

# Ajax

Développeur Web et Web Mobile

Janvier 2021  
Yendi REIVAX





# Sommaire



1. Ajax : *Qu'est ce que c'est ?*
2. Petit rappel JQuery
3. Petit rappel DOM
4. Ajax : *Quel sont les avantages à l'utiliser ?*
5. Ajax : *Dans quel cas l'utiliser ?*
6. Ajax : *Comment utiliser l'Ajax ?*



## Qu'est ce que c'est ?

L'Ajax est un moyen de communication **asynchrone** entre le **client** et le **serveur**.





## Oui mais encore ?

Ajax permet de *récupérer de la données* ou d'en *envoyer* sur un serveur sans pour autant *recharger toute la page*.





## Définition

Ajax est le nom de la technologie permettant au navigateur de lancer une requête au serveur en temps réel pour lire ou écrire des données sans recharger la page.

Ajax est également appelé **XHR** pour **XML HTTP Request**, même si on utilise en général le JSON. On parle de requêtes asynchrones car elles ne bloquent pas l'exécution du script pendant que les données transitent vers le serveur.

Ajax permet un web beaucoup plus dynamique, et des économies de ressources serveur.





## Scénario Ajax: Etape 1



**Client**



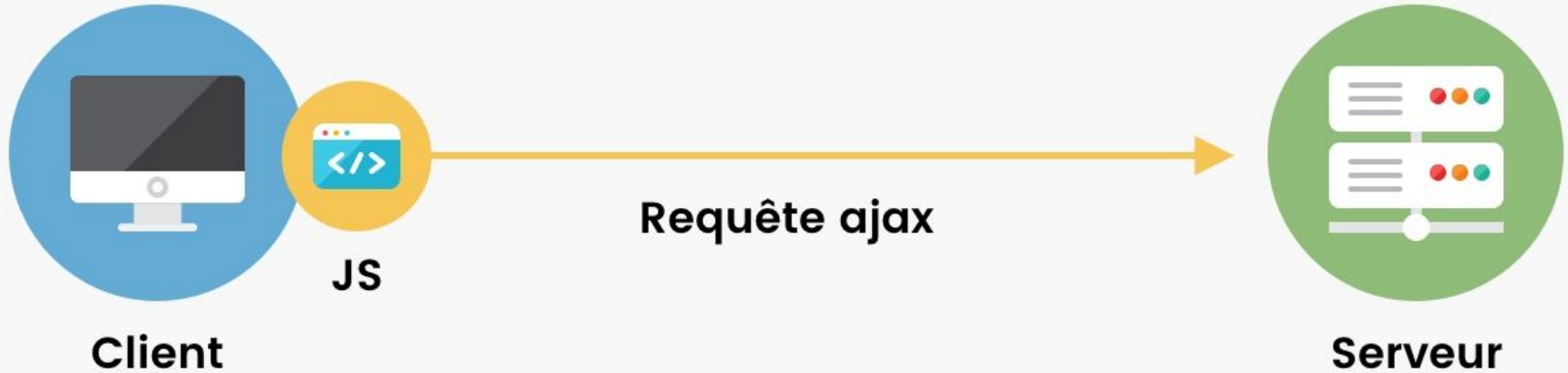
**Nouvelle page HTML**



**Serveur**



## Scénario Ajax: Etape 2





## Scénario Ajax: Etape 3





# Petit Rappel jQuery



# Sommaire



1. Comparatif JavaScript/jQuery
2. Les sélecteurs et les filtres jQuery
3. Gestion d'évènement

# jQuery VS JavaScript

*Qu'est ce que jQuery ?*

*Qu'est ce que JavaScript ?*

*Quelle est la différence entre les 2 ?*

*Peut on utiliser jQuery et JavaScript en même temps ?*



# jQuery VS Javascript

Tu préfères écrire **1 ligne** de code ou **5 lignes** de code ?

```
$("#span").addClass("bolder");
```

```
var listSpan = document.querySelectorAll('span');  
var i;  
  
for (i = 0; i < listSpan.length; i++) {  
    listItems[i].className = 'bolder';  
}
```

# Comparatif JQuery JavaScript

## jQuery

- Librairie(Bibliothèque) JavaScript
- Année de création 2006
- **Dynamiser et rendre des pages web interactives**
- Se libérer des spécificités des navigateurs
- Se base sur le système de sélecteur CSS

## JavaScript

- Langage de programmation
- Année de création 1995
- **Dynamiser et rendre des pages web interactives**

# Les sélecteurs et les filtres



# Les sélecteurs de base

En jQuery sélectionner un élément revient à l'identifier.

