w3schools.com/php/php\\_arrays.asp](http://www.w3schools.com/php/php_arrays.asp)
- Fonctions de trie :
  - asort()
  - ksort()
  - arsort()
  - krsort()

```
$age = array("Peter"=>"35", "Ben"=>"37",  
"Joe"=>"43");
```

Eq.

```
$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";  
echo "Peter is " . $age['Peter'] . " years old.";  
  
foreach($age as $x => $x_value) {  
    echo "Key=" . $x . ", Value=" . $x_value;  
    echo "<br>";
```

# PHP

- Les tableaux à connaître

- ***\$GLOBALS***
- ***\$\_SERVER***
- ***\$\_REQUEST***
- ***\$\_POST***
- ***\$\_GET***
- ***\$\_FILES***
- ***\$\_ENV***
- ***\$\_COOKIE***
- ***\$\_SESSION***

```
<?php  
echo $_SERVER['PHP_SELF'];  
echo "<br>";  
echo $_SERVER['SERVER_NAME'];  
echo "<br>";  
echo $_SERVER['HTTP_HOST'];  
echo "<br>";  
echo $_SERVER['HTTP_REFERER'];  
echo "<br>";  
echo $_SERVER['HTTP_USER_AGENT'];  
echo "<br>";  
echo $_SERVER['SCRIPT_NAME'];  
?>
```

# PHP

- Les tableaux à connaître
  - ***\$GLOBALS***
  - ***\$\_SERVER***
  - ***\$\_REQUEST***
  - ***\$\_POST***
  - ***\$\_GET***
  - ***\$\_FILES***
  - ***\$\_ENV***
  - ***\$\_COOKIE***
  - ***\$\_SESSION***

```
<!DOCTYPE html>
<html>
<body>
<form method="post" action=<?php echo $_SERVER['PHP_SELF'];?>>
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

# PHP

- **Formulaire html**

```
<html>  
<body>  
<form action="welcome.php"  
method="post">  
  
Name: <input type="text"  
name="name"><br>  
  
E-mail: <input type="text"  
name="email"><br>  
  
<input type="submit">  
  
</form>  
</body>  
</html>
```

- **Formulaire php**

```
<html>  
<body>  
  
Welcome <?php echo $_POST["name"]; ?><br>  
Your email address is: <?php echo $_POST["email"]; ?>  
  
</body>  
</html>
```

Dans l'URL après page tapez : /%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E

# PHP

- **Formulaire html**

```
<html>
<body>

<form action="<?php echo
htmlspecialchars('welcome.php');?>"
method="post">

Name: <input type="text"
name="name"><br>
E-mail: <input type="text"
name="email"><br>
// Ajouter website, comment, gender
<input type="submit">
</form>
</body>
</html>
```

```
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = test_input($_POST["name"]);
    $email = test_input($_POST["email"]);
    $website = test_input($_POST["website"]);
    $comment = test_input($_POST["comment"]);
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
```

# PHP

- **Formulaire html**

```
<html>  
<body>  
  
<form action="w <?php echo  
htmlspecialchars(welcome.php);?>"  
method="post">  
  
Name: <input type="text"  
name="name"><br>  
  
E-mail: <input type="text"  
name="email"><br>  
  
<input type="submit">  
  
</form>  
</body>  
</html>
```

```
<?php  
// define variables and set to empty values  
$nameErr = $emailErr = $genderErr = $websiteErr = "";  
$name = $email = $gender = $comment = $website = "";  
  
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    if (empty($_POST["name"])) {  
        $nameErr = "Name is required";  
    } else {  
        $name = test_input($_POST["name"]);  
    }  
}
```



PHP

- Voilà pour les bases de la version 5 et la 7 !

# PHP 7

- Amélioration des performances
  - Les gains (entre PHP 5.6 et PHP 7) peuvent atteindre 50% en temps d'exécution CPU, et près de 50% également en consommation de mémoire
- Nouvel opérateur `<=>` et `??`
  - `$prenom = $_GET['user'] ?? 'personne';`
- Déclaration des fonctions avec type de retour
  - `function foo(): array { return []; }`
- Simplification du switch

<http://www.journaldunet.com/web-tech/developpeur/1152109-php-7-la-future-version-majeure-de-php-au-crible/>

# PHP

- **A voir encore :**
  - **Gestion des fichiers**
  - **Gestion des BD**
  - **XML & JSON**
  - **Modèle objet**
  - **AJAX**
- **Limitations**
  - **Concurrents**
    - .NET (*non libre et langages propriétaires*)
    - JEE (plus ancien, plus mature, plus robuste) complexe et infrastructure lourde
    - Ruby, Python, ou JavaScript côté serveur (Node - Meteor) trop jeune mais de forte possibilités
  - **PHP**
    - Pas adapté pour les énormes sites (Amazon, Facebook)
    - Pas adapté pour les applications natives

# PHP

- Exemple de fichier txt

AJAX = Asynchronous JavaScript  
and XML

CSS = Cascading Style Sheets

HTML = Hyper Text Markup

Language

PHP = PHP Hypertext

Preprocessor

SQL = Structured Query

Language

SVG = Scalable Vector

Graphics

XML = EXtensible Markup

Language

- Exemple de fichier php

```
<!DOCTYPE html>
<html>
<body>
<?php
myfile = fopen("webdictionary.txt", "r") or
die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
</body>
</html>
```

# PHP

- Exemple de fichier txt

AJAX = Asynchronous JavaScript  
and XML

CSS = Cascading Style Sheets

HTML = Hyper Text Markup  
Language

PHP = PHP Hypertext  
Preprocessor

SQL = Structured Query  
Language

SVG = Scalable Vector Graphics

XML = EXtensible Markup  
Language

- Exemple de fichier php

```
<!DOCTYPE html>
<html>
<body>
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to
open file!");
while(!feof($myfile)) {
    echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
</body>
</html>
```

# PHP



```
<?php  
  
$myfile = fopen("newfile.txt", "w") or  
die("Unable to open file!");  
  
$txt = "Mickey Mouse\n";  
  
fwrite($myfile, $txt);  
  
$txt = "Minnie Mouse\n";  
  
fwrite($myfile, $txt);  
  
fclose($myfile);  
  
?>
```

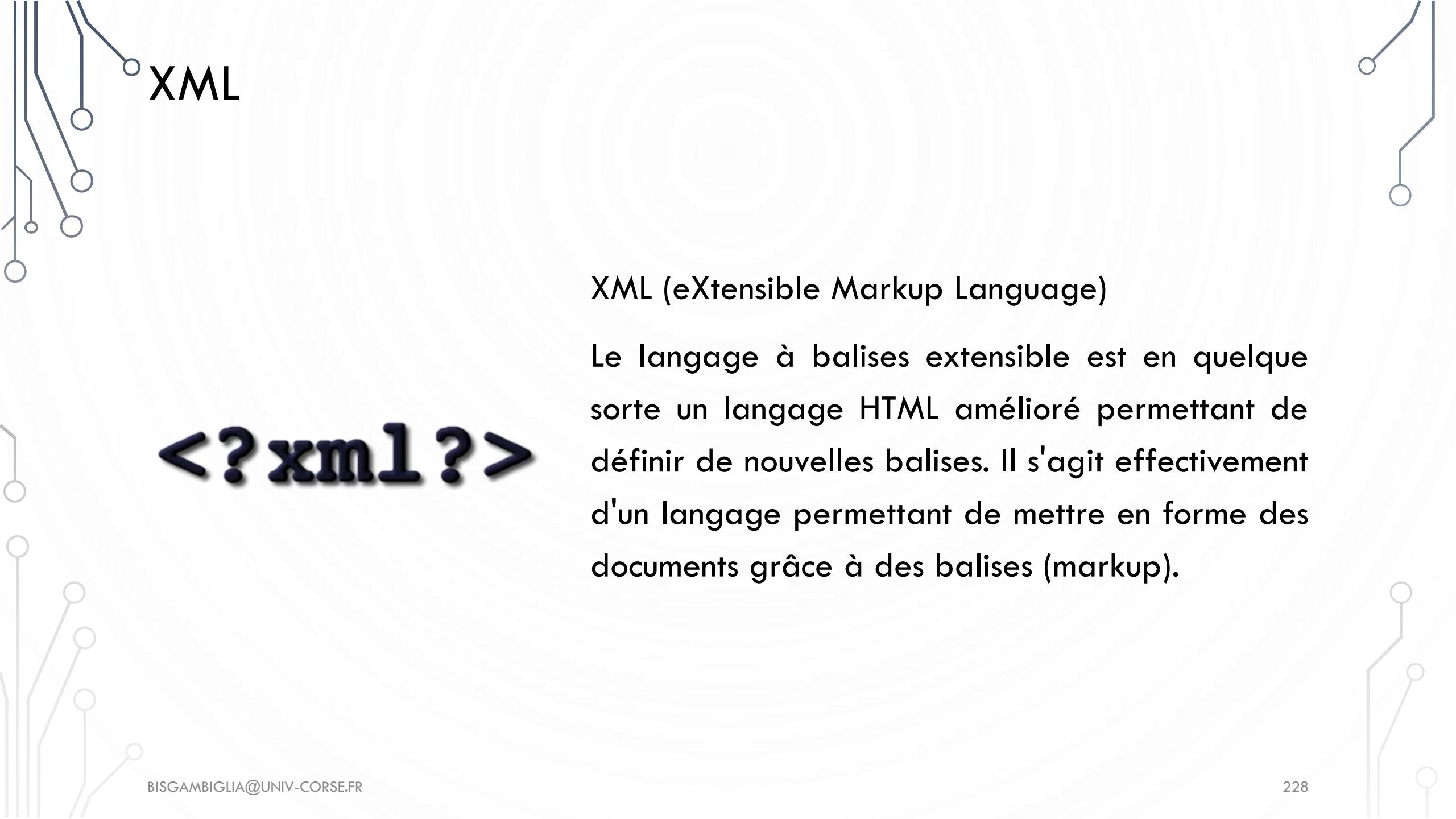
# PHP

- Exercice

A vous !

1. Créer un fichier txt (ou csv) avec les prénoms et noms de vos voisins
2. En PHP lisez ce fichier pour créer une page HTML qui affiche chaque ligne comme une puces (`<ul><li></li></ul>`)
3. Ajouter une fonction JavaScript qui tire un nombre aléatoire entre 1 et le nombre de puces
4. Ajouter un bouton pour lancer la fonction
5. L'item d'indice sélectionné doit passer en gras

```
<!DOCTYPE html>
<html>
<body>
<?php
myfile = fopen("noms.txt", "r") or die("Unable to open file!");
echo "<ul id='myList'>";
$i= 1;
while(!feof($myfile)) {
    echo "<li name=\"".$i."\">".fgets($myfile) . "</li>";
    $i++;
}
echo "</ul>";
fclose($myfile);
?>
<button type="button" onclick=myRand();>Rand!</button>
</body>
<script>
function myRand()
{
    var isize = document.getElementById('myList').childNodes.length;
    var iRand = Math.floor((Math.random() * isize));
    document.getElementById('myList').childNodes[iRand].style.fontweight = 'bold';
}
</script>
</html>
```

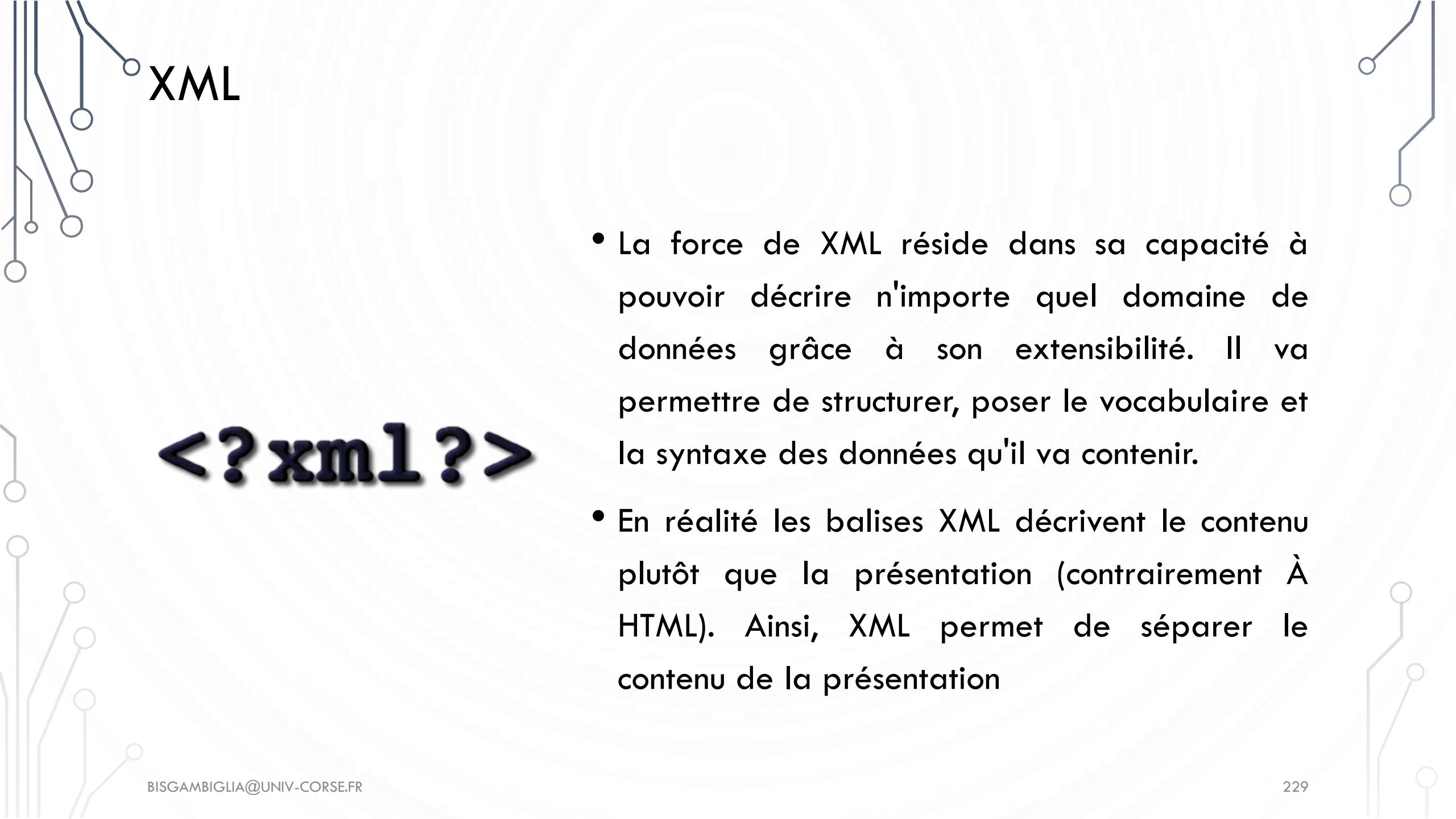


XML

<?xml?>

## XML (eXtensible Markup Language)

Le langage à balises extensible est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises (markup).



XML

<?xml?>

- La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.
- En réalité les balises XML décrivent le contenu plutôt que la présentation (contrairement à HTML). Ainsi, XML permet de séparer le contenu de la présentation

## Avantages

- La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML
  - Autodescriptif et extensible
  - Une structure arborescente : permettant de modéliser la majorité des problèmes informatiques
  - Universalité et portabilité : les différents jeux de caractères sont pris en compte
  - Déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP
  - Intégrabilité : un document XML est utilisable par toute application pourvue d'un parser (c'est-à-dire un logiciel permettant d'analyser un code XML)
  - Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications
- 

# XML

Il existe plusieurs solutions pour mettre en forme un document XML, comme :

- CSS (Cascading StyleSheet), la solution la plus utilisée actuellement, étant donné qu'il s'agit d'un standard qui a déjà fait ses preuves avec HTML
- XSL (eXtensible StyleSheet Language), un langage de feuilles de style extensible développé spécialement pour XML. Toutefois, ce nouveau langage n'est pas reconnu pour l'instant comme un standard officiel
- XSLT (eXtensible StyleSheet Language Transformation). Il s'agit d'une recommandation W3C du 16 novembre 1999, permettant de transformer un document XML en document HTML accompagné de feuilles de style

# XML

XML fournit un moyen de vérifier la syntaxe d'un document grâce aux DTD (Document Type Definition)

- Il s'agit d'un fichier décrivant la structure des documents y faisant référence grâce à un langage adapté. Ainsi un document XML doit suivre scrupuleusement les conventions de notation XML et peut éventuellement faire référence à une DTD décrivant l'imbrication des éléments possibles.
- Un document suivant les règles de XML est appelé document bien formé. Un document XML possédant une DTD et étant conforme à celle-ci est appelé document valide.

# XML

## Exemple

