IFT-6800, Automne 2016

Cours 10: Communiquer avec un serveur en javascript

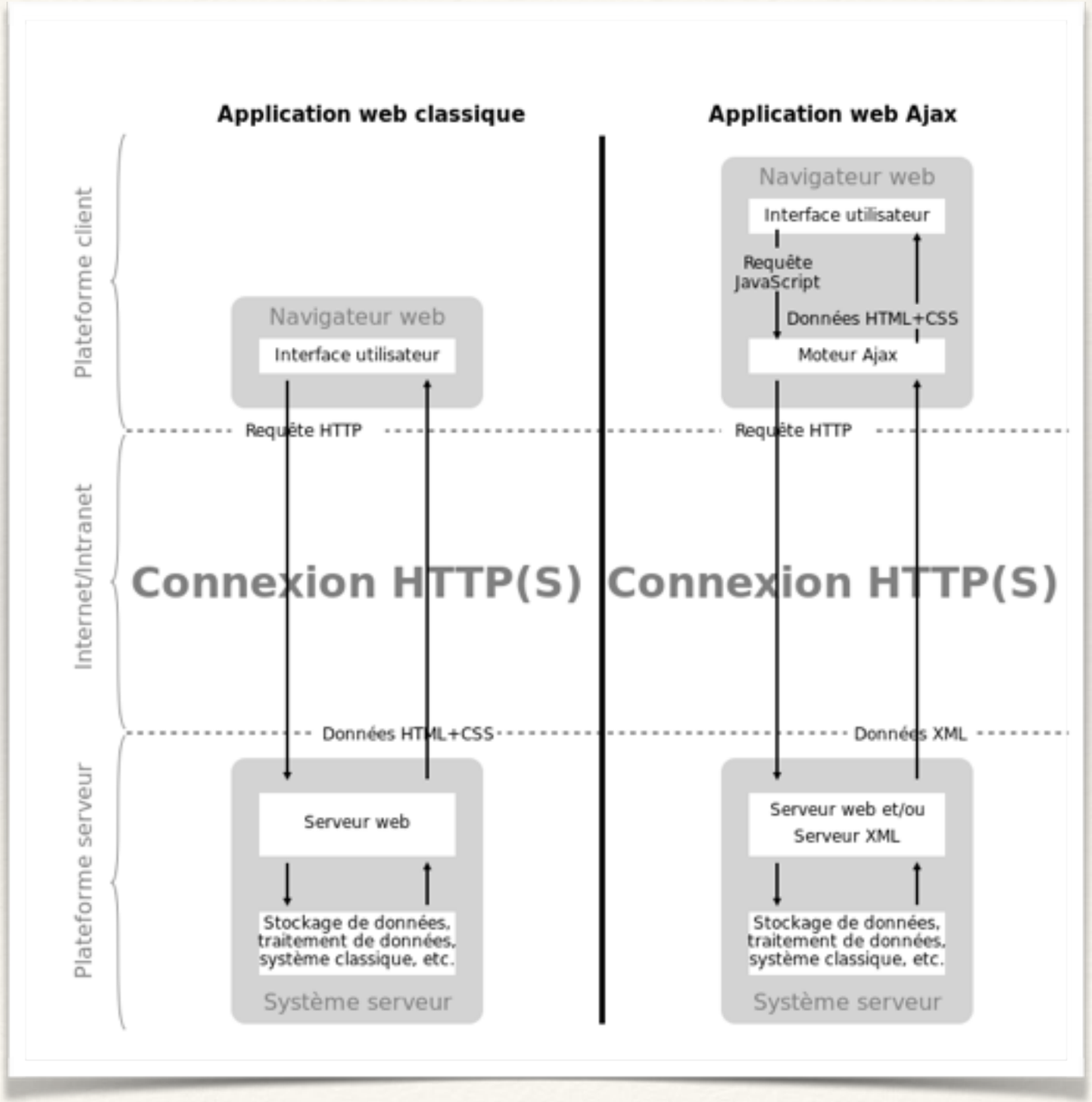
Louis Salvail
André-Aisenstadt, #3369
salvail@iro.umontreal.ca

Les limitations de javascript de base

- La mise à jour de la page web par javascript est faite par la réalisation du standard W3C Document Object Model (DOM, 1998) qui permet à des programmes et à des scripts d'accéder et de mettre à jour le contenu, la structure et le style d'un document HTML.
  - C'est le standard qui est utilisé lorsque les événements sont interceptés dans les balises pour lancer une fonction javascript.
- Javascript a été développé pour rouler du côté client. Il n'avait pas de mécanisme pour transmettre des requêtes à des serveurs.
- Les choses ont changé avec Ajax (2005):
  - Ajax: *Asynchronous Javascript and XML*.
  - des composantes de Microsoft ActiveX ajoutées aux navigateurs entre 2002-2005.