*“Allô centrale : Le suspect est une balise de type span avec une class bold”*

La sélection en jQuery se base sur le même principe que la sélection CSS.

Nous avons donc 3 types de sélection élémentaire

- Sélection par **type d'élément** (span, p, h1, h2, ...)
- Sélection par **classe** (.bold)
- Sélection par **id** (#header)



# Les sélecteurs par relation

Il arrive souvent que la sélection par type d'élément, par classe et par id ne soit pas suffisante pour sélectionner un ou plusieurs élément de la page

- **Relation descendante** `$("mère fille")` : sélectionne tous les éléments *filles* qui ont pour élément parent *mère*
- **Relation descendante direct** `$("mère > fille")` : sélectionne tous les éléments *filles* qui ont pour élément parent premier *mère*
- **Relation frère** `$("élément ~ frère")` : sélectionne tous les éléments *frères* qui sont au même niveau html *de l'élément*.





# Les filtres

Les système de sélecteurs bien que puissant atteint certaine limite dans des cas d'usage ou les relations entre les éléments ne nous intéressent pas.

jQuery offre alors une autre possibilité pour sélectionner des éléments , soit les filtres.

Les filtres eux s'intéressent à l'ordre de positionnement des élément HTML, leur indice de positionnement ou encore leur absence de sélecteur...

- `:first` : sélectionne la première occurrence d'un élément
- `:last` : sélectionne la dernière occurrence d'un élément
- `:eq(indice)` : sélectionne l'élément à un indice donné
- `:gt(indice)` : sélectionne le ou les élément(s) situé(s) à un indice supérieur au nombre spécifié.
- `:lt(indice)` : même concept mais pour les éléments à un indice plus petit que le nombre spécifié.
- `:not(sélecteur)` : sélectionne les éléments qui ne sont pas... eh bien, le sélecteur

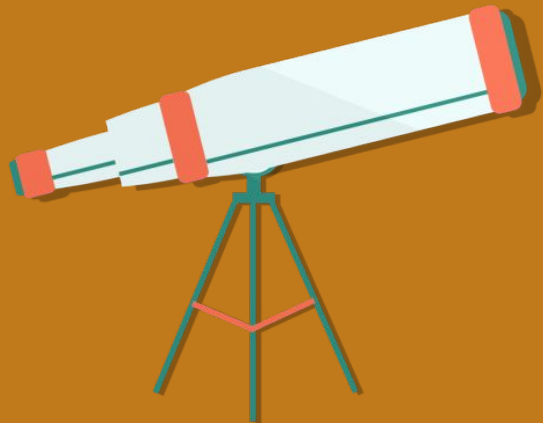


## Encore plus de filtres

- `:hidden` : sélectionne les éléments cachés (les éléments dont la valeur CSS `display` est `none`, qui sont de `type="hidden"`, ont une hauteur et une largeur nulles, ou ont un ancêtre lui-même invisible)
- `:visible` : sélectionne les éléments visibles
- `:contains("texte")` : éléments qui contiennent le texte spécifié
- `:has("élément")` : éléments qui contiennent l'élément spécifié
- `[attribut]` : éléments qui ont l'attribut spécifié, par exemple `$("[checked]")`
- `[attribut="valeur"]` : éléments qui ont l'attribut et la valeur spécifiés, par exemple `$("[align=center]")`
- `[attribut!="valeur"]` : Les éléments qui n'ont pas l'attribut et la valeur spécifiés, par exemple `$("[align!=center]")`

# Démo

- Créer un formulaire avec deux input de type text.
- Créer un bouton reverse qui permet d'inverser les contenus des deux input text.