```
<annuaire>
  <personne class = "etudiant">
    <nom>Pillou</nom>
    <prenom>Jean-Francois</prenom>
    <telephone>555-123456</telephone>
    <email>webmaster@commentcamarche.net</email>
  </personne>
  <personne>
  ...
  </personne>
</annuaire>
```

# XML

## Exemple de lecture

```
<?php  
$dom = new DomDocument;  
$dom->load("test.xml");  
$listeNoms = $dom->  
getElementsByTagName('nom');  
foreach($listeNoms as $noms)  
    echo $noms->firstChild->nodeValue . "<br />";  
  
    echo "---<br />";  
?>
```

# XML

## Exemple

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<continents>
    <europe>
        <pays>France</pays>
        <pays>Belgique</pays>
        <pays>Espagne</pays>
    </europe>
    <asie>
        <pays>Japon</pays>
        <pays>Inde</pays>
    </asie>
</continents>
```

# XML

## Exemple de lecture

```
<?php  
    $dom = new DomDocument;  
    $dom->load("test.xml");  
    $listePays = $dom->getElementsByTagName('pays');  
    foreach($listePays as $pays)  
        echo $pays->firstChild->nodeValue . "<br />";  
  
    echo "---<br />";  
  
    $europe = $dom->getElementsByTagName('europe')->item(0);  
    $listePaysEurope = $europe->getElementsByTagName('pays');  
    foreach($listePaysEurope as $pays)  
        echo $pays->firstChild->nodeValue . "<br />";  
?>
```

# XML

Exemple création en php à partir d'une base de données

```
<?php
require "connect.php";
$Fnm = "mon_dossier/mon_fichier.xml";
$reponse = mysql_query("SELECT * FROM voiture ") or die(mysql_error()); // Requête SQL
$xml = '<?xml version="1.0" encoding="ISO-8859-1"?>';
$xml .= '<articles>';
while( $row = mysql_fetch_assoc($reponse) ) {
    $xml .= '<article id="'. $row['idd'] .'">';
    $xml .= '<titre>' . htmlspecialchars($row['marque']) . '</titre>';
    $xml .= '<contenu>' . htmlspecialchars($row['modele']) . '</contenu>';
    $xml .= '</article>';
}
$xml .= '</articles>';
$inF = fopen($Fnm,"w");
fwrite($inF,$xml);
fclose($inF); ?>
```

# JSON



JavaScript Object Notation est une forme de données textuelle en JavaScript.

Principaux avantages :

un format d'écriture simple et léger, et un langage nativement interprété contrairement au XML

# JSON

Il s'agit donc d'une arborescence de données, inspirée de XML mais dont l'emploi en JavaScript est plus aisé et plus performant, à partir du moment où on en connaît la structure.

Les types utilisables sont issus de JavaScript, on retrouve notamment tout ce qui est booléen (Boolean), valeur numérique (Number), chaîne de texte (String), tableau (Array), objet (Object) ou null. On combine le tout au sein d'un objet, le plus souvent, pour utiliser une notation clé:valeur afin de retrouver les variables dans l'arborescence des données.

# JSON

Exemple

```
var courses = {  
    "fruits": [  
        { "kiwis": 3,  
        "mangues": 4,  
        "pommes": null  
    },  
    { "panier": true },  
],  
    "legumes":  
        { "patates": "amandine",  
        "figues": "de barbarie",  
        "poireaux": false  
    } };
```

# JSON

```
var courses = {  
    "fruits": [  
        { "kiwis": 3,  
          "mangues": 4,  
          "pommes": null  
        },  
        { "panier": true },  
    ],  
    "legumes":  
        { "patates": "amandine",  
          "figues": "de barbarie",  
          "poireaux": false  
    } };
```

# JSON VS XML

```
<?xml version="1.0" ?>  
<root>  
  <fruits>  
    <item>  
      <kiwis>3</kiwis>  
      <mangues>4</mangues>  
      <pommes></pommes>  
    </item>  
    <item>  
      <panier>true</panier>  
    </item>  
  </fruits>  
  <legumes>  
    <patates>amandine</patates>  
    <figues>de barbarie</figues>  
    <poireaux>false</poireaux>  
  </legumes>  
</root>
```

# JSON

Exemple :

On peut visualiser le contenu de la variable sous forme d'arborescence grâce à la console JavaScript et la fonction `console.log` (sous Firefox équipé de Firebug par exemple, en ayant activé l'onglet Console) :

```
console.log(courses);
```

Pour lire une valeur en particulier :

```
courses.fruits[0].kiwis;  
// Retourne 3
```

```
fruits
  0
    kiwis
    mangues
    pommes
  1
    panier
legumes
  figues
  patates
  poireaux
[ Object { kiwis=3, mangues=4 }, Object {
  panier=true } ]
Object { kiwis=3, mangues=4 }
3
4
null
Object { panier=true }
true
Object { patates="amandine", figues="de barbarie",
poireaux=false }
"de barbarie"
"amandine"
false
```

# JSON

Exemple :

[http://www.w3schools.com  
/js/json\\_intro.asp](http://www.w3schools.com/js/js_json_intro.asp)

```
<!DOCTYPE html> <html> <body>
<h2>JSON Object Creation in JavaScript</h2>
<p id="demo"></p>
<script>
var text = '{"name":"John Johnson","street":"Oslo West
16","phone":"555 1234567"}';
var obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.name + "<br>" +
obj.street + "<br>" +
obj.phone;
</script> </body> </html>
```

# JSON

## FORCE

La puissance de JSON provient du fait que l'on peut récupérer une chaîne de texte ou un fichier dans ce format pour l'exploiter directement avec la fonction **eval()** de JavaScript afin de constituer un objet. On peut alors s'adresser directement aux propriétés membres de cet objet (et de ses sous-membres) sans avoir à écrire d'autres fonctions d'analyse du texte

# JSON

## Exemple

A voir

<http://www.alsacreations.com/article/lire/1675-json-stockage-leger-pratique-donnees-multiples.html>

```
if(typeof JSON!="undefined")  
  
var textejson =  
JSON.stringify({"kiwis":3,"mangues":4})  
console.log(textejson);
```

JSON



- json\_decode: transforme une chaîne de caractères JSON en variable PHP
- json\_encode: transforme une variable PHP en chaîne de caractères JSON

A lire <http://30minparjour.labnbox.fr/blog/2013/07/17/manipuler-du-json-en-php/>

# BASE DE DONNÉES

Ensemble de données organisé en vue de son utilisation par des programmes correspondant à des applications distinctes et de manière à faciliter l'évolution indépendante des données et des programmes.

Anglais : data base.

<http://www.futura-sciences.com/tech/definitions/informatique-base-donnees-518/>



# BASE DE DONNÉES

## Avec PHP

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

# BASE DE DONNÉES

## Avec PHP

- 3 types de fonctions :
1. Objet (`new`)
  2. Procédural (fonction)
  3. D'abstraction PDO (`new`)

Tous les exemples sont dispo sur  
[http://www.w3schools.com/php/php\\_mysql\\_connect.asp](http://www.w3schools.com/php/php_mysql_connect.asp)

# BASE DE DONNÉES

- Plusieurs étapes :
- 1. Connexion**
  - 2. Requête**
  - 3. Exécution**
  - 4. Affichage**

```
<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
  
// Create connection  
$conn = new mysqli($servername, $username, $password);  
  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}  
echo "Connected successfully";  
?>  
  
$conn->close();
```

# BASE DE DONNÉES

- Plusieurs étapes :
1. Connexion
  - 2. Requête**
  3. Exécution
  4. Affichage

```
$sql = "SELECT item1, item2, item3 FROM MyTable";
```

```
$sql = "INSERT INTO MyTable (item1, item2, item3)  
VALUES ('value1', 'value2', 'value3');
```

```
$sql = "DELETE FROM MyTable WHERE item1=3";
```

```
$sql = "UPDATE MyTable SET item2='value2'  
WHERE item1=2";
```

# BASE DE DONNÉES

Plusieurs étapes :

- 1. Connexion**
- 2. Requête**
- 3. Exécution**
- 4. Affichage**

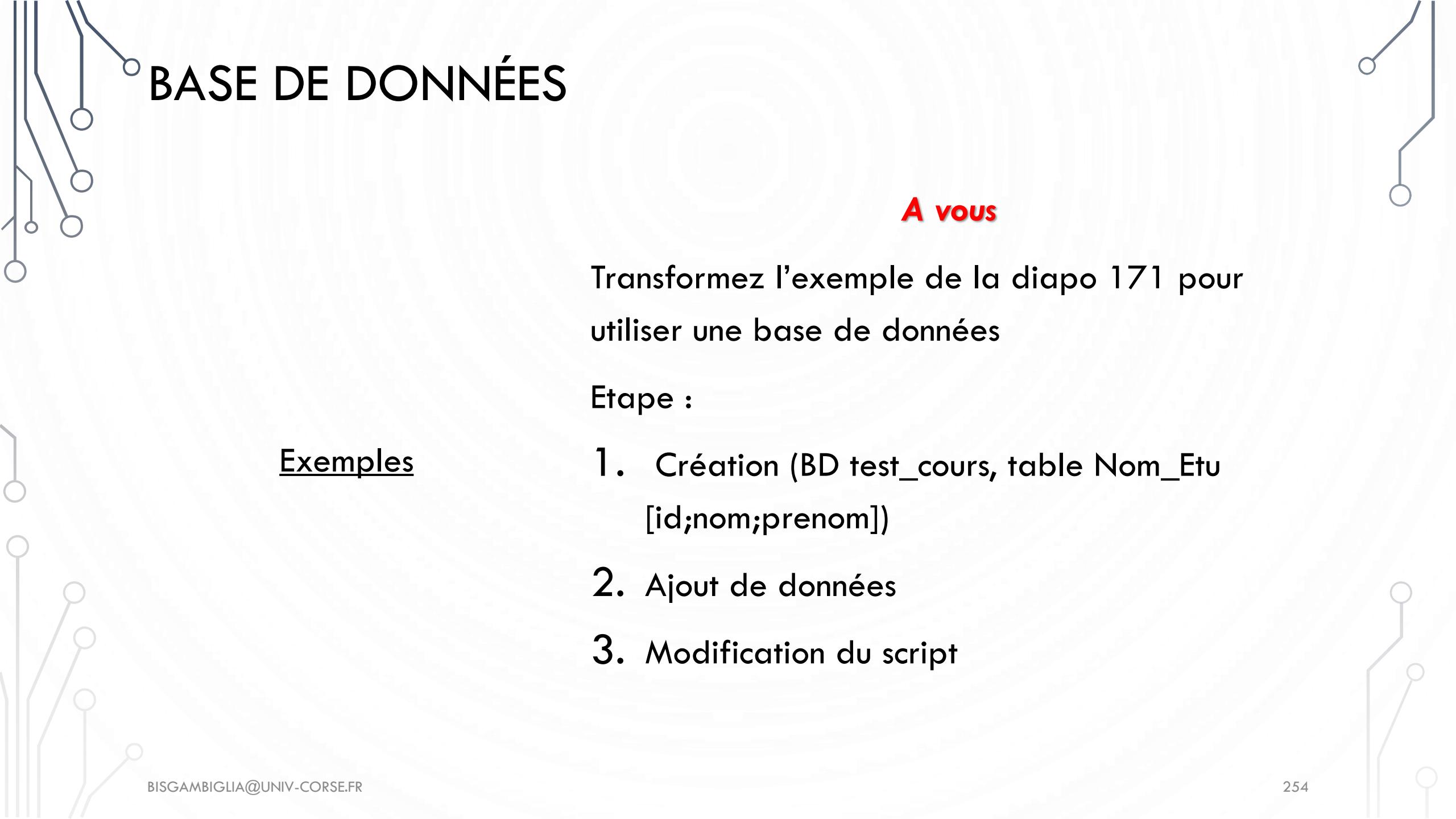
```
if ($conn->query($sql) === TRUE) {  
    echo "successfully";  
} else {  
    echo "Error : " . $conn->error;  
}  
  
$result = $conn->query($sql);
```

# BASE DE DONNÉES

- Plusieurs étapes :
1. Connexion
  2. Requête
  3. Exécution
  - 4. Affichage**

```
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["item1"]. " - Name: " .
        $row["item2"]. " " . $row["item3"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
```



# BASE DE DONNÉES

**A vous**

Transformez l'exemple de la diapo 171 pour utiliser une base de données

Etape :

## Exemples

1. Crédation (BD test\_cours, table Nom\_Etu  
[id;nom;prenom])
2. Ajout de données
3. Modification du script

# PHP

- Exercice

A vous !