Ajax

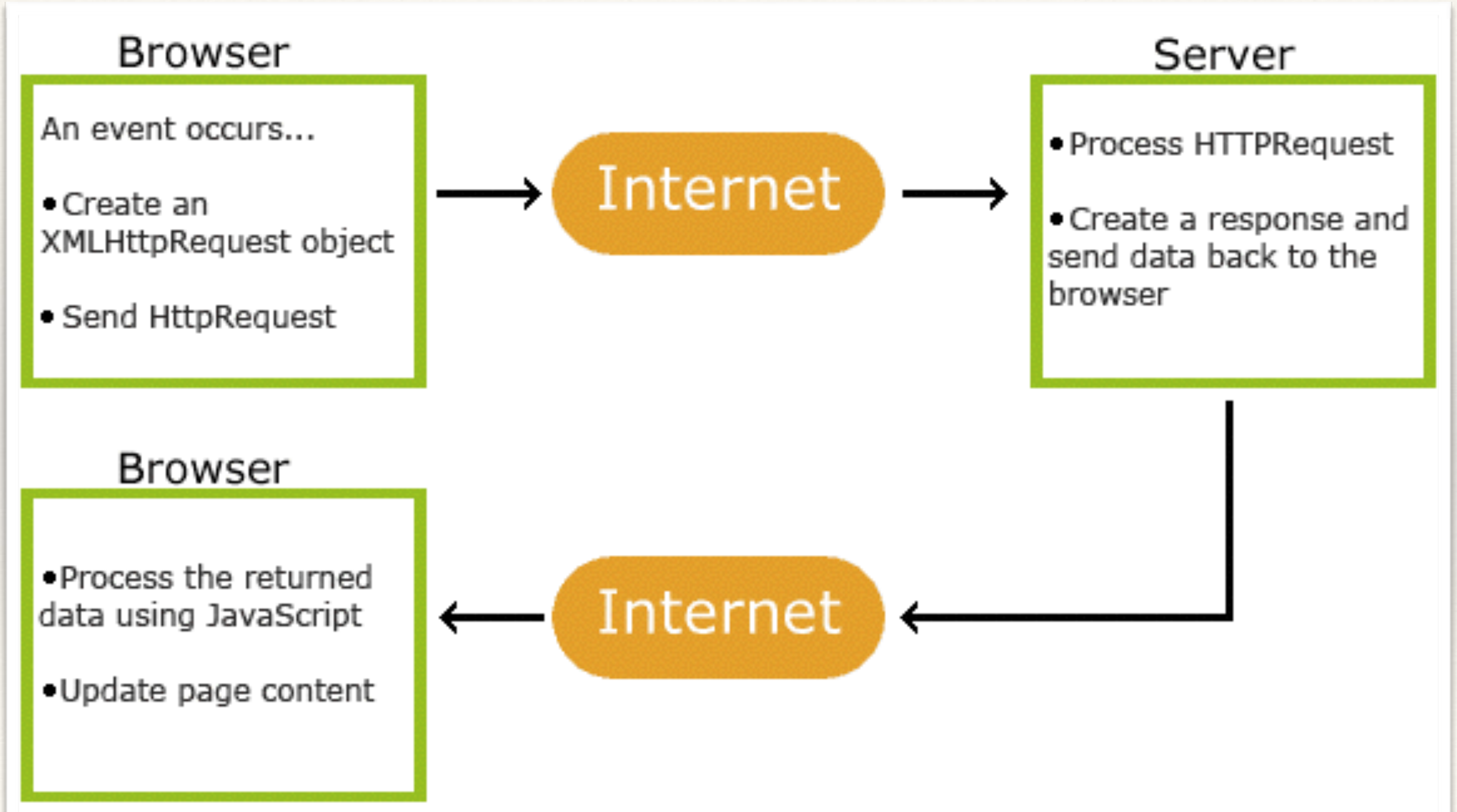
- DOM et Javascript permettent de modifier l'information présentée par le navigateur,
- L'objet Ajax, `XMLHttpRequest` permet les dialogues asynchrones avec le serveur web
- XML et d'autres formats (comme JSON) permettent de structurer l'information transmise entre le serveur et le navigateur.