# jQuery et la gestion d'évènement



# Gestion d'évènement

jQuery offre la possibilité de se **raccrocher** à un événement et de lier ce dernier à une traitement que l'on aurait défini.

- blur
- focus
- load
- resize
- scroll
- unload
- beforeunload
- click
- dblclick
- mousedown
- mouseup

- mousemove
- mouseover
- mouseout
- mouseenter
- mouseleave
- change
- select
- submit
- keydown
- keypress
- keyup.



# Gestion d'évènement

```
// Méthode 1
$('p').on( 'click', function () {
    alert("Click sur une balise p");
});

// Méthode 2
$('p').click(function () {
    alert("Click sur une balise p");
});
```

# Gestion d'évènement

## .click()

Categories: [Events](#) > [Mouse Events](#)

**.click( handler )**

Returns: [jQuery](#)

**Description:** Bind an event handler to the "click" JavaScript event, or trigger that event on an element.

 **.click( handler )**

version added: 1.0

**handler**

Type: [Function](#)( [Event](#) eventObject )

A function to execute each time the event is triggered.

 **.click( [ eventData ], handler )**

version added: 1.4.3

**eventData**

Type: [Anything](#)

An object containing data that will be passed to the event handler.

**handler**

Type: [Function](#)( [Event](#) eventObject )

A function to execute each time the event is triggered.

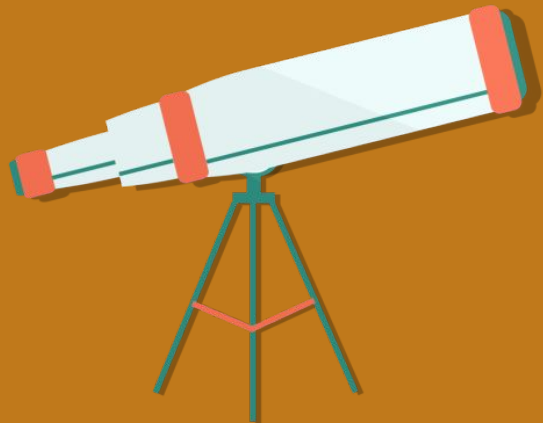
 **.click()**

version added: 1.0

This signature does not accept any arguments.

# Démo

- Au click sur un li afficher son contenu dans une popup
- Au survol d'une div avec le contenu Dark mode changer le background en gris.





**Connaissez vous le DOM ?  
(Document Object Model)**



# DOM (Document Object Model)

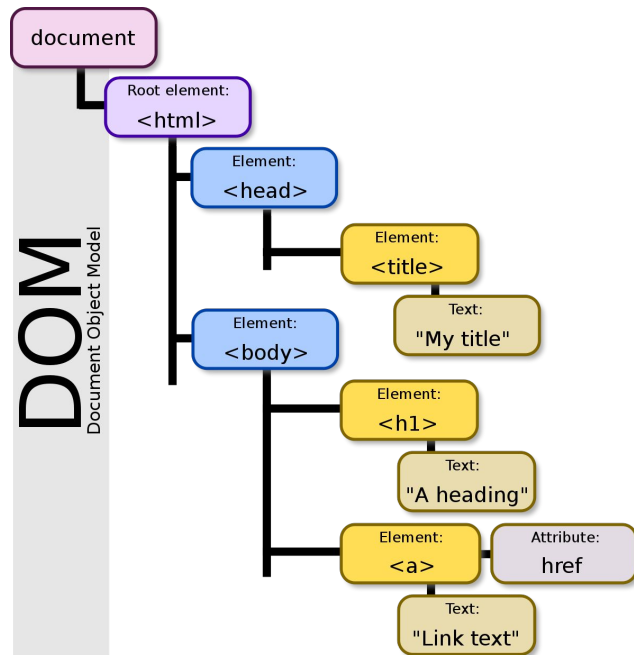
Le DOM est la modélisation que fait le navigateur du code HTML.