1. Créer une base de données `expl_pw` avec une table : `id` ; `prénom` ; `nom` (à remplir)
2. En PHP connectez vous à la base puis à l'aide d'une requête il va falloir créer un page HTML qui affiche chaque ligne comme une puce (`<ul><li></li></ul>`)
3. Ajouter un fonction JavaScript qui tire un nombre aléatoire entre 1 et le nombre de puces
4. Ajouter un bouton pour lancer la fonction
5. L'item d'indice sélectionné doit passer en gras

AJAX



Src : <http://www.abondance.com/actualites/20151015-15675-google-modifie-son-systeme-dindexation-des-sites-en-ajax.html>

# AJAX

Définitions :

**synchrone** : modalités d'échange d'informations en direct (exemple : téléphone, visioconférence, visiophonie, audiophonie, etc.).

**asynchrone** : modalités d'échange d'informations en différé (mél, forum, etc.)

- Permet d'appeler des données depuis un client Web sur un serveur en **asynchrone**
- AJAX nécessite une architecture client/serveur Web
- Composants conformes aux standards du W3C

# AJAX

# AJAX

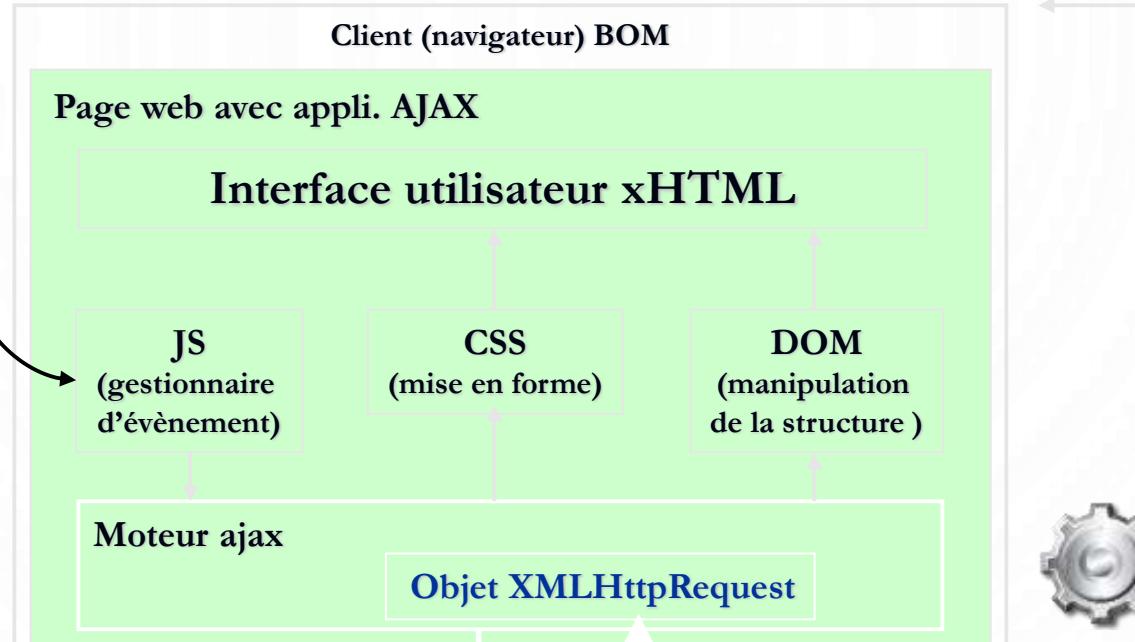
AJAX est une façon d'utiliser intelligemment plusieurs technologies préexistantes :

- Une présentation sur des standards utilisant **HTML** et **CSS**
- Un affichage dynamique et une interaction utilisant le Modèle Objet Document (**DOM**)
- L'échange de données et leur manipulation en utilisant **XML**, **XSLT** et **JSON**
- La récupération asynchrone de données en utilisant **XMLHttpRequest**
- Le langage de scripts **JavaScript** pour lier le tout ensemble

# AJAX

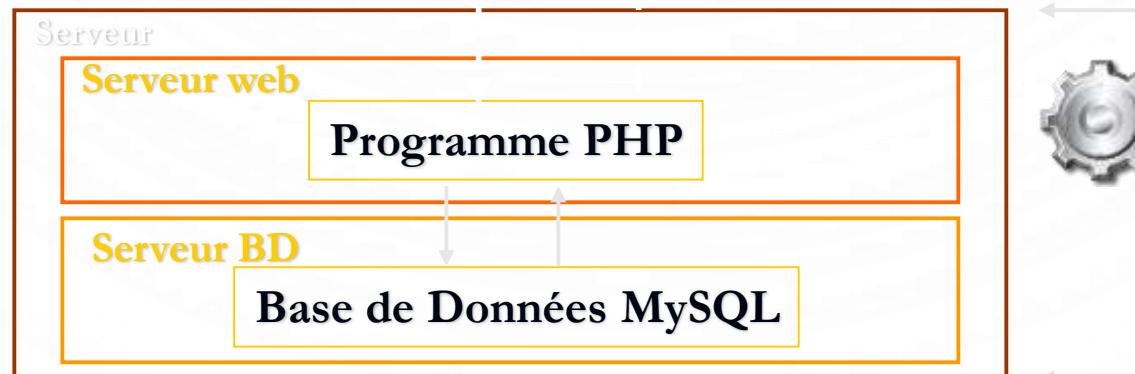


événement



Echange HTTP asynchrone

<data> (text,xml, json)



# Répondre

AJAX

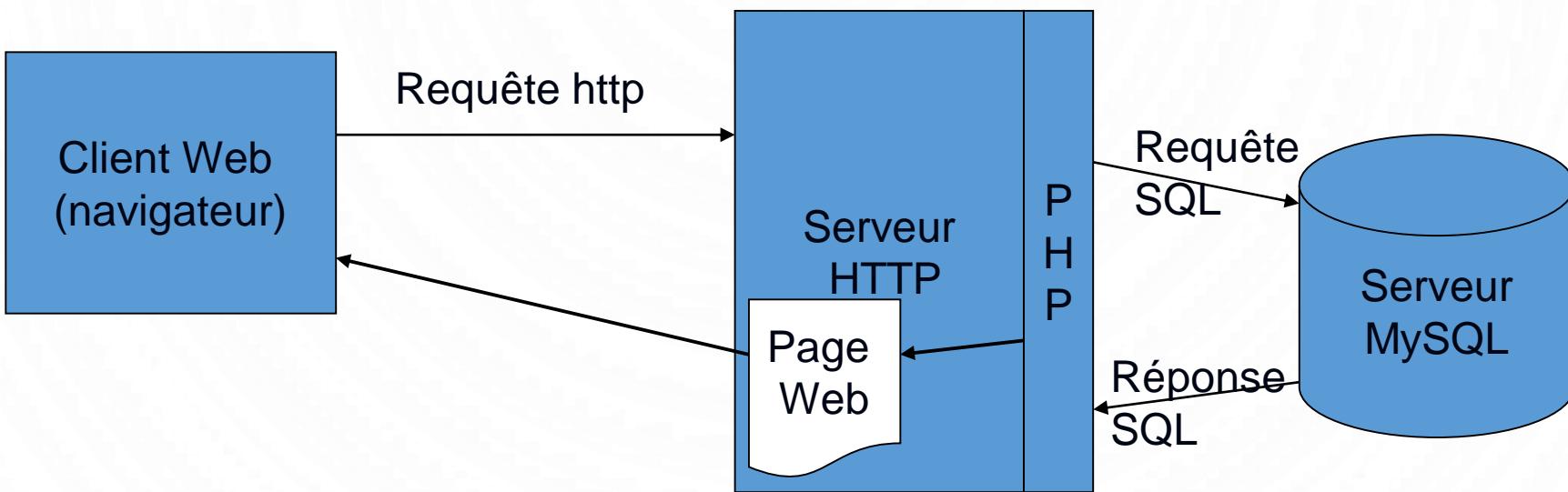
**Architecture 2-tiers**

⇒ client-serveur

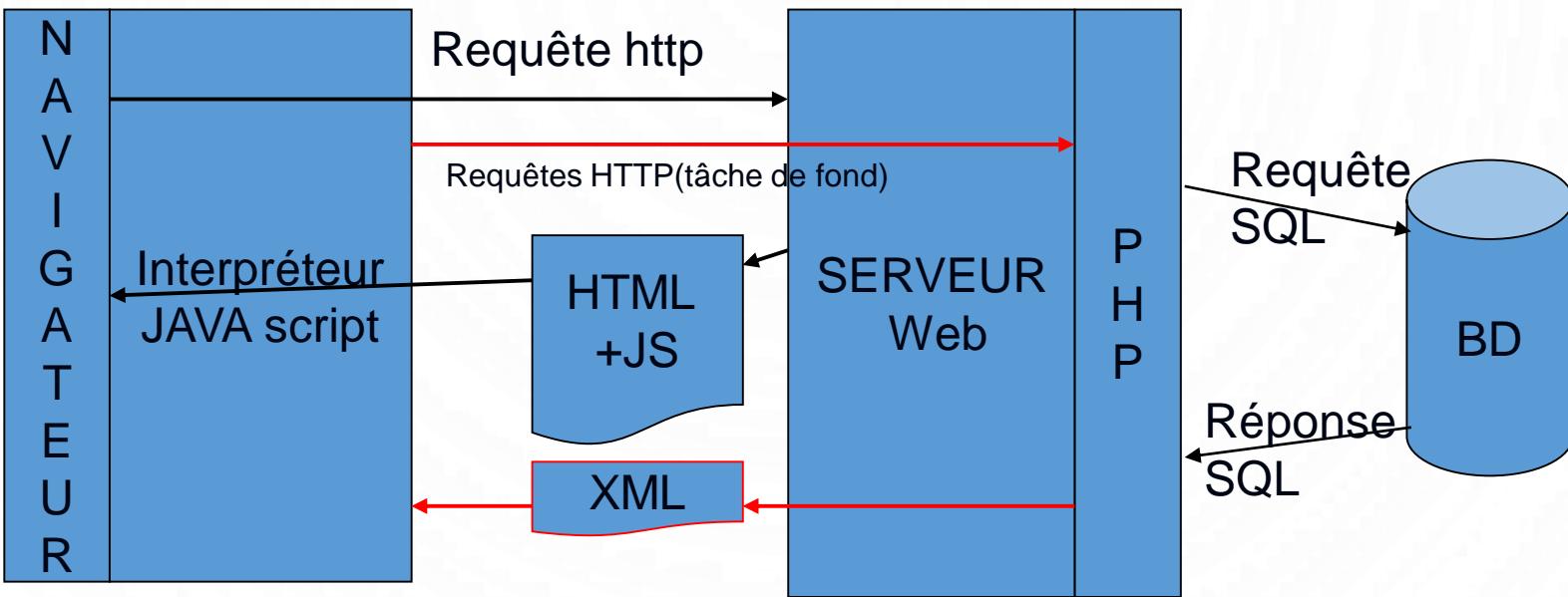
**Architecture 3-tiers**

=> client-serveur d'applications-base de données

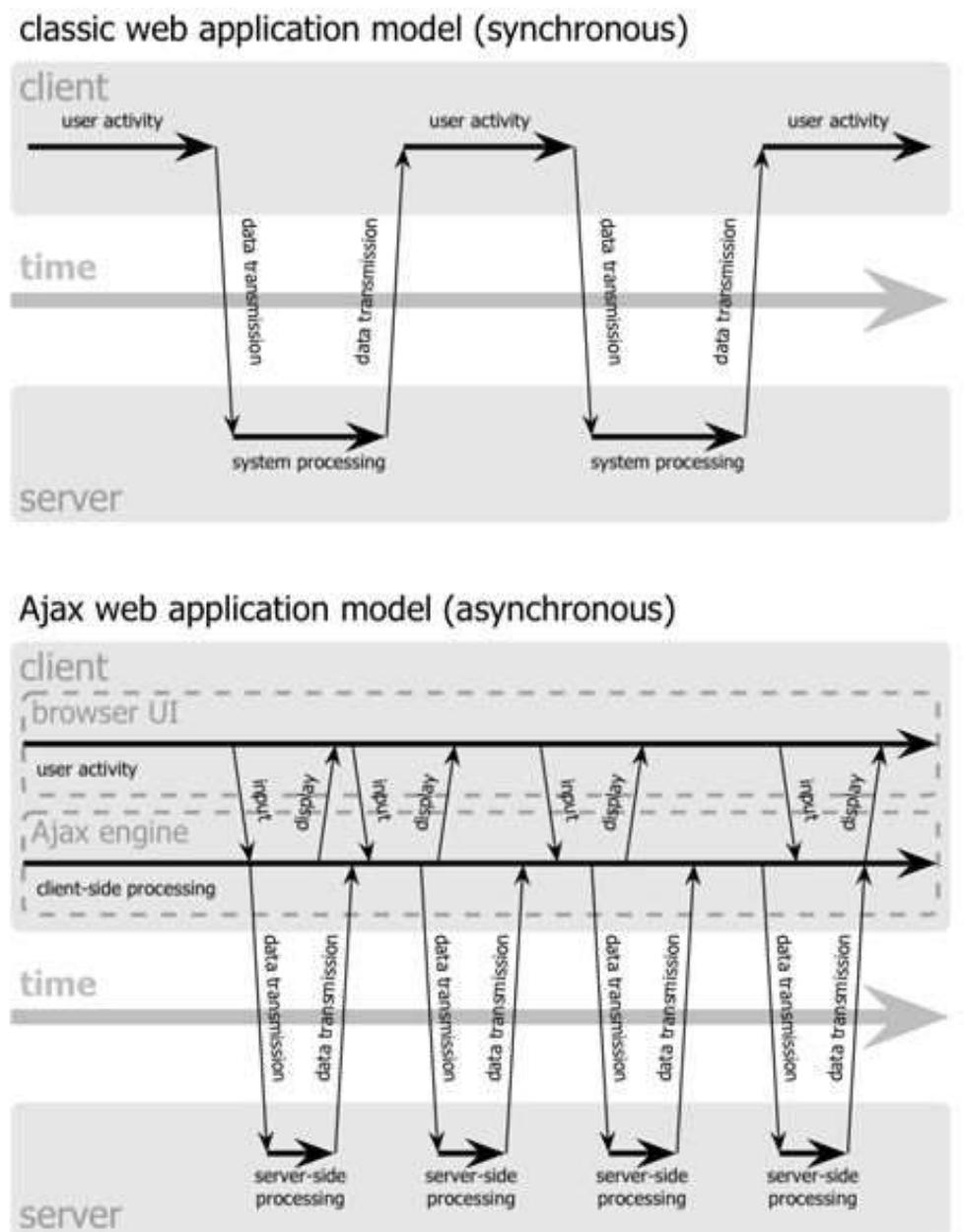
# AJAX



# AJAX



# AJAX



# AJAX

## L'objet XMLHttpRequest (1 / 3)

Permet de contrôler les transactions HTTP au moyen de langages de programmation côté client (principalement JavaScript).

