# TP3 JQuery / AJAX / JSON /PHP-MYSQL

JSON (JavaScript Oject Notation) est un format de données dérivé de la notation des objets et tableaux de ECMAScript (donc de JavaScript). L'avantage de JSON est qu'il est reconnu nativement par JavaScript.
Les éléments en JSON :

- Les objets : {chaîne : valeur, chaîne : valeur...}
- Les tableaux : [valeur, valeur, ...]
- Les valeurs : chaîne, nombre, objet, tableau, true, false, null
- Les chaînes : "abcdef" ou "abcd\n\t"
- Les nombres : -1234.12

**Fichier JSON 1:**

```
{"unObjet": {
  "unTableau": [12, 13, 53],
  "unNombre" : 53,
  "unChaîne" : "truc\n"
  "unObjet" : { "style" : "gras" }
}}
```

**Fichier JSON 2:**

```
{"machin": {
  "taille": 12,
  "style": "gras",
  "bidule": {
    "machin": [
      {"style" : "italique" },
      {"style" : "gras" }
    ]
  }
}}
```

**NB.** Chaque objet JSON est définit par une **clé** (ou une chaîne) et une **valeur**. Un fichier JSON peut contenir des objets JSON imbriqués (voir l'objet **bidule** du Fichier JSON 2).

**Exemple 1 : JS /JSON:**

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

<h2>Créer ub objet à partir d'une chaine JSON</h2>

<p id="demo"></p>

<script>
var text = '{"etudiants":[' +
'{"nom":"X","prenom":"Y" },' +
'{"nom":"XX","prenom":"YY" },' +
'{"nom":"XXX","prenom":"YYY" }]}';

obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.etudiants[1].nom + " " + obj.etudiants[1].prenom;
</script>

</body>
</html>
```

**Exemple 2 : AJAX /JSON :**

- Contenu du fichier JSON **« Listeetudiants.json »**

```
[
{
"nom": "X",
"prenom": "Y"
},
{
"nom": "XX",
"prenom": "YY"
},
{
"nom": "XXX",
"prenom": "YYY"
}
]
```

- Contenu de la page « **parseJson.html** »

```
<!DOCTYPE html>
<html>
<body>
<h2>Affichier un fichier JSON depuis une URL</h2>
<div id="demo"></div>
<script>
var xmlhttp = new XMLHttpRequest();
var url = "http://localhost/myapp/listeetudiants.json";

xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myArr = JSON.parse(this.responseText);
        myFunction(myArr);
    }
};
xmlhttp.open("GET", url, true);
xmlhttp.send();

function myFunction(arr) {
    var out = "";
    var i;
    for(i = 0; i < arr.length; i++) {
        out += arr[i].nom + ' ' +
        arr[i].prenom+'<br>';
    }
    document.getElementById("demo").innerHTML = out;
}
</script>
```

## Exemple 3 : JS /JSON / AJAX /SQL:

- Contenu de la page HTML **« afficherEtudiants.html »**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
</head>
<body>
<h1>Liste des étudiants</h1>
<div id="demo"></div>
<script>
var xmlhttp = new XMLHttpRequest();
var url = "http://localhost/Ajax/listeetudiants.php";

xmlhttp.onreadystatechange=function() {
    if (this.readyState == 4 && this.status == 200) {
        myFunction(this.responseText);
        alert(this.responseText);
    }
}
xmlhttp.open("GET", url, true);
xmlhttp.send();

function myFunction(response) {
    var obj = JSON.parse(response);
   var arr=obj.etudiants;
    var i;
    var out = "<table>";

    for(i = 0; i < arr.length; i++) {
        out += "<tr><td>" +
        arr[i].cin +
        "</td><td>" +
        arr[i].nom +
        "</td><td>" +
        arr[i].prenom +
        "</td><td>" +
        arr[i].email +
        "</td><td>" +
        arr[i].adresse +
        "</td><td>" +
        arr[i].classe +
        "</td></tr>";
    }
    out += "</table>";
    document.getElementById("demo").innerHTML = out;
}
</script>
</body>
</html>
```