Le DOM transforme le HTML brute en une liste d'objet appelé noeud qui sont lié ensemble.

Le DOM donne à JavaScript et donc jQuery de pouvoir interagir sur le contenu HTML.

Ces interaction se manifeste par la possibilité de :

- Modifier du contenu
- Ajouter ou Supprimer des éléments HTML à la volée
- Parcourir, compter, filtrer des éléments HTML.
- etc



**Ajax : Quel sont les  
avantages à l'utiliser ?**



# Avantages de l'Ajax



- Asynchrone
- Economie de temps - Vitesse
- Economie de calcul
- Dynamisme de la page

# Ajax : Dans quel cas l'utiliser ?



# Exemple de cas d'utilisation

- Mise à jour d'un champ unique (Exemple : Activation/ Désactivation)
- Filtre
- Barre de recherche
- Pagination
- Scroll infini
- Envoi de formulaire après traitement spécifique



# Méthode HTTP

Le protocole HTTP définit un ensemble de **méthodes de requête** qui indiquent l'action que l'on souhaite réaliser sur la ressource indiquée.

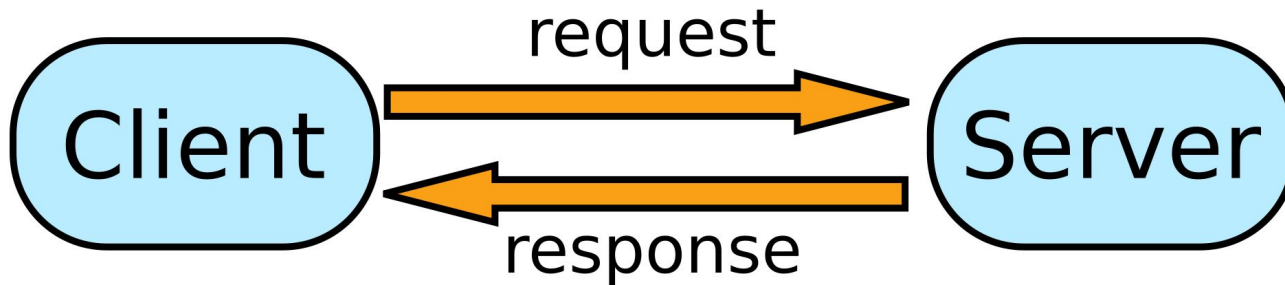
Ces méthodes sont souvent appelées *verbes HTTP*.



# Méthode HTTP

Méthode :

- *GET*
- *POST*
- *PUT*
- *DELETE*
- *etc*







# Principales méthodes HTTP

## GET

La méthode GET demande une représentation de la ressource spécifiée. Les requêtes GET doivent uniquement être utilisées afin de récupérer des données.

## POST

La méthode POST est utilisée pour envoyer une entité vers la ressource indiquée. Cela entraîne généralement un changement d'état ou des effets de bord sur le serveur.

## PUT

La méthode PUT remplace toutes les représentations actuelles de la ressource visée par le contenu de la requête.

## DELETE

La méthode DELETE supprime la ressource indiquée.

# Ajax : Comment utiliser l'Ajax ?



# Documentation JQuery Ajax


## jQuery.ajax()

Categories: [Ajax](#) > [Low-Level Interface](#)

**jQuery.ajax( url [, settings ] )**

**Returns:** [jqXHR](#)

**Description:** *Perform an asynchronous HTTP (Ajax) request.*

 **jQuery.ajax( url [, settings ] )**

version added: 1.5

**url**

Type: [String](#)

A string containing the URL to which the request is sent.

**settings**

Type: [PlainObject](#)

A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with [\\$.ajaxSetup\(\)](#). See [jQuery.ajax\( settings \)](#) below for a complete list of all settings.