Pour supporter la communication asynchrone entre navigateur et serveur, il autorise:

l'enregistrement de fonctions de rappel: invoquées pour chaque changement d'état transactionnel.

l'accès à tous les champs d'en-tête (requête et réponse).

Il fournit toutes les méthodes et propriétés pour une communication asynchrone.

# AJAX

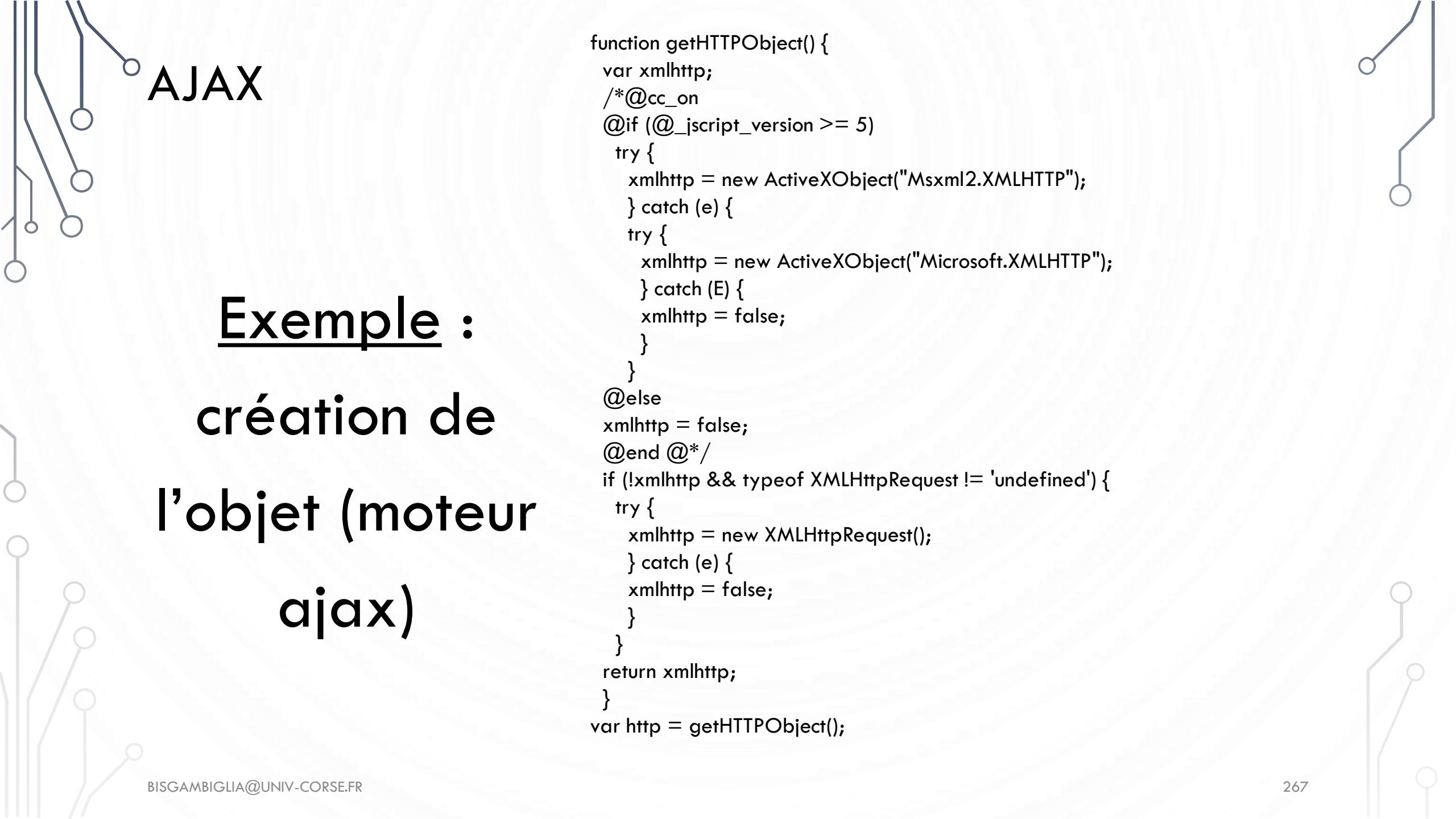
## L'objet XMLHttpRequest (2/3)

Méthodes de l'objet XMLHttpRequest	
Méthode	Description
open()	Met fin à la requête en cours et en prépare une nouvelle, en indiquant la méthode (GET ou POST) et l'URL.
send()	Lance la requête
abort()	Met fin à la requête en cours
setRequestHeader()	Assigne un couple nom/valeur à l'en-tête accompagnant la requête
getResponseHeader()	Récupère la valeur d'une chaîne de l'en-tête de réponse
getAllResponseHeader()	Récupère l'ensemble des en-têtes de réponse

# AJAX

## L'objet XMLHttpRequest (3/3)

Propriétés de l'objet XMLHttpRequest	
Propriété	Description
status	Code renvoyé par le serveur (exemple, 200 pour OK ou 404 pour un fichier introuvable)
statusText	La chaîne accompagnant le code
readyState	Un entier indiquant l'état de l'objet. Peut prendre 5 valeurs: 0 = non initialisé 1 = en cours de chargement 2 = chargé 3 = interaction 4 = terminé
onreadystatechange	Gestionnaire d'événement pour chaque changement d'état de l'objet
responseText	Chaîne correspondant à la réponse du serveur à la requête
responseXML	Version XML de la chaîne de réponse



# AJAX

## Exemple : création de l'objet (moteur ajax)

```
function getHTTPObject() {  
    var xmlhttp;  
    /*@cc_on  
    @if (@_jscript_version >= 5)  
        try {  
            xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");  
        } catch (e) {  
            try {  
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  
            } catch (E) {  
                xmlhttp = false;  
            }  
        }  
    @else  
        xmlhttp = false;  
    @end @*/  
    if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {  
        try {  
            xmlhttp = new XMLHttpRequest();  
        } catch (e) {  
            xmlhttp = false;  
        }  
    }  
    return xmlhttp;  
}  
var http = getHTTPObject();
```

# AJAX

## Exemple : partie PHP

```
<?php  
$recherche=$_GET['recherche'];  
  
$oDb = mysqli_connect('localhost', 'root', "");  
$oSdb = mysqli_select_db($oDb,'testmovie');  
  
$req = "SELECT FName, LName FROM Actor  
WHERE LName LIKE ' ".$recherche." ';"  
$res = mysqli_query($req);  
$row = mysqli_fetch_row($res);  
if ($row) {  
    echo "$row[0],$row[1]";  
}
```

# AJAX

## Exemple :

### partie HTML

```
<form action="get">  
    Recherche rapide par ID:  
    <input type="text" name="rapide"  
          id="rapide" onkeypress="majActeur();"/>  
    />  
  
    Nom:  
    <input type="text" name="nom" id="nom"  
          /><br />  
  
    Prénom:  
    <input type="text" name="prenom"  
          id="prenom" /><br />  
  
</form>
```

# AJAX

## Exemple : partie JS

```
function lancement()
{
    var rechRapide = document.getElementById("rapide").value;
    http.open("GET", "ajax.php?recherche="+escape(rechRapide),
    true);
    http.onreadystatechange = handleHttpResponse;
    http.send(null);
}

function handleHttpResponse()
{
    if (http.readyState == 4)
    {
        results = http.responseText.split(",");
        document.getElementById('prenom').value =
        results[0];
        document.getElementById('nom').value = results[1];
    }
}
```



AJAX

AJAX

avec

iQuery



# AJAX & JQUERY

jQuery :

```
$(document).event(function(){  
    $("element").event(function(){  
        $("element").function();  
    });  
});
```

AJAX :

1. html
2. moteur Ajax (JS)
3. PHP & BD

# AJAX & JQUERY

```
$(selector).load(URL,data,callback);
```

## Méthode « load » :

- URL : l'adresse de la ressource
- data : (optionnel) les données à envoyer
- callback : la fonction à exécuter si la fonction load a bien marché

# AJAX & JQUERY

Code :

[http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery\\_ajax\\_load](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax_load)

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("demo_test.txt");
    });
});
</script>
</head>
<body>
<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>
<button>Get External Content</button>
</body>
</html>
```

# AJAX & JQUERY

```
$(selector).load(URL,data,callback(responseTxt, statusTxt, xhr));
```

**callback : paramètres optionnels de la fonction de callback :**

- **responseTxt** : variable qui contient le résultats
- **statusTxt** : statut de la requête
- **xhr** : l'objet XMLHttpRequest

# AJAX & JQUERY

Code :

[http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery\\_ajax\\_load\\_callback](http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax_load_callback)

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
            if(statusTxt == "success")
                alert("External content loaded successfully!");
            if(statusTxt == "error")
                alert("Error: " + xhr.status + ": " + xhr.statusText);
        });
    });
});
</script>
</head>
<body>


BISGAMBIGLIA@UNIV-CORSE.FR



276


```

# AJAX & JQUERY

```
$.get(URL,callback);
```

Méthodes GET et POST

```
$.post(URL,data,callback);
```

# AJAX & JQUERY

Exemple méthode GET :

Fichier PHP :

```
<?php  
echo "hello world";
```

```
$(document).ready(function(){  
    $("button").click(function(){  
        $.get("demo_test.php", function(data,  
status){  
            alert("Data: " + data + "\nStatus: " +  
status);  
        });  
    });  
});
```

# AJAX & JQUERY

Exemple méthode POST :

Fichier PHP :

```
<?php  
    echo "hello ".  
    $_POST['name'];
```

```
$(document).ready(function(){  
    $("button").click(function(){  
        $.post("demo_test_post.php",  
        {  
            name: "PA Bisgambiglia",  
            city: "Corte"  
        },  
        function(data,status){  
            alert("Data: " + data + "\nStatus: " + status);  
        });  
    });  
});
```

# FETCH API

- [https://developer.mozilla.org/fr/docs/Web/API/Fetch\\_API/Using\\_Fetch](https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch)
- L'API Fetch vient remplacer XMLHttpRequest.

```
const myImage = document.querySelector('img');

fetch('flowers.jpg')

    .then(function(response) { return response.blob(); })

    .then(function(myBlob) { const objectURL =
URL.createObjectURL(myBlob); myImage.src = objectURL; });

});
```

# POO EN PHP

## DÉcrire un objet



# PHP ET POO

- En PHP 5, il y a un nouveau model objet. La gestion des objets a été complètement réécrite, permettant de meilleures performances ainsi que plus de fonctionnalités.
- *Chaque définition de classe commence par le mot-clé **class** , suivi par le nom de la classe, qui peut être quelconque à condition que ce ne soit pas un mot réservé du PHP. Suivent une paire d'accolade contenant la définition des membres et des méthodes. Une pseudo-variable **\$this** est disponible lorsqu'une méthode est appelée depuis un contexte objet.*

# PHP

- **Définition simple d'une classe :**

```
<?php  
class SimpleClass  
{  
    // déclaration d'un membre  
    public $var = 'une valeur par défaut';  
  
    // déclaration de la méthode  
    public function displayVar() {  
        echo $this->var;  
    }  
}  
?>
```

# PHP

- **Le mot clé new :**

- Pour créer une instance d'un objet, un nouvel objet doit être créé et assigné à une variable. Un objet doit toujours être assigné lors de la création d'un nouvel objet à moins qu'un l'objet ait un constructeur défini qui lance un exception en cas d'erreur. Création d'une instance :

```
<?php  
$instance = new SimpleClass();  
?>
```

- **Le mot clé extends**

- Une classe peut **hériter** des méthodes et des membres d'une autre classe en utilisant le mot clé **extends** dans la déclaration. Il n'est pas possible d'étendre de multiples classes, une classe peut uniquement hériter d'une seule classe de base.
- Les méthodes et membres hérités peuvent être **surchargés**, à moins que la classe parent ait défini une méthode comme **final**. Pour surcharger, il suffit de redéclarer la méthode avec le même nom que celui défini dans la classe parent. Il est possible d'accéder à une méthode ou un membre surchargé avec l'opérateur **parent::**

# PHP

- **Le mot clé extends**

```
<?php  
class SimpleClass  
{  
    // déclaration d'un membre  
    public $var = 'une valeur par  
    défaut';  
    // déclaration de la méthode  
    public function displayVar() {  
        echo $this->var;  
    }  
}
```

```
// extension de la classe  
class ExtendClass extends SimpleClass  
{  
    // Redéfinition de la méthode parent  
    function displayVar()  
    {  
        echo "Classe étendue\n";  
        parent::displayVar();  
    }  
}  
$extended = new ExtendClass();  
$extended->displayVar();  
?>
```