- Contenu de la page **listeetudiants.php**

```
<?php
```

```
try
{
$bdd = new
PDO('mysql:host=localhost;dbname=gestion_etudiant;charset=utf8'
, 'root', 'lfig2');
}
catch(Exception $e)
{
        die('Erreur : '.$e->getMessage());
}
$req="SELECT * FROM etudiant";
$reponse = $bdd->query($req);
if($reponse->rowCount()>0) {
      $outputs["etudiants"]=array();
while ($row = $reponse ->fetch(PDO::FETCH_ASSOC)) {
        $etudiant = array();
        $etudiant["id"] = $row["id"];
        $etudiant["nom"] = $row["nom"];
        $etudiant["prenom"] = $row["prenom"];
        $etudiant["adresse"] = $row["adresse"];
        $etudiant["email"] = $row["email"];
        $etudiant["classe"] = $row["Classe"];
         array_push($outputs["etudiants"], $etudiant);
    }
    // success
    $outputs["success"] = 1;
     echo json_encode($outputs);
} else {
    $outputs["success"] = 0;
    $outputs["message"] = "Pas d'étudiants";
    // echo no users JSON
    echo json_encode($outputs);
```

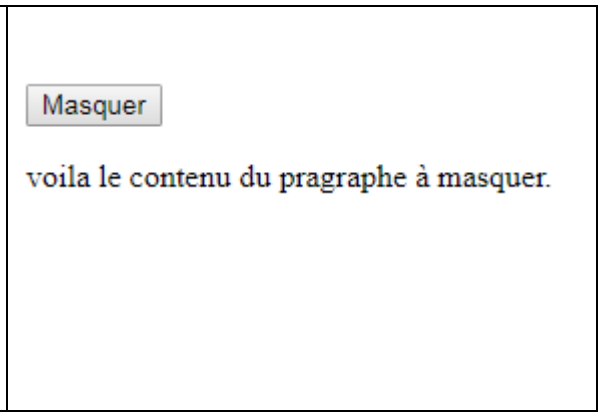
```
}

?>
```

## Exercice 4 : Fonctions Callback JQuery

Soit le code HTML suivant.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs
/jquery/3.2.1/jquery.min.js"></script>
</head>
<body>
<button>Masquer</button>
<p>voila le contenu du pragraphe à
masquer.</p>
</body>
</html>
```




On vous demande de créer un script JS qui permet de

- Masquer l'élément paragraphe en cliquant sur le bouton masquer. Utiliser l'option « slow » comme effet d'animation.
- Afficher un message d'alerte à la fin de l'animation (paragraphe masquée).

| Solution sans Fonction callback | Solution avec Fonction callback |
|---|---|
| `<script>`<br>`$(document).ready(function(){`<br>`  $("button").click(function(){`<br>`  $("p").hide(1000);`<br>`  alert("Le paragraphe est masqué maintenant");`<br>`  });`<br>`});`<br>`</script>` | `<script>`<br>`$(document).ready(function(){`<br>`  $("button").click(function(){`<br>`  $("p").hide("slow", function(){`<br>`  alert("Le paragraphe est masqué maintenant");`<br>`  });`<br>`  });`<br>`});`<br>`</script>` |

## Exercice 5 : JQuery, Requêtes Ajax (Get/Post)

# Liste des étudiants


On veut utiliser JQuery pour afficher la liste des étudiants. Il s'agit d'une version JQuery de l'exercice 3 (on veut simplifier les requêtes Ajax).

- Contenu de la page **afficherEtudiantsJQJSON.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>
<script>
$(document).ready(function(){
    $("#btn").click(function(){

$.get("http://localhost/Ajax/listeetudiants.php",function(data,
status){
                myFunction(data);
        },"json");
    });
});
function myFunction(response) {
   var arr=response.etudiants;
    var i;
    var out = "<table>";
    for(i = 0; i < arr.length; i++) {
        out += "<tr><td>" +
        arr[i].cin +
        "</td><td>" +
        arr[i].nom +
        "</td><td>" +
        arr[i].prenom +
        "</td><td>" +
        arr[i].email +
        "</td><td>" +
        arr[i].adresse +
        "</td><td>" +
        arr[i].classe +
        "</td></tr>";
    }
    out += "</table>";
    document.getElementById("demo").innerHTML = out;
}
</script>
```

```
</head>
<body>

<h1>Liste des étudiants</h1>
<div id="demo"></div>
<br>
<input id="btn" type="button" value="Cliquer ici!">
</body>
</html>
```