## Exemple JQuery Ajax

```
1 | $.ajax({  
2 |   url: "test.html",  
3 |   context: document.body  
4 | }).done(function() {  
5 |   $( this ).addClass( "done" );  
6 | });
```



## Exemple JQuery Ajax

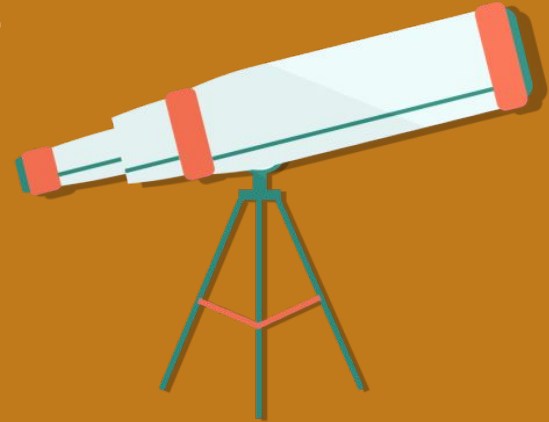
```
1 // Assign handlers immediately after making the request,  
2 // and remember the jqXHR object for this request  
3 var jqxhr = $.ajax( "example.php" )  
4   .done(function() {  
5     alert( "success" );  
6   })  
7   .fail(function() {  
8     alert( "error" );  
9   })  
10  .always(function() {  
11    alert( "complete" );  
12  });  
13  
14 // Perform other work here ...  
15  
16 // Set another completion function for the request above  
17 jqxhr.always(function() {  
18   alert( "second complete" );  
19 });
```

# Démo

- Exemple de récupération et affichage d'un chiffre aléatoire entre une valeur minimum et une valeur maximum.

random.php (Serveur)

random.html (Client)



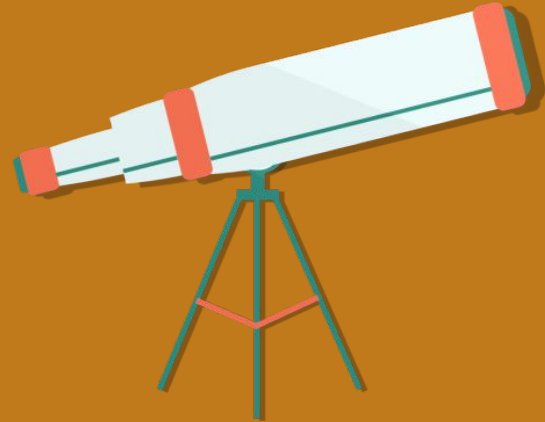
# Démo

- En vous basant de la démo précédente, créer un projet qui renvoie le bénéfice d'une entreprise en envoyant en paramètre une valeur CA dans l'url. Le bénéfice représente 60% du CA.