Ajax (II)

- Ajax est une technique qui permet de créer des pages web dynamiques efficaces.
- Permet aux pages web d'être mises à jour d'une façon asynchrone par l'échange de petites quantités d'information avec le serveur.
- Ceci permet de mettre à jour des parties d'une page sans recharger la page au complet.
- Les pages web d'avant Ajax devaient recharger la page au complet si son contenu était modifié.
- Si une requête était fait à site web, le script client devait attendre la réponse du serveur avant de pouvoir procéder. Ces requêtes étaient synchrones.

Ajax (III)


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

XMLHttpRequest

- La création d'une requête vers un serveur est faite en déclarant et en créant une requête XML de la façon suivante:
  - `var  req = new XMLHttpRequest();`
- Cet objet nous permettra de faire des requêtes asynchrones vers un serveur web.
- Il contient plusieurs propriétés et méthodes pour faire le travail et plus encore.

Transmettre une requête

- Une fois un objet `XMLHttpRequest` `req` créé, il est assez facile de transmettre une requête.

- Supposez qu'un fichier `roman.txt` est disponible sur le serveur:

```
req.open("GET", "roman.txt", true);

req.send();
```

asynchrone =true
synchrone=false

Les deux méthodes possibles à utiliser sont
GET: simple et rapide.
POST: utile lorsque les données sont nombreuses. Lorsque des fichiers en cache ne doivent pas être considérés. Plus robuste, plus sûre.

puisque l'URL n'est pas spécifié, le fichier est accessible à l'URL du serveur courant au même niveau.

Recevoir une requête

- Une fois la requête transmise, javascript doit faire quelque chose avec la réponse.
- La façon de faire d'Ajax pour une requête asynchrone est de fournir la fonction à exécuter lorsque la réponse à la requête est disponible.
- Avant que la réponse ne soit disponible le script local peut continuer à rouler et faire autre chose.
- On peut voir ça comme si javascript devenait multi-tâche ce qu'il n'est pas sans Ajax.

Recevoir un requête (II)

- Soit `req` un objet `XMLHttpRequest`.
- Indiquer quoi faire lorsque la requête sera prête se fait de cette façon:

```
req.onreadystatechange=function(){
    if (req.readyState==4 && req.status==200){
        faire ce qu'il faut…;
    }
};
```

`req.responseText` contient la réponse sous forme texte.

terminé et prêt

OK

ajax1.html

Exemple

```
<!DOCTYPE html>
<html>
<body>
<h2>Testons AJAX</h2>
<button type="button" onclick="chargeDoc()">Demande données</button>
<p id="demo"></p>
<script>
function chargeDoc() {
  var req = new XMLHttpRequest();
    req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      document.getElementById("demo").innerHTML = req.responseText;
    }
  };
  req.open("GET", "http://www.w3schools.com/ajax/demo_get.asp", true);
  req.send();
}
</script>

</body>
</html>
```

Fonction pour une fonction

- Il est possible en javascript de donner des fonctions en paramètre d'une autre fonction. Ceci est appelé *callback function* en anglais.
- Ceci peut nous être utile pour la réception de la réponse d'une requête Ajax.
- Au lieu de faire tout le traitement dans la fonction qui est appelée lorsque la réponse de la requête est prête, le traitement, une fois la réponse reçue, peut être fait dans une fonction séparée envoyée en paramètre à la fonction appelée lorsque la requête est reçue.

ajax2.html

Exemple

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

<p id="demo">Ce texte va changer par Ajax.</p>

<button type="button"
   onclick="loadDoc('http://www.w3schools.com/ajax/ajax_info.txt', maFonction)">Changer le contenu</button>

<script>
function loadDoc(url, cfunc) {
  var req;
  req=new XMLHttpRequest();
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      cfunc(req);
    }
  };
  req.open("GET", url, true);
  req.send();
}
function maFonction(req) {
  document.getElementById("demo").innerHTML = req.responseText;
}
</script>