# PHP

- **Constructeurs** : `void __construct ( mixed args , ... )`
  - Les classes qui possèdent une méthode constructeur appellent cette méthode à chaque création d'une nouvelle instance de l'objet.
  - Note : Les constructeurs parents ne sont pas appelés implicitement si la classe enfant définit un constructeur. Si vous voulez utiliser un constructeur parent, il sera nécessaire de faire appel à `parent::__construct()`.
- **Exemple** :

```
<?php  
class BaseClass {  
    function __construct() {  
        print "In BaseClass constructor\n";  
    }  
}
```

```
class SubClass extends BaseClass {  
    function __construct() {  
        parent::__construct();  
        print "In SubClass constructor\n";  
    }  
}  
$obj = new BaseClass();  
$obj = new SubClass();  
?>
```

# PHP

- **Destructeurs : void destruct()**

- PHP 5 introduit un concept de destructeur similaire aux autres langages orientés objet, comme le C++ . La méthode destructeur doit être appelée aussitôt que toutes les références à un objet particulier sont effacées ou lorsque l'objet est explicitement détruit. Exemple avec un Destructeur

```
<?php  
  
class MyDestructableClass {  
  
    function __construct() {  
        print "In constructor\n";  
  
        $this->name = "MyDestructableClass";  
    }  
  
    function __destruct() {  
        print "Destruction de " . $this->name . "\n";  
    } } $obj = new MyDestructableClass(); ?>
```

# PHP

- **Visibilité** : la visibilité d'une propriété ou d'une méthode peut être définie en prefixant la déclaration avec un mot-clé : **public**, **protected** ou **private**.
- Les éléments déclarés publics (**public**) peuvent être utilisés par n'importe quelle partie du programme.
- L'accès aux éléments protégés (**protected**) est limité aux classes et parents hérités.
- L'accès aux éléments privés (**private**) est uniquement réservé à la classe qui les a définis.

```
<?php
/**
 * Définition de MyClass
 */
class MyClass
{
    public $public = 'Public';
    protected $protected = 'Protected';
    private $private = 'Private';

    function printHello()
    {
        echo $this->private;
        echo $this->protected;
        echo $this->private;
    }
}
```

- **L'opérateur de résolution de portée (::) :**

- Il fournit un moyen d'accéder aux membres statiques ou constants ainsi qu'aux éléments redéfinis par la classe.
- Lorsque vous référez ces éléments en dehors de la définition de la classe, utilisez le nom de la classe.

```
<?php
class MyClass {
    const CONST_VALUE = 'Une valeur
constante';
}

echo MyClass::CONST_VALUE;
?>
```

# PHP

- **Self et parent :**

- Deux mots-clé spéciaux, `self` et `parent`, sont utilisés pour accéder aux membres ou aux méthodes depuis la définition de la classe.

```
<?php
class MyClass {
    const CONST_VALUE = 'Une
valeur constante';
}

echo MyClass::CONST_VALUE;
?>
```

```
:: depuis la définition de la classe
<?php
class OtherClass extends MyClass
{
    public static $my_static = 'variable statique';

    public static function doubleColon() {
        echo parent::CONST_VALUE . "\n";
        echo self::$my_static . "\n";
    }
}
OtherClass::doubleColon();
?>
```

- **Statische :**

- Le fait de déclarer des membres ou des méthodes comme statiques vous permet d'y accéder sans avoir besoin d'instancier la classe. Un membre déclaré comme statique ne peut être accédé avec l'objet instancié d'une classe (bien qu'une méthode statique le peut).
- La déclaration **static** doit être faite après la déclaration de visibilité. Pour des raisons de compatibilité avec PHP 4, si aucune déclaration de visibilité n'est utilisée, alors le membre ou la méthode sera traité comme s'il avait été déclaré comme public.
- Comme les méthodes statiques sont appelables sans instance d'objet créée, la pseudo variable `$this` n'est pas disponible dans la méthode déclarée en tant que statique.

# PHP

- **Statische :**

- En fait, les appels de méthodes statiques sont résolus au moment de la compilation. Lorsque l'on utilise un nom de classe explicite, la méthode est déjà identifiée complètement et aucune notion d'héritage n'est appliquée. Si l'appel est effectué par le mot clé `self` , alors `self` est traduit en la classe courante, qui est la classe appartenant au code. Ici aussi, aucune notion d'héritage n'est appliquée.
- On ne peut pas accéder à des propriétés statiques à travers l'objet en utilisant l'opérateur `->` .

# PHP

Exemple avec un membre statique

```
<?php  
class Foo  
{  
    public static $my_static = 'foo';  
    public function staticValue() {  
        return self::$my_static;  
    }  
}  
  
class Bar extends Foo  
{  
    public function fooStatic() {  
        return parent::$my_static;  
    }  
}
```

```
}  
}  
  
print Foo::$my_static . "\n";  
  
$foo = new Foo();  
print $foo->staticValue() . "\n";  
print $foo->my_static . "\n";  
// propriété my_static non définie  
// $foo::my_static n'est pas possible  
  
print Bar::$my_static . "\n";  
$bar = new Bar();  
print $bar->fooStatic() . "\n";  
?>
```

# PHP

- Constantes de classe :
- Il est possible de définir des valeurs constantes à l'intérieur d'une classe, qui ne seront pas modifiables. Les constantes diffèrent des variables normales du fait qu'on n'utilise pas le symbole \$ pour les déclarer ou les utiliser. Tout comme pour les membres statiques , on ne peut pas accéder aux valeurs constantes depuis une instance de l'objet (en utilisant \$object::constant ).

```
<?php  
class MyClass  
{  
    const constant = 'valeur constante';  
  
    function showConstant() {  
        echo self::constant . "\n";  
    }  
}  
  
echo MyClass::constant . "\n";  
  
$class = new MyClass();  
$class->showConstant();  
  
// echo $class::constant; n'est pas autorisé  
?>
```

```
<?php  
abstract class AbstractClass  
...
```

- **Abstraction de classes**

- PHP 5 introduit les classes et les méthodes abstraites. Il n'est pas autorisé de créer une instance d'une classe définie comme abstraite. Toutes les classes contenant au moins une méthode abstraite doivent également être abstraites. Pour définir une méthode abstraite, il faut simplement déclarer la signature de la méthode et ne fournir aucune implémentation.
- Lors de l'héritage depuis une classe abstraite, toutes les méthodes marquées comme abstraites dans la déclaration de la classe parent doivent être définies par l'enfant ; de plus, ces méthodes doivent être définies avec la même (ou plus faible) visibilité . Par exemple, si la méthode abstraite est définie comme protégée, l'implémentation de la fonction doit être définie en tant que protégée ou publique.

# PHP

- **Interfaces**

- Les interfaces objet vous permettent de créer du code qui spécifie quelles méthodes et variables une classe peut implémenter, sans avoir à définir comment ces méthodes seront gérées.
- Les interfaces sont définies en utilisant le mot clé `interface`, de la même façon qu'une classe standard mais sans aucun contenu de méthode.
- Toutes les méthodes déclarées dans une interface doivent être publiques.

- **implements**

- Pour implémenter une interface, l'opérateur `implements` est utilisé. Toutes les méthodes de l'interface doivent être implementées dans une classe ; si ce n'est pas le cas, une erreur fatale sera émise. Les classes peuvent implémenter plus d'une interface en séparant chaque interface par une virgule.

# PHP

- Interfaces

```
<?php
// Declaration de l'interface 'iTemplate'
interface iTemplate
{
    public function setVariable($name,
$var);
    public function getHtml($template);
}
```

```
// Implémentation de l'interface
// Ceci va fonctionner
class Template implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }

    public function getHtml($template)
    {
        foreach($this->vars as $name => $value)
        {
            $template = str_replace('{'. $name .
'}', $value, $template);
        }
        return $template;
    }
}
```

- **Surcharge**

- Les appels de méthodes et l'accès aux membres peuvent être surchargés via les méthodes `__call`, `__get` et `__set`. Ces méthodes ne seront déclenchées que si votre objet, hérité ou non, ne contient pas le membre ou la méthode auquel vous tentez d'accéder. Toutes les méthodes surchargées doivent être définies en tant que public .
- Depuis PHP 5.1.0, il est également possible de surcharger les fonctions `isset` et `unset` via, respectivement, les méthodes `__isset` et `__unset`.

- **Mot clé 'final'**

- PHP 5 introduit le mot-clé " final " qui empêche les classes filles de surcharger une méthode en en préfixant la définition par le mot-clé " final ". Si la classe elle-même est définie comme finale, elle ne pourra pas être étendue.

- **Parcours d'objets**

- PHP 5 fournit une façon de définir les objets de manière à ce qu'on puisse parcourir une liste de membres avec une structure `foreach`. Par défaut, toutes les propriétés visibles seront utilisées pour le parcours.

```
$class = new MyClass();

foreach($class as $key => $value) {
    print "$key => $value\n";
}
```

```
<?php
class MyIterator implements Iterator
```

- Comme nous le montre l'affichage, *l'itération foreach* affiche toutes les variables visibles disponibles. Pour aller plus loin, vous pouvez implémenter l' interface interne de PHP 5 nommée `Iterator` . Ceci permet de déterminer comment l'objet doit être parcouru.

- **Méthodes magiques**

- Les noms de fonction `__construct` , `__destruct`, `__call` , `__get` , `__set` , `__isset` , `__unset`, `__sleep` , `__wakeup` , `__toString` , `__set_state` , `__clone` et `__autoload` sont magiques dans les classes PHP. Vous ne pouvez pas utiliser ces noms de fonction dans aucune de vos classes sauf si vous voulez modifier le comportement associé à ces fonctions magiques.
- Le but de `__sleep` est de clore toutes les connexions aux bases de données que l'objet peut avoir, valider les données en attente ou effectuer des tâches de nettoyage.
- Le but de `__wakeup` est de rétablir toute connexion base de données qui aurait été perdue durant la linéarisation et d'effectuer des tâches de réinitialisation.
- La méthode `__toString` détermine comment la classe doit réagir lorsqu'elle est convertie en chaîne de caractères

- **Auto-chargement de classes**

- De nombreux développeurs qui créent des applications orientées objet, créent un fichier source par définition de classe. L'inconvénient majeur de cette méthode est d'avoir à écrire une longue liste d'inclusions de fichier classes au début de chaque script : une inclusion par classe.
- En PHP 5, ce n'est plus nécessaire. Vous pouvez définir la fonction **`__autoload`** qui va automatiquement être appelée si une classe n'est pas encore définie au moment de son utilisation. Grâce à elle, vous avez une dernière chance pour inclure une définition de classe, avant que PHP ne déclare une erreur.
- Note : Les exceptions lancées depuis la fonction **`__autoload`** ne peuvent être interceptées par un bloc `catch` : elles provoqueront une erreur fatale.

# PHP

- Class MaClasse [extends SuperClass]

```
function MaClasse(parametre1, parametre2) {  
    this.attributPublic = parametre1;  
    var attributPrive = parametre2;  
    this.methodePublique = function() {}  
    var methodePrivee = function() {}  
}
```

- Instanciation

- var objetDeMaClasse = new MaClasse("valeur1", "valeur2");

- Appel des méthodes

- \$obj = new Classe(); //Instanciation de classe
  - \$obj->methodeInstance(); //Application de méthode

# PHP

```
class Livre extends Produit {  
    private $auteur;  
    private $editeur;  
    private $nb_pages;  
    function __construct($auteur, $editeur, $nb_pages) {  
        $this->auteur = $auteur;  
        $this->editeur = $editeur;  
        $this->nb_pages = $nb_pages; }  
    function changer_nb_pages($nb){...}  
}  
  
$livre = new Livre(...);  
$livre->changer_nb_pages(150) ;
```

Exemple de classe

- **Pattern**

- Les patterns sont un moyen de décrire les meilleures pratiques et les bonnes conceptions. Ils proposent une solution flexible aux problèmes habituels de programmation.

- Singleton (classe qui sera instanciée qu'une seule)  
<https://www.grafikart.fr/formations/programmation-objet-php/singleton>
- Factory (<https://openclassrooms.com/courses/programmez-en-orienté-objet-en-php/les-design-patterns>)
- Observer
- Strategy

# MVC

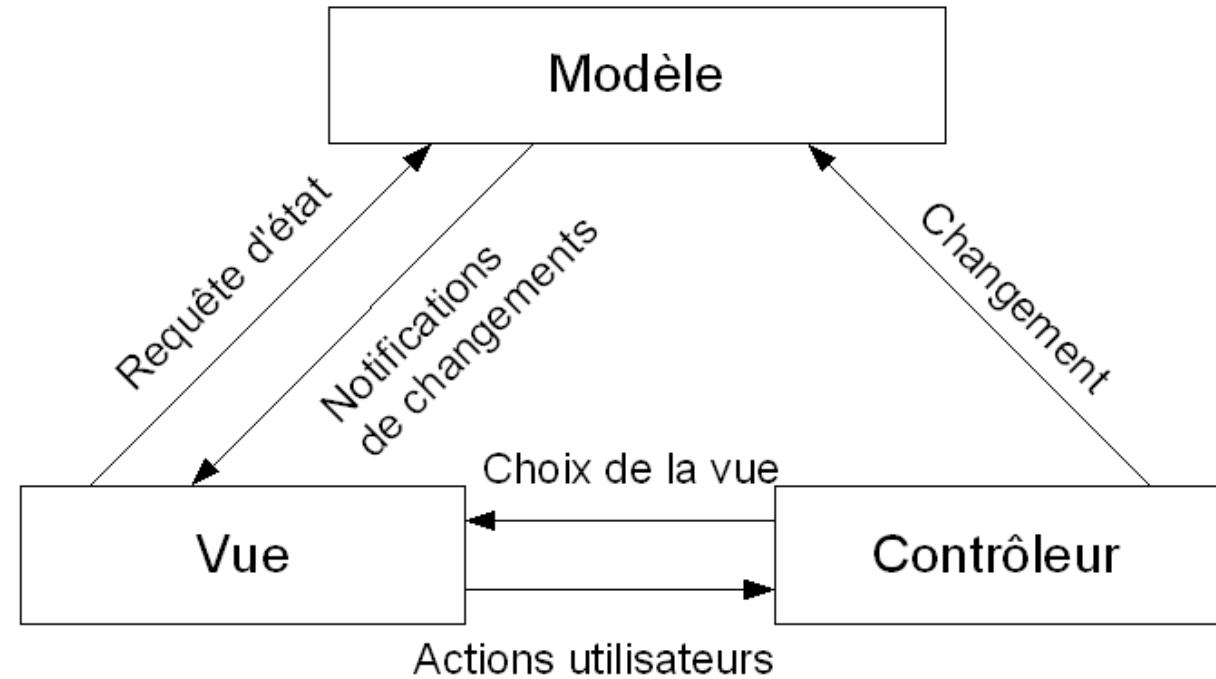
- Un design pattern ou modèle de conception répond à des problèmes d'ordre architectural au niveau des développements informatiques.
- Un Design pattern est intimement lié à la POO. Il consiste en un ensemble de règle de bonne pratique visant à améliorer l'organisation du projet en privilégiant la réutilisabilité du code. Une telle organisation permet de catalyser les temps de développements, et améliore considérablement la maintenance de l'application