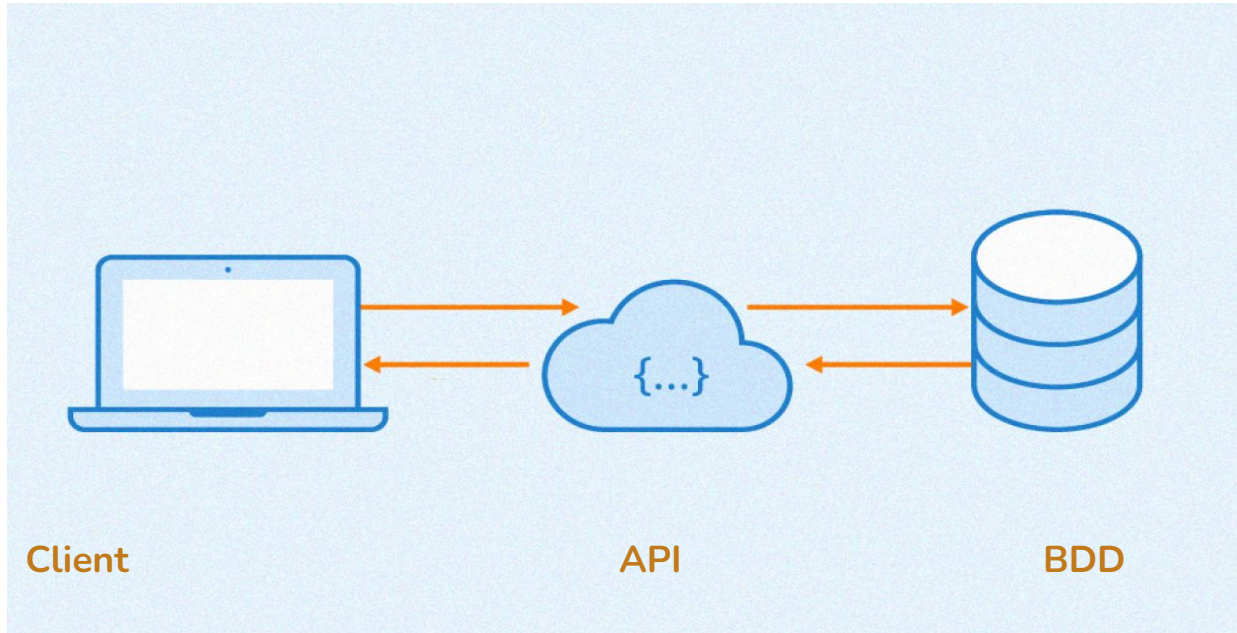
**benefice.php (Serveur)**

**benefice.html (Client)**

***Bénéfice = (CA \* 60) / 100***



# API







# Simulation serveur : API JsonPlaceholder

## {JSON} Placeholder

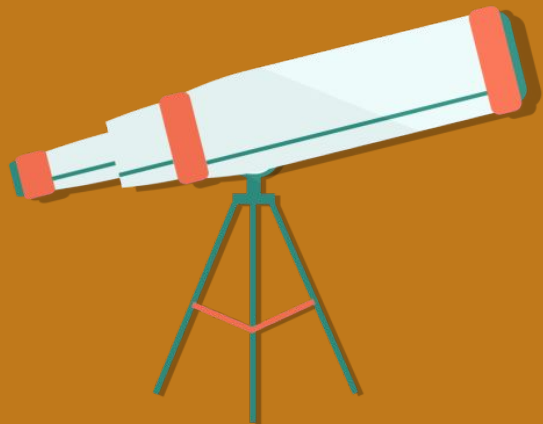
Free fake API for testing and prototyping.

Powered by [JSON Server](#) + [LowDB](#)

As of Dec 2020, serving **~1.8 billion requests each month**.

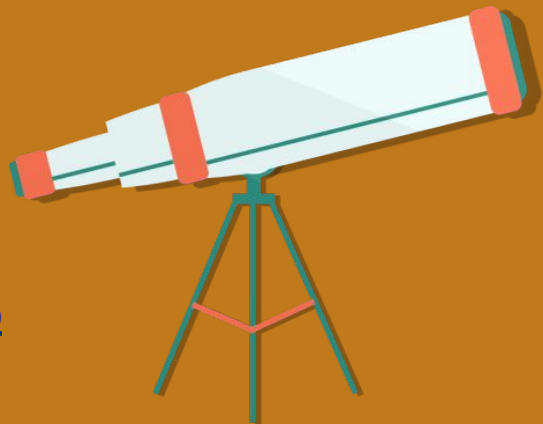
# Démo

- Analysons cette réponse serveur  
<https://jsonplaceholder.typicode.com/photos>
- Analysons cette réponse serveur  
<https://jsonplaceholder.typicode.com/photos?albumId=1>



# Démo

- Créer une page HTML avec une balise h1 avec le contenu Liste photo et une balise ul avec un li ayant la valeur Default.
- Ajouter la librairie jQuery.
- A l'aide du module Ajax récupérer la liste des photos sur le serveur **jsonplaceholder** et afficher le résultats dans la console  
(<https://jsonplaceholder.typicode.com/photos?albumId=1>)



# Démo

- Récupérer la liste de photo puis boucler sur les résultats et ajouter un li avec le titre de chaque photo.
- Afficher maintenant en plus du titre, la photo de miniature (**thumbnailUrl**)

*(Aide: jQuery Function append)*