</body>
</html>
```

Comment Google vous propose des choses

- Lorsque vous cherchez quelque chose sur Google, vous obtenez des propositions pour compléter votre requête à mesure que des caractères sont entrés sur l'incitateur.
- Évidemment, ces propositions ne peuvent pas venir du côté client. La page web que vous regardez ne contient pas tous les sites web possibles.
- Il faut donc utiliser le serveur web de Google pour y parvenir.
- Ceci peut être réalisé par une requête Ajax à un script qui roule sur le serveur. Habituellement un script `php`.

ajax3.html

Exemple

```
<!DOCTYPE html>
<html>
<body>
<h3>Entrez du texte:</h3>
<form action="">
Prénom: <input type="text" id="txt1" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span style="color:blue" id="txtHint"></span></p>
<script>
function showHint(str) {
  var req;
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  }
  req = new XMLHttpRequest();
  req.onreadystatechange = function() {
    if (req.readyState == 4 && req.status == 200) {
      document.getElementById("txtHint").innerHTML = req.responseText;
    }
  };
  req.open("GET", "http://www.w3schools.com/ajax/gethint.asp?q="+str, true);
  req.send();
}
</script>
</body>
</html>
```

Lorsque l'utilisateur lâche une touche

La valeur de ce champs texte

… désigne une partie de texte que l'on peut ensuite meubler en javascript

Les avantages et limites d'Ajax

- Dans le passé récent, les pages web devaient être rechargées entièrement pour être mises à jour.
- Par exemple, un système courriel web (i.e. webmail) demandait à l'usager de recharger sa boîte d'entrée pour voir ses nouveaux courriels. Le serveur devait donc reconstruire et transmettre la page (HTML+CSS+javascript+courriels). Très inefficace lorsque seulement les nouveaux messages sont manquants dans la page courante du client.
- L'objet `XMLHttpRequest` d'Ajax permet d'aller chercher seulement les nouveaux messages sans le reste de la page. Javascript est ensuite responsable pour la mise à jour de la page en utilisant l'information obtenue du serveur.

Les avantages et limites d'Ajax (II)

- Ajax devenait très utile pour la mise en forme de site réactif comme GoogleMaps, Gmail, etc… sans avoir à recharger la page, rendant les échanges beaucoup plus efficaces.
- Les requêtes Ajax sont lancées par javascript qui transmet une requête à un URL pour qu'une fonction callback (puisque la requête est asynchrone) soit lancée lorsque la réponse est prête du côté client.
- Malheureusement, différents navigateurs réalisent l'API Ajax de façons différentes. Les développeurs devaient donc tenir compte des différences de chaque navigateur pour que les pages fonctionnent de façon universelle.
- Une autre limite des anciennes versions d'Ajax est son incapacité à contacter un autre serveur que le serveur web de la page courante. Les requêtes doivent être faites à l'endroit où la page web origine.

JQuery

- Ces problèmes peuvent être évités en utilisant une bibliothèque javascript qui donne accès à Ajax sans avoir à se soucier des différences entre navigateurs.
- JQuery offre toutes les possibilités d'Ajax ainsi que des méthodes simples qui facilitent son utilisation.
- La plupart des applications JQuery n'utilisent pas XML (même si Ajax est derrière). Les données transmises sont habituellement du HTML ou JSON. JSON (JavaScript Object Notation) permet de transmettre des objets javascript.
- JQuery permet la connexion à des serveurs étrangers par l'utilisation de JSON à l'intérieur de la balise `<script>` permettant de charger des données JSON.

Types de données JQuery

- Les deux façons de transmettre des requêtes à un serveur sont GET et POST:
  - GET: Utilisée pour des opérations non destructives, qui ne modifient pas l'état du serveur. Seulement pour obtenir de l'information. Les requêtes GET peuvent être mises en cache par le navigateur causant des comportements imprévisibles dans certains cas. Les données de ces requêtes sont habituellement dans la chaîne de requête (i.e. ajoutées à l'URL).
  - POST: Utilisée pour les opérations destructives. Elles ne sont habituellement pas mises en cache. Les données sont habituellement transmises séparément de la requête.
- Les types de données supportés pour le transmission du résultat des requêtes:
  - **text:** pour transporter des chaînes de caractères simples,
  - **html:** pour transporter des blocs de code HTML,
  - **script:** pour ajouter des nouveaux scripts à la page,
  - **json:** pour le transport d'objet, de tableaux, etc…
  - **xml:** pour le transport de données selon un schéma XML.

L'objet JQuery

- JQuery se présente comme un fichier javascript de 247Ko nommé `jquery.js`.
- Il est inclus dans la page web en indiquant le fichier dans la balise `<script>`:
  - `<script src="/chemin/vers/jquery.js"></script>`
- La bibliothèque JQuery peut être appelée de deux façon: `JQuery` ou `$`.

jquery1.html

Exemple

inclure JQuery

$("#sortie") est l'objet qui correspond à la balise avec id="sortie"

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Tester JQuery</title>
  <meta charset=UTF-8">
  <script src="jquery.js"></script>
</head>
<body>
  <h1>L'objet JQuery $</h1>
  <p>L'objet $ donne accès à la bibliothèque JQuery.
     Une méthode de $ est 'each' qui permet d'évaluer une fonction tour à tour sur
     une suite d'éléments. Voici le résultat pour cette instruction:</p>
  <span style="font-family:Courier New; color:blue">
     $.each([1,2,3], function(){document.write(this+1);});
  </span>
  <br><br>
  <script>
     $.each([1,2,3], function(){document.write(this+1);});
  </script>
  <p>Maintenant, voici le fameux exemple 'Allo le monde'
     qui sera affichée entre les balises 'span' avec id=''sortie''.</p>
  <br>
    <span id="sortie"></span>
  <script>
     $(function(){$("#sortie").html("<b>Bonjour le monde!</b>");});
  </script>
</body>
</html>
```