# MVC

- Le pattern Modèle-Vue-Contrôleur organise l'interface Homme-machine d'une application logicielle en
  - un modèle (objet métier, modèle de données),
  - une vue (présentation, interface utilisateur) et
  - un contrôleur (logique de contrôle, gestion des événements, traitement),

# MVC

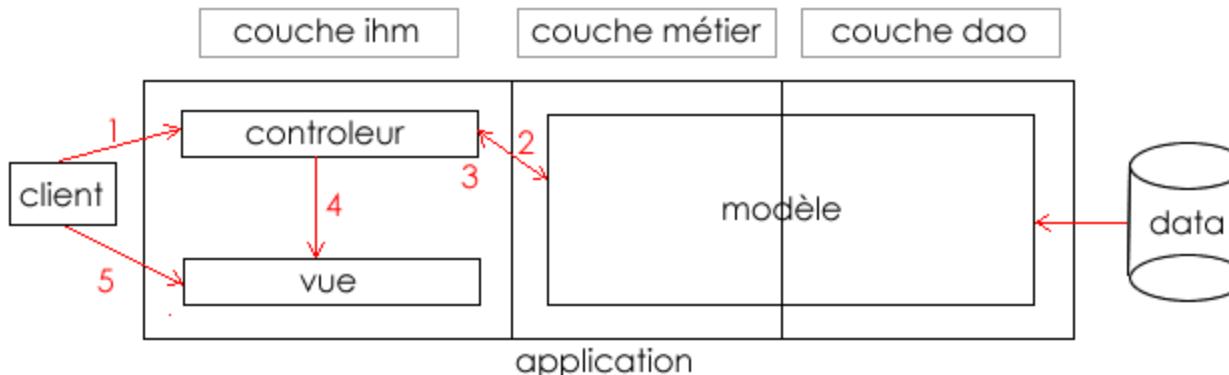
1. la requête est analysée par le contrôleur
2. le contrôleur demande au modèle approprié d'effectuer les traitements
3. le contrôleur renvoie la vue adaptée.



# MVC

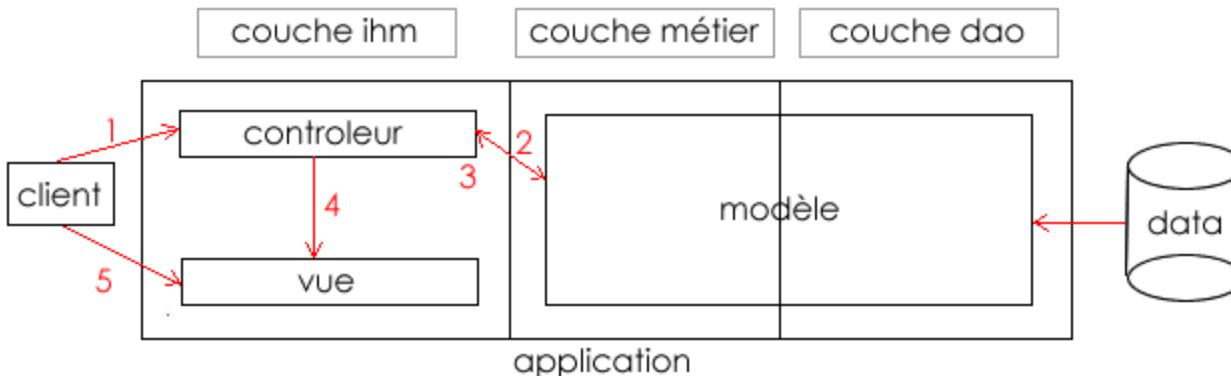
## MVC dans le contexte web

- MVC est bien adaptée à des applications web écrites avec des langages orientés objet, elles tirent ainsi le meilleur parti du PHP5 ou 7.



# MVC

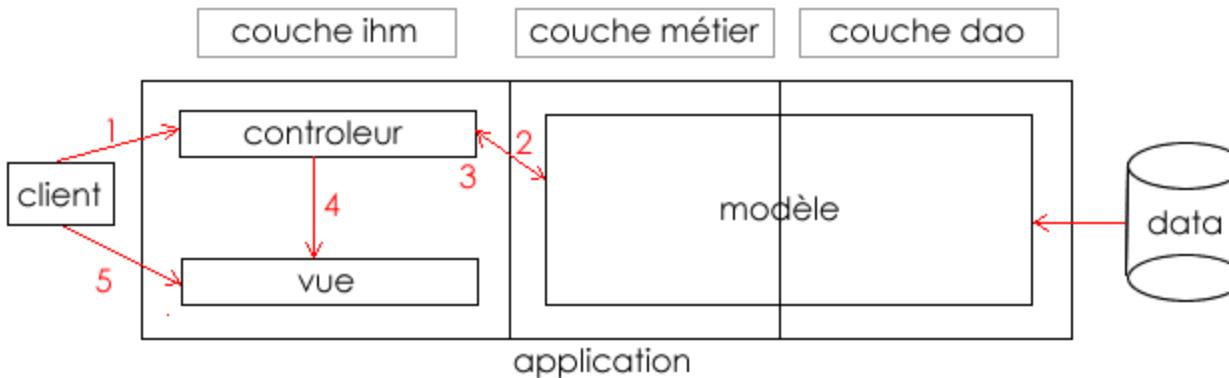
MVC dans le contexte web



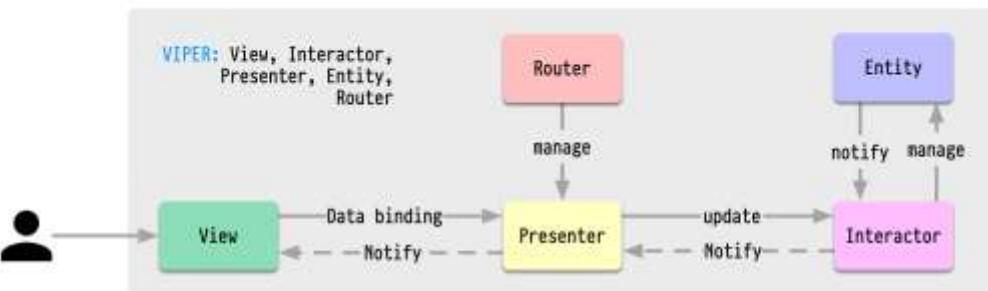
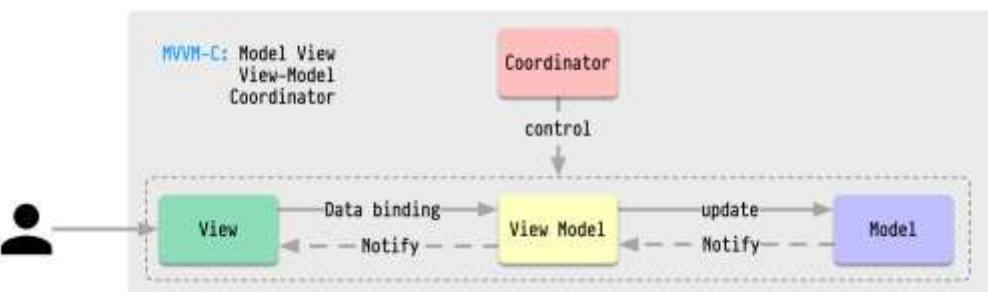
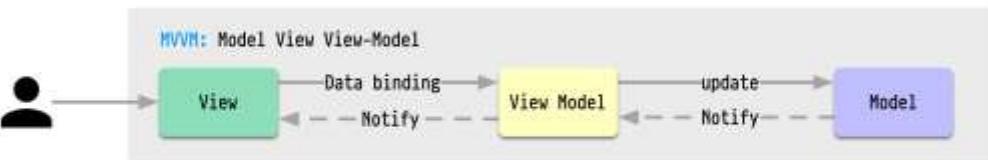
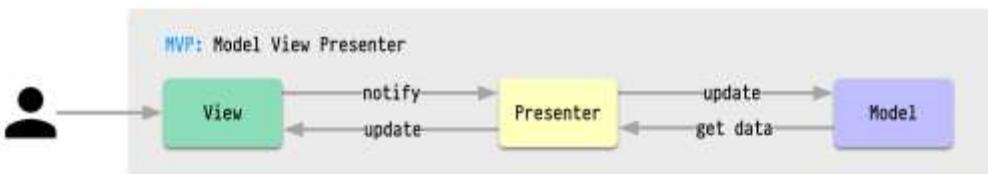
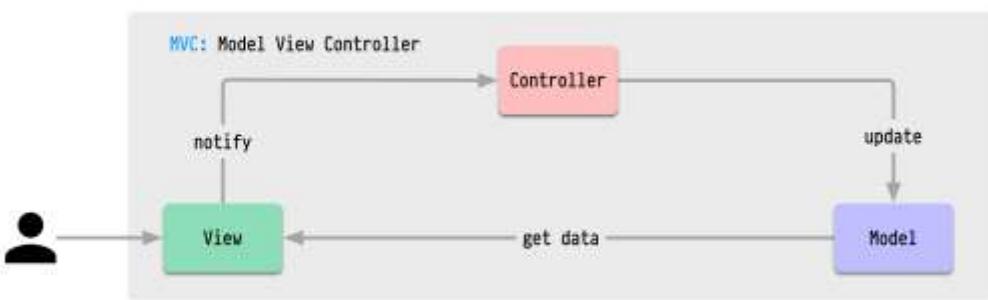
- couche ihm: c'est l'interface utilisateur encore appelé interface homme machine
- couche métier : c'est le cœur de l'application où réside les objets traités par l'application
- couche dao : couche d'accès aux données (data access object). Cette couche permet une indépendance de la logique métier et du stockage des données associées

# MVC

## MVC dans le contexte web



1. le client fait une demande au contrôleur. Ce contrôleur voit passer toutes les demandes des clients
2. le contrôleur traite la demande. Pour ce faire, il peut avoir besoin de l'aide de la couche métier, et éventuellement de la couche dao.
3. le contrôleur reçoit une réponse de la couche métier et effectue les actions demandées par l'utilisateur
4. le contrôleur sélectionne et nourrit une vue pour présenter les résultats de l'action qui vient d'être effectuée
5. la vue est enfin envoyée au client



# PHP ... BIEN CODER

- <http://www.arthurweill.fr/guide-des-bonnes-pratiques-php/>
  - Les noms des variables, des fonctions, des méthodes et des attributs sont écrits en camelCase.
  - Les noms des clés dans un tableau sont écrits en snake\_case.
  - Les noms des constantes sont écrits en MAJUSCULE
  - En SQL, les tables et les noms des colonnes sont écrits en snake\_case.
  - Les noms des objets sont écrits en PascalCase.
  - Chaque objet est stocké dans un fichier qui porte le même nom que l'objet et ne contient que cet objet.
- <https://blog.pascal-martin.fr/post/php71-fr-typage/>
  - Typer vos fonctions function func02(?int \$a, ?int \$b) : ?int {

# PHP ... BIEN CODER

- <https://www.php.net/manual/fr/control-structuresdeclare.php>
- declare (directive)