## Exercice 6 : JQuery, Methode getJSON()

Afin de recuperer des données JSON depuis un serveur, il vaut mieux utiliser la méthode getJSON() qui permet de simplifier le traitement des documents JSON (version simplifiée de la méthode $.ajax()). On vous demande de modifier l'**exercice 5** afin d'utiliser la fonction getJSON().

- Contenu de la page **afficherEtudiantsgetJSON.html**

```
<script>
    $(document).ready(function(){
        $("#btn").click(function(){
        $.getJSON("http://localhost/Ajax/listeetudiants.php")
                .done(function(data, status){
                    console.log(data);
                    myFunction(data);
                });
        });
    });
</script>
```

## Exercice 7: JSONP avec la méthode getJSON()

JSON fonctionne bien avec XmlHttpRequest. Mais, les navigateurs interdisent de faire du cross-domain (cross-site) avec XMLHttpRequest pour des raisons de sécurité. D'où, il n'est pas possible d'utiliser JSON pour des demandes à d'autres sites. JSONP (JSON Padding) est une technique utilisée pour surmonter les **restrictions inter-domaines** (*cross-domain*) imposées par les navigateurs pour permettre aux données d'être récupérées à partir des serveurs autres que celui d'où la page a été servie. Son principe est le suivant :

- Au lieu d'aller charger un fichier JSON, on charge un script : une fonction JavaScript de callback (nommée Padding)

- La fonction est renvoyée par le serveur au client, qui l'exécute au retour, autorisant ce même client à charger des données JSON depuis un domaine externe, et donc à notifier l'objet appelant avec cette fonction de callback.

On vous demande alors de modifier l'**exercice 6** afin d'utiliser la technique JSONP.

- Contenu de la page **afficherEtudiantsgetJSONP.html**

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jque
ry.min.js"></script>
<script>
    $(document).ready(function(){
        $("#btn").click(function(){
            $.getJSON("http://www-lipn.univ-
paris13.fr/~saad/etudiants.js?jsonpcallback=?");
        });
    });
    function jsonpcallback(response) {
        var arr=response.etudiants;
        var i;
        var out = "<table>";
        for(i = 0; i < arr.length; i++) {
            out += "<tr><td>" +
                arr[i].cin +
                "</td><td>" +
                arr[i].nom +
                "</td><td>" +
                arr[i].prenom +
                "</td><td>" +
                arr[i].email +
                "</td><td>" +
                arr[i].adresse +
                "</td><td>" +
                arr[i].classe +
                "</td></tr>";
        }
        out += "</table>";
        document.getElementById("demo").innerHTML = out;
    }
</script>
```

- Script JS **etudiants.js**

```
jsonpcallback({"etudiants":[{"cin":"1111","nom":"N1","prenom":"P1","adresse
":"Tunis","email":"N1.P1@gmail.com","classe":"2-GNINF"},
{"cin":"2222","nom":"N2","prenom":"P2","adresse":"Tunis","email":"N2.P2@
gmail.com","classe":"2-GNINF"},
{"cin":"3333","nom":"N3","prenom":"P3","adresse":"Tunis","email":"N3.P3@
gmail.com","classe":"3-GNINF"},
```

{"cin":"4444","nom":"N4","prenom":"P4","adresse":"Tunis","email":"N4.P4@gmail.com","classe":"2-GNINF"}]

});