signifie id=

Demande la modification du contenu entre cette balise et sa fermante.

jquery2.html

Exemple (II)

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Tester JQuery</title>
  <meta charset=UTF-8">
  <script src="jquery.js"></script>
</head>
<body>
  <h1>L'objet JQuery $</h1>
  <p>L'objet $ donne accès à la bibliothèque JQuery. Une méthode de $ est
     'each' qui permet d'évaluer une fonction tour à tour sur une suite d'éléments.
     Voici le résultat pour cette instruction:</p>
     <span style="font-family:Courier New; color:blue">
        $.each([2,4,6], function(){document.write(this+1);});
     </span>
     <br><br>
  <script>
   $.each([2,4,6], function(){document.write(this+1);});
  </script>
  <p>Maintenant, voici le fameux exemple 'Bonjour le monde'
       qui sera affichée entre les balises 'span' avec id=''sortie''.</p>
       <br>
       <span class="test" id="sortie">Ce texte sera modifié</span>
     <script>
       $(function(){$("#sortie").html("<b>Bonjour le monde!</b>");});
       $(".test").attr("style", "font-family:Courier New;color:red");
       </script>
</body>
</html>
```

this = la valeur courante parmi [2,4,6]

La balise sera meublée avec le style 'test'

Définition JQuery du style 'test': Rouge avec police 'Courier New'.

JQuery: Réagir à un bouton et faire une requête Ajax

- L'objet $ permet d'intercepter des événement un peu différemment de la façon vue qui déclare l'événement dans la balise ouvrante via ses paramètres:
  - `$("#nom_de_ID").click(function(event){…});`
- Voyons maintenant comment faire une requête Ajax en utilisant l'objet $:
  - Un fichier texte: `$.get(url/fichier.txt,function(res){});`
  - Un fichier html: `$.get(url/fichier.html,function(res){});`
  - Un fichier JSON: `$.getJSON(url/fichier.json,function(res){…});`

jquery3.html

Exemple

Lorsque l'usager click sur le bouton 'déclencheur' la fonction function(event) est exécutée avec l'événement 'event' en paramètre.

```
<html>
   <head>
      <title>JQuery réagit à un bouton et se connecte</title>
      <script src = "jquery.js"></script>
      <script>
         $(document).ready(function() {
            $("#declencheur").click(function(event){
                  $.getJSON('http://www.tutorialspoint.com/jquery/result.json', function(jd) {
                  $('#sortie').html('<p>Nom: ' + jd.name + '</p>');
                  $('#sortie').append('<p>Âge: ' + jd.age+ '</p>');
                  $('#sortie').append('<p>Genre: ' + jd.sex+ '</p>');
               });