## Version

7.0.0

7.0.0

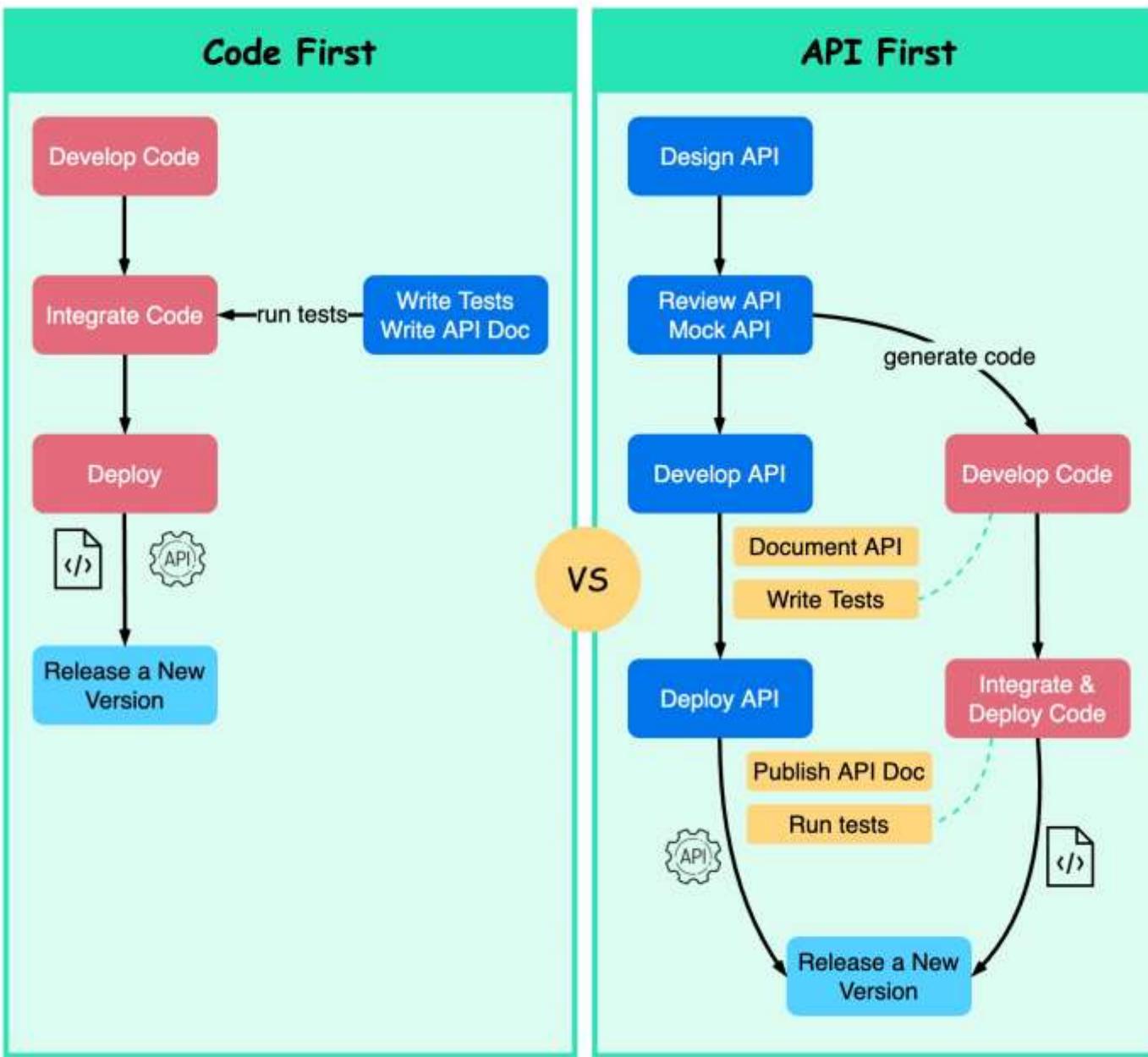
5.3.0

## Description

Ajout de la directive *strict\_types*

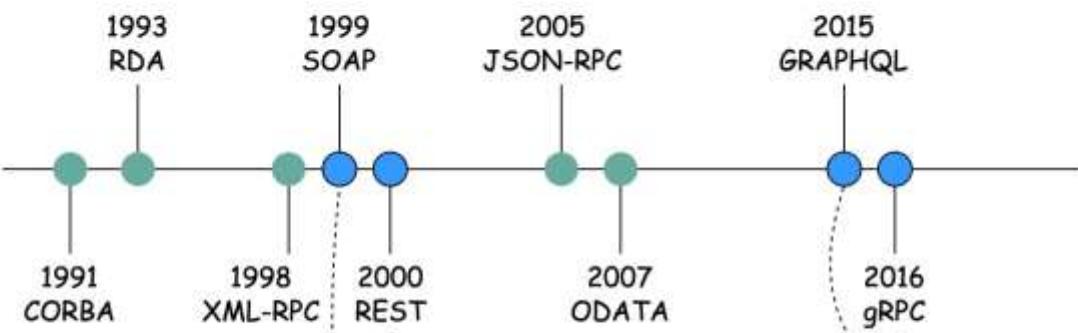
La directive *ticks* ne fuit plus dans des unités de compilation différentes.

Ajout de la directive *encoding*

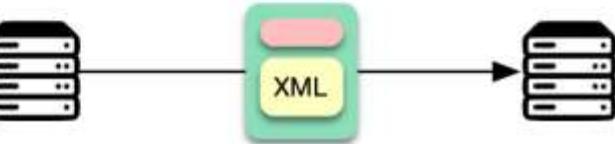
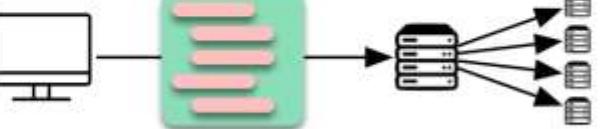
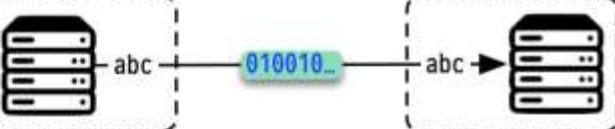
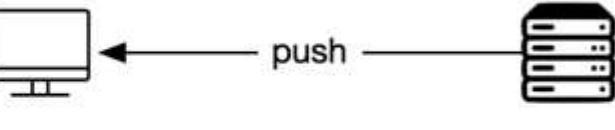
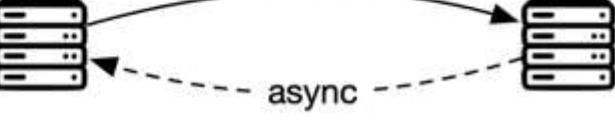


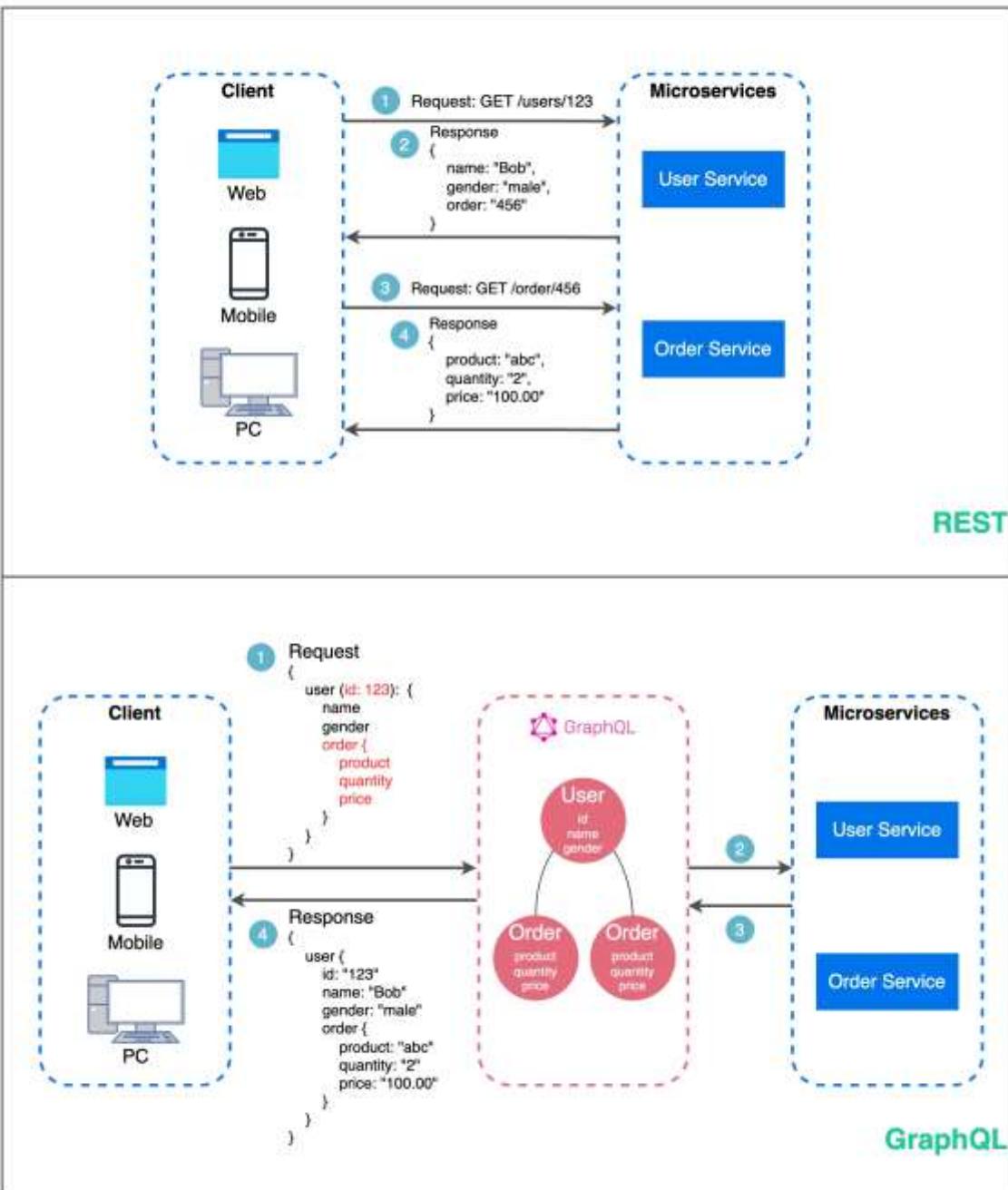
# API Architectural Styles Comparison

Source: altexsoft



	SOAP (Simple Object Access Protocol)	REST (REpresentational State Transfer)	GraphQL	RPC (Remote Procedure Call)
Organized in terms of	enveloped message structure	compliance with six architectural constraints	schema & type system	local procedure call
Format	XML only	XML, JSON, HTML, plain text	JSON	JSON, XML, Protobuf, Thrift, FlatBuffers
Learning curve	Difficult	Easy	Medium	Easy
Community	Small	Large	Growing	Large
Use cases	<ul style="list-style-type: none"><li>- payment gateways</li><li>- identity management</li><li>- CRM solutions</li><li>- financial and telecommunication services</li><li>- legacy system support</li></ul>	<ul style="list-style-type: none"><li>- public APIs</li><li>- simple resource-driven apps</li></ul>	<ul style="list-style-type: none"><li>- mobile APIs</li><li>- complex systems</li><li>- micro-services</li></ul>	<ul style="list-style-type: none"><li>- command and action-oriented APIs</li><li>- high performance communication in massive micro-services systems</li></ul>

Style	Illustration	Use Cases
SOAP		XML-based for enterprise applications
RESTful		Resource-based for web servers
GraphQL		Query language reduce network load
gRPC		High performance for microservices
WebSocket		Bi-directional for low-latency data exchange
Webhook		Asynchronous for event-driven application



# BILAN

- Les technos
- HTML
- CSS
- JS
- PHP
- AJAX
- JSON & XML

## Créer des sites web

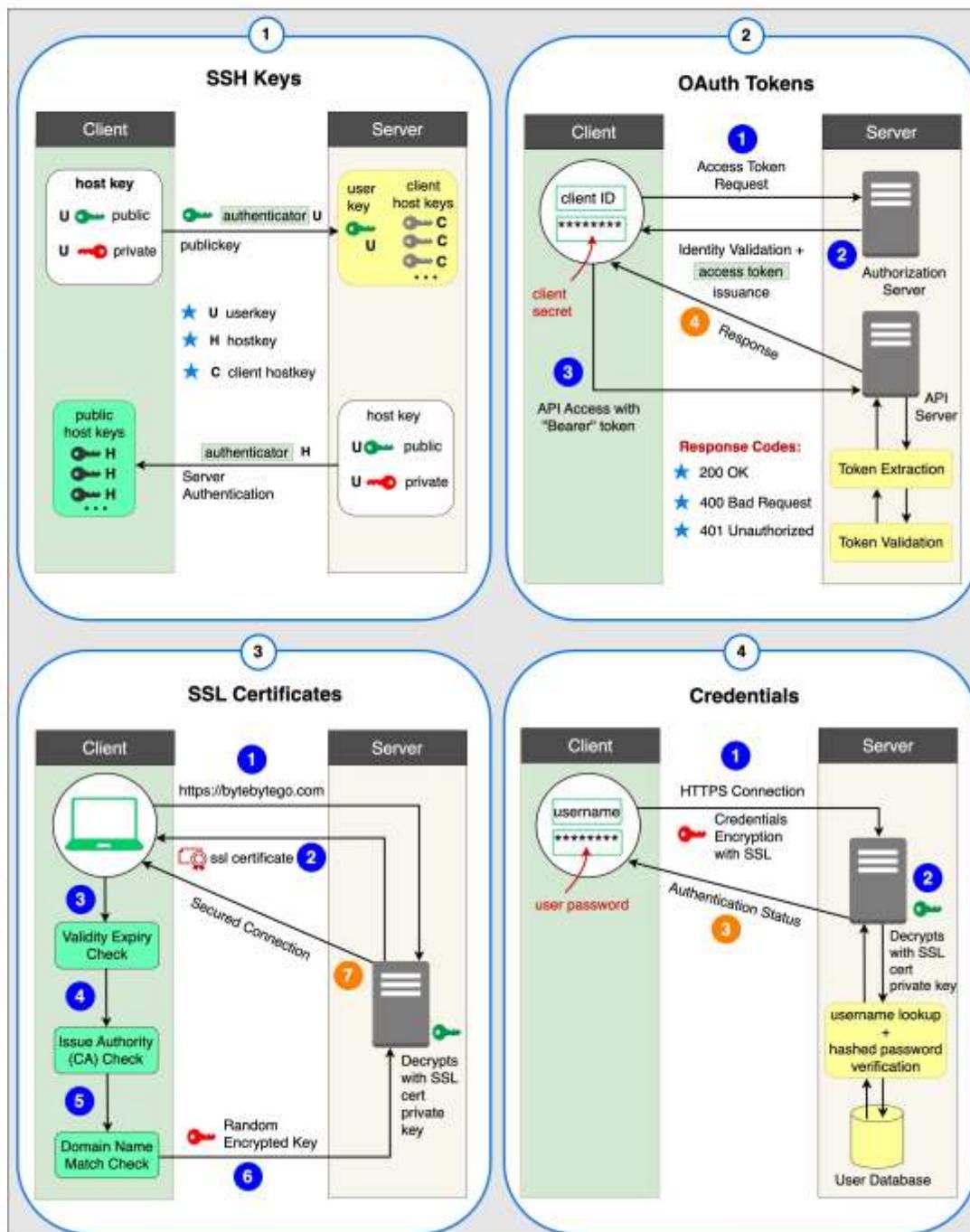
- Index
- Formulaires
- Partie admin
- Gestion des données
- Flux RSS
- Google map
- ...