            });
         });
      </script>
   </head>
   <body>
      <p>Appuyez sur le bouton pour charger le fichier result.json.</p>
      <div id = "sortie" style = "background-color:#eee;">
         Les données seront mises ici...
      </div>
      <input type = "button" id = "declencheur" value = "Lire le fichier" />
   </body>
</html>
```

jd est un objet avec 3 champs: jd.name, jd.age, jd.sex

Le fichier result.json est transformé en objet javascript avec lequel la fonction function(jd) est appelée, jd est l'objet javascript généré.

jquery4.html

La charrue avant ou après le boeuf?

Voici maintenant d'autres méthodes JQuery qui permettent de modifier la page web en ajoutant *avant* ou *après* une balise:

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("#boeuf").before("<img src='charrue1.jpg' width='80' height='80'>");
    });
    $("#btn2").click(function(){
        $("#boeuf").after("<img src='charrue1.jpg' width='80' height='80'>");
    });
});
</script>
</head>
<body>
<h1>La charrue avant ou après le boeuf?</h1>
Placez la charrue où vous voulez:<br>
<br>
<img src="boeuf.jpg" id="boeuf"  width="100" height="140"><br><br>
<button id="btn1">Charrue après</button>
<button id="btn2">Charrue avant</button>
</body>
</html>
```

jquery5.html

Enlever les char

Toutes les charrues à droite du boeuf sont de classe charrueD, celles à gauche sont de classe charrueG.

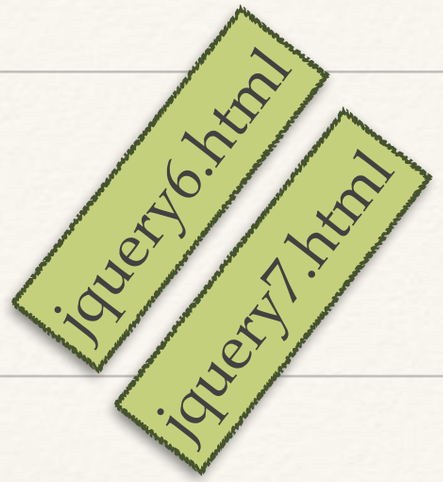
```
<!DOCTYPE html>
<html>
<head>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("#boeuf").before("<img src='charrue1.jpg' class="charrueD" width='80' height='80'>");
    });
    $("#btn2").click(function(){
        $("#boeuf").after("<img src='charrue1.jpg' class="charrueG" width='80' height='80'>");
    });
    $("#btn3").click(function(){
        $("img").remove(".charrueD");
    });
    $("#btn4").click(function(){
        $("img").remove(".charrueG");
    });
});
</script>
</head>
<body>
<h1>La charrue avant ou après le boeuf?</h1>
Placez la charrue où vous voulez:<br>
<br>
<img src="boeuf.jpg" id="boeuf"  width="100" height="140"><br><br>
<button id="btn1">Charrue avant</button>
<button id="btn2">Charrue après</button>
<button id="btn3">Enlève les charrues avants</button>
<button id="btn4">Enlève les charrues après</button>
</body>
</html>
```

Efface les charrues à droite du boeuf.

JQuery pour les méthodes Ajax

- Charger du texte:

  - `$(sélecteur).load(url, données, callback)`
  - Ex: `$("#monId).load("demo.txt");`
  - Ex: `$("#monId).load("demo.txt #p1");` charge la balise qui contient `id=monId` avec le contenu de l'élement `id="p1"` dans `demo.txt`.
  - Ex: `$("#monId).load("demo.txt", function(rTxt,sTxt, req ) {…});` charge le texte à l'endroit désigné après avoir appelé la fonction callback avec le texte reçu, l'état de la requête et la requête Ajax XMLHttpRequest.



Exemple

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("http://www.w3schools.com/jquery/demo_test.txt", function(rTxt, sTxt, req){
            if(sTxt == "success")
                alert("Le contenu a été chargé avec succès.");
            if(sTxt == "error")
                alert("Erreur: " + req.status + ": " + req.statusText);
        });
    });
  });
</script>
</head>
<body>
<div id="div1"><h2>Requête jQuery AJAX qui change ce texte</h2></div>
<button>Charger et afficher le contenu</button>
</body>
</html>
```

GET & POST

Si la requête nécessite que des données soient transmises au serveur alors celles-ci sont ajoutées à l'URL du script avec un séparateur:
Ex: www.ab.ca/script.php?input

- `$.get(url, fctn_callback(données, status){…});`
  - GET: requête pour une ressource indiquée à l'URL `url` (souvent un script `php` ou `asp`). Lorsque le résultat de la requête est retourné, la fonction `fctn_callback(données, status)` est appelée avec les données et l'état résultant en paramètre.
- `$.post(url, données, fctn_callback(données, status){…});`
  - POST: soumet des données qui seront traitées à l'URL `url` (habituellement un script `php` ou `asp`). Lorsque le résultat de la requête est retournée, la fonction `fctn_callback(données, status)` est appelée avec les données et l'état résultant en paramètre.

jquery8.html

Exemple: GET

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.get("http://www.w3schools.com/jquery/demo_test.asp", function(data, status){
            alert("Données: " + data + "\nÉtat: " + status);
        });
    });
});
</script>
</head>
<body>
<button>Transmet une requête HTTP GET <br> à
    un script asp et obtient le résultat</button>
</body>
</html>
```