**Podcast 17**

# SÉCURITÉ



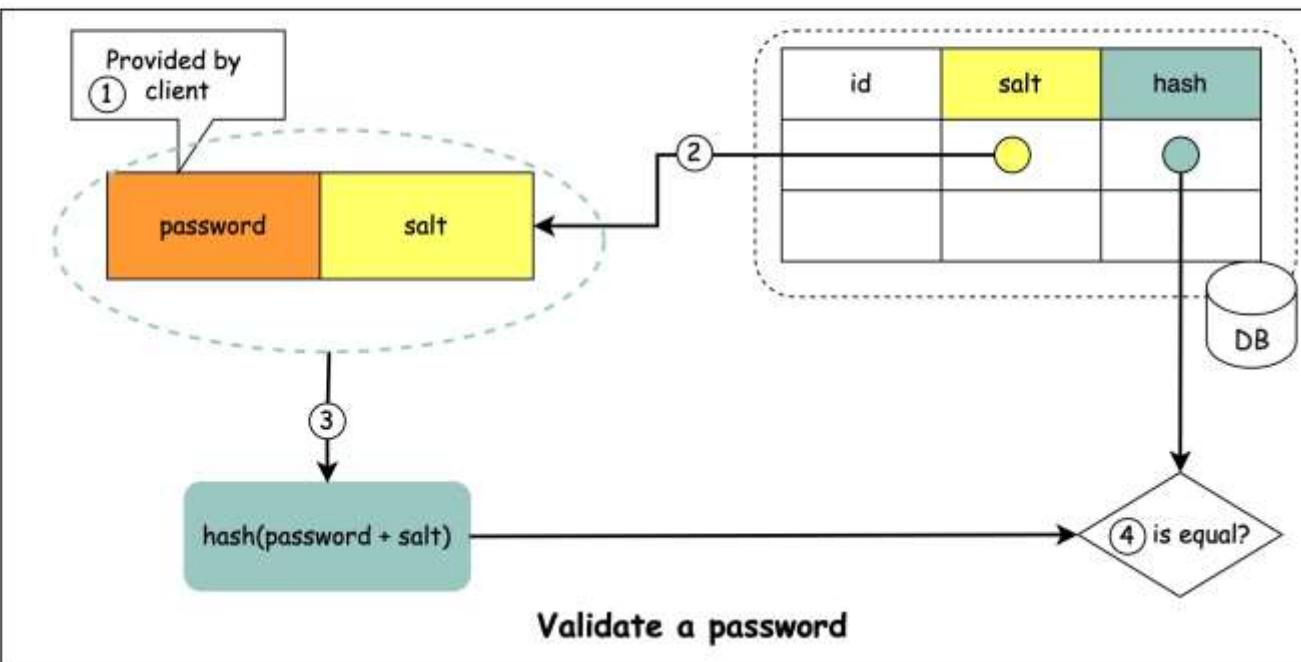
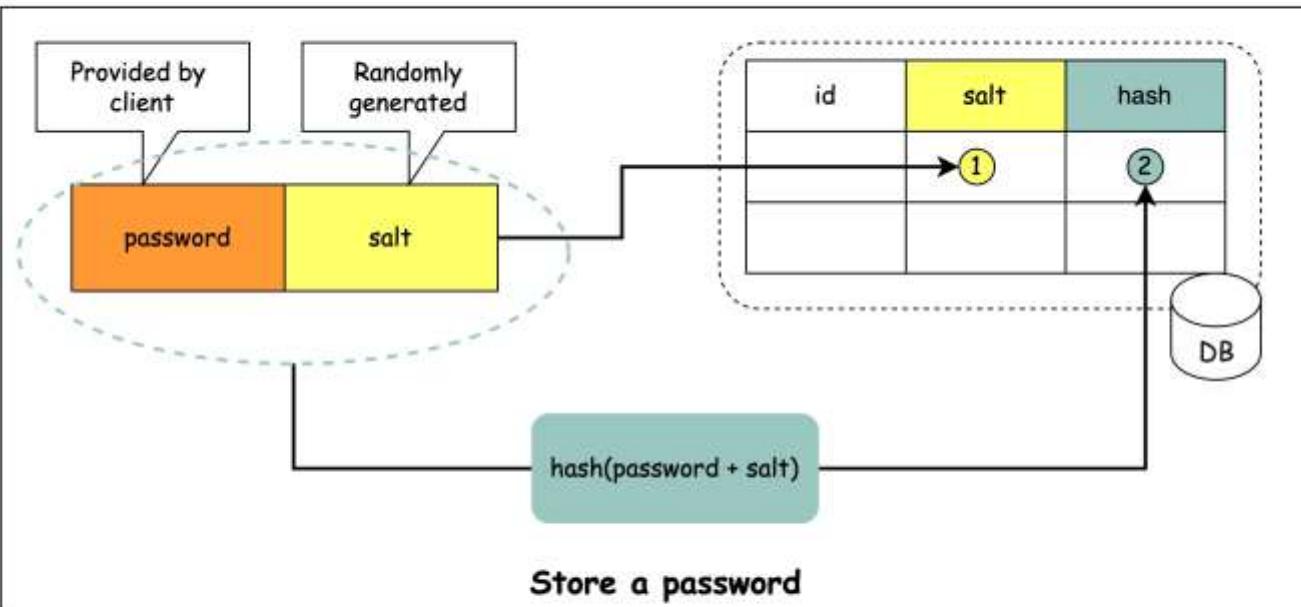
# SÉCURITÉ



# SÉCURITÉ

## How to store passwords in DB?

 blog.bytebytogo.com



# SOFT UTILIES

- [readme.so/fr](https://readme.so/fr) un générateur de Readme pour documenter les projets GitHub.
- [metatags.io](https://metatags.io) génère en 3 clics tous les meta tags de Social Preview pour votre html. Vous renseignez le titre, la description et une image.
- [remove.bg/fr](https://remove.bg/fr) outil pour supprimer le fond d'une image.
- [unsplash.com/developers](https://unsplash.com/developers) api pour générer des images aléatoires pour vos sites web.
- [smartmockups.com/fr/mockups/devices](https://smartmockups.com/fr/mockups/devices) Un site pour générer des mockups de vos sites web.
- [smalldev.tools](https://smalldev.tools) une librairie de 28 petits outils pour formater du JSON, minifier du CSS, générer des Lorem Ipsum



# DISCUSSION

BISGAMBIGLIA@UNIV-CORSE.FR

324

# LIENS



- <http://detailformation.com/marketing/20-ans-de-design-web-evolution-et-les-grandes-tendances/>
- <http://www.commentcamarche.net/faq/29726-les-differentes-proprietes-en-css3>
- <https://www.keacrea.com/blog/ergonomie-web-qu'est-ce-que-c-est>

# JAVASCRIPT

- K-NETWORK. Cours de JavaScript et DHTML. Disponible sur  
<http://www.editeurJavaScript.com/cours/>
- HONDERMARCK Olivier. Tutoriaux JavaScript et PHP. Disponible sur  
<http://www.toutJavaScript.com/savoir/savoir.php3>
- MARTIN-RABAUD Bernard. Aide JavaScript. Disponible sur  
<http://www.aideJavaScript.com/>
- D'autre exo à l'adresse :
  - [http://denis-orban.ifrance.com/20022003/html/p\\_mp/javascript.html](http://denis-orban.ifrance.com/20022003/html/p_mp/javascript.html)
- Pour en faire +
  - <http://www.toutjavascript.com/savoir/savoir12.php3>
  - [http://www.toutjavascript.com/savoir/savoir06\\_2.php3](http://www.toutjavascript.com/savoir/savoir06_2.php3)
- MVC
  - <http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/>
  - <http://notes.mazenod.fr/le-design-pattern-mvc1.html>

# ENCORE UN PEU DE COURS



Rapide introduction à nodejs !

<https://www.w3schools.com/nodejs/default.asp>

# NODE JS



- Node est un framework javascript :
  - qui permet de concevoir des applications rapides à partir d'un serveur http codé en javascript
  - opensource assez à la mode
  - Multiplateformes
- Première étape le téléchargement :
  - <https://nodejs.org/en/download/>

# NODE JS

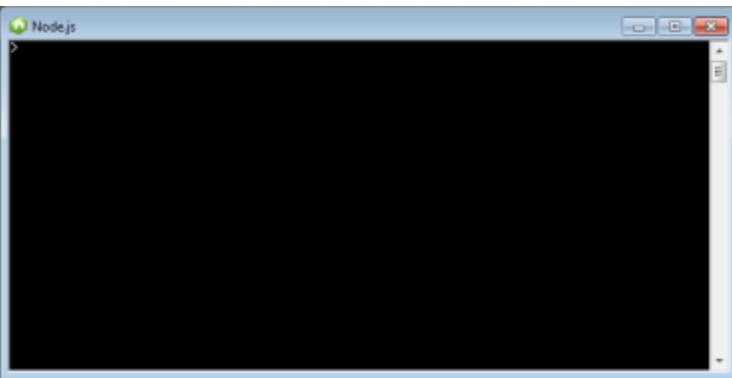
- Node.js est asynchrone!
- Voici comment le PHP ou l'ASP gère une demande de fichier :
  1. Envoie la tâche au système de fichiers de l'ordinateur.
  2. Attend l'ouvre du fichier et le lit
  3. Renvoie le contenu au client
  4. Prêt à gérer la prochaine demande.
- Voici comment Node.js gère une requête de fichier :
  1. Envoie la tâche au système de fichiers de l'ordinateur.
  2. Prêt à gérer la prochaine demande.
  3. Lorsque le système de fichiers a ouvert et lu le fichier, le serveur renvoie le contenu au client.
- Node.js élimine l'attente, et continue simplement avec la prochaine requête.
- Node.js exécute une programmation dite asynchrone, non bloquant, qui est très efficace en mémoire.

# NODE JS



- Avec node on peut :
  - Générer des pages html
  - Gérer des fichiers
  - Gérer des bases de données
  - Gérer des collections

# NODE JS



- Après l'installation

- Créer un fichier .js (exemple.js) dans C:\path\
- il faudra le lancer à partir de l'invite de commande de node
  - Exemple :
  - C:\path> node exemple.js
- Lancer votre navigateur à l'adresse
  - <http://localhost:8080>
- Votre serveur exécute votre code

# NODE JS

```
var http = require('http');
```

Un module eq. Bibliothèque

```
http.createServer(function (req, res) {
```

Création de l'objet http

```
    res.writeHead(200, {'Content-Type': 'text/plain'});
```

req argument passé par le client  
res argument renvoyé par le serveur ici  
Une entête de page avec du texte

```
    res.end('Hello my students!');
```

```
}).listen(8080);
```

# NODE JS

Faire et utiliser son module

Créer un dossier /test/

Créer un fichier  
monModule.js

Dans le dossier

```
exports.myDateTime = function () {  
    return Date();  
};
```

# NODE JS

Faire et utiliser son module  
Dans le dossier /test/  
Créer un fichier monServeur.js

```
var http = require('http');
var dt = require('./monModule');

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write("The date and time are currently: " +
dt.myDateTime());
    res.write("<br>")
    res.write(req.url);
    res.end();
}).listen(8888);
```

# NODE JS

Passer des paramètres  
en url

```
var http = require('http');
var url = require('url');
var dt = require('./monModule');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.write("<br>")
  // url à tester http://localhost:8888/?annee=2017&mois=Juin
  var q = url.parse(req.url, true).query;
  var txt = q.annee + " " + q.mois;
  res.end(txt);
}).listen(8888);
```

# NODE JS

## Gestion des fichiers

```
var fs = require('fs');
```

Avec ce module il est possible de lire, créer, modifier renommer et supprimer des fichiers.

- Créer un fichier html simple (test.html)

```
<html>
<body>
<h1>Mon titre</h1>
<p>mon texte</p>
</body></html>
```

# NODE JS

## Gestion des fichiers

```
var fs =  
require('fs');
```

```
var http = require('http');  
var url = require('url');  
var fs = require('fs');  
var dt = require('./monModule');  
  
http.createServer(function (req, res) {  
    fs.readFile('test.html', function(err, data) {  
        res.writeHead(200, {"Content-Type": "text/html"});  
        res.write(data);  
        res.write("The date and time are currently: " + dt.myDateTime());  
        res.write("<br>")  
        // url à tester http://localhost:8888/?annee=2017&mois=Juin  
        var q = url.parse(req.url, true).query;  
        var txt = q.annee + " " + q.mois;  
        res.write(txt);  
        res.write("<br>")  
        res.end();  
    });  
}).listen(8888);
```

# NODE JS



- Outil de débogage
  - `console.log(test!);`
- Outil de gestion de package
  - npm
  - Exemple
    - `C:\Users\Your Name>npm install upper-case`
- +++

# JSX

```
const name = 'Clarisse  
Agbegnenou';  
  
const element =  
<h1>Bonjour,  
{name}</h1>;  
  
ReactDOM.render(  
  element,  
  document.getElementById('root'));
```

<https://fr.reactjs.org/docs/introducing-jsx.html>

- JSX est une extension syntaxique de JavaScript pour React comme typescript pour angular.
- Au lieu de séparer artificiellement les *technologies* en mettant le balisage et la logique dans des fichiers séparés, React sépare les préoccupations via des unités faiblement couplées appelées « composants », qui contiennent les deux.

# Exemple à essayer de faire fonctionner

<https://fr.reactjs.org/docs/introducing-jsx.html>

```
function formatName(user) {  
  return user.firstName + ' ' + user.lastName;  
}  
  
const user = {  
  firstName: 'Harper',  
  lastName: 'Perez',  
};  
  
const element = <h1>Hello, {formatName(user)}!</h1>;  
  
ReactDOM.render(element, document.getElementById('root'));
```

# REACT

<https://fr.reactjs.org/docs/getting-started.html#try-react>

React en 1 min !

```
index.html
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8" />
5          <title>Add React in One Minute</title>
6      </head>
7      <body>
8
9          <h2>Add React in One Minute</h2>
10         <p>This page demonstrates using React with no build tooling.</p>
11         <p>React is loaded as a script tag.</p>
12
13         <!-- We will put our React component inside this div. -->
14         <div id="like_button_container"></div>
15
16         <!-- Load React. -->
17         <!-- Note: when deploying, replace "development.js" with "production.min.js". -->
18         <script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>
19         <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>
20
21         <!-- Load our React component. -->
22         <script src="like_button.js"></script>
23
24     </body>
25 </html>
```

# REACT

<https://fr.reactjs.org/docs/getting-started.html#try-react>

## like\_button.js

```
1  'use strict';
2
3  const e = React.createElement;
4
5  class LikeButton extends React.Component {
6    constructor(props) {
7      super(props);
8      this.state = { liked: false };
9    }
10
11   render() {
12     if (this.state.liked) {
13       return 'You liked this.';
14     }
15
16     return e(
17       'button',
18       { onClick: () => this.setState({ liked: true }) },
19       'Like'
20     );
21   }
22 }
23
24 const domContainer = document.querySelector('#like_button_container');
25 ReactDOM.render(e(LikeButton), domContainer);
```

React en 1 min !