jquery9.html

Exemple: POST

```
...
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.post("http://www.w3schools.com/jquery/demo_test_post.asp",
        {name: "Bonhomme Carnaval",city: "Québec"},
        function(data,status){
            alert("Données: " + data + "\nÉtat: " + status);
        });
    });
});
</script>
</head>
<body>
<p>Le script asp accepte un couple {name: "nom", city: "ville"}
    et retourne quelque chose...</p>
<button>Transmet une requête HTTP POST <br> à
    un script asp et obtient le résulat</button>
</body>
...
```

JQuery et SQL

- Nous pouvons maintenant formuler des requêtes à un serveur de base de données, en autant qu'un script du côté serveur achemine nos requêtes SQL.
- C'est l'approche que nous avons adoptée pour nous permette d'interroger la base de données sur le baseball.
- À l'URL `www-ens.iro.umontreal.ca/~dift6800/baseball`, un script `php` (i.e. `db.php`) attend une requête SQL pour la retransmettre au serveur SQL.
- Le script attend les requête de type POST.
- Le script retourne la requête en format JSON qui peut être traité par des méthodes de JQuery.

dbscript1.html

Squelette HTML pour le baseball

- Voici le squelette HTML avec javascript qui permet de transmettre une requête SQL au script `db.php`:

```
..
<head>
  <title>IFT6800 Baseball</title>
  <meta charset="utf-8">
  <script src="jquery.js"></script>
  <script>
  var postData = {};//contient ce que mon backend en php s'attend à recevoir
  postData["db"] = "dift6800_baseball";
  postData["query"] = "* from Teams where teamID='OAK'";
  $.post(
    "http://www-ens.iro.umontreal.ca/~dift6800/baseball/db.php",
    postData,
    function(reponse,status){
      console.log(status);
      console.log(reponse);
    }
  );
  </script>
</head>
<body>
  <h1>SQL en javascript.</h1>
</body>
...
```

`postData={"db":"dift6800_baseball", "query":"..."};`

La requête transmise ne contient pas le mot `select` au début.

La requête SQL sans le mot `select` au début.

`console.log(..)` écrit sur la console du navigateur.

C'est cette fonction qui doit être modifiée pour mettre à jour votre page.

Le plus de circuits depuis 1950

- La requête SQL suivante retourne les joueurs qui ont frappé le plus de circuits en saison (avec le nombre de circuits) depuis 1950 dans la ligue Nationale:

```
select CC.yearId, CC.MaxCC, Master.nameFirst,
Master.nameLast

from Batting, (select yearID, max(HR) as MaxCC from
Batting where lgID='NL' group by yearId having
yearId>=1950) as CC, Master

where Master.playerId=Batting.playerId and
Batting.HR=CC.MaxCC AND Batting.yearId=CC.yearId

order by CC.yearId DESC;
```

Sortie SQL

yearId	MaxCC	nameFirst	nameLast
2014	37	Chris	Carter
2014	37	Giancarlo	Stanton
2013	36	Edwin	Encarnacion
2013	36	Pedro	Alvarez
2013	36	Paul	Goldschmidt
2012	41	Ryan	Braun
2012	41	Adam	Dunn
2011	39	Mark	Teixeira
2011	39	Matt	Kemp
2010	42	Albert	Pujols
2009	47	Albert	Pujols
2008	48	Ryan	Howard
2007	50	Prince	Fielder
2006	58	Ryan	Howard
2005	51	Andruw	Jones
2004	48	Adrian	Beltre
2003	47	Alex	Rodriguez
2003	47	Jim	Thome
2002	49	Sammy	Sosa
2001	73	Barry	Bonds
2000	50	Sammy	Sosa
1999	65	Mark	McGwire
1998	70	Mark	McGwire

… … … …

dbscript2.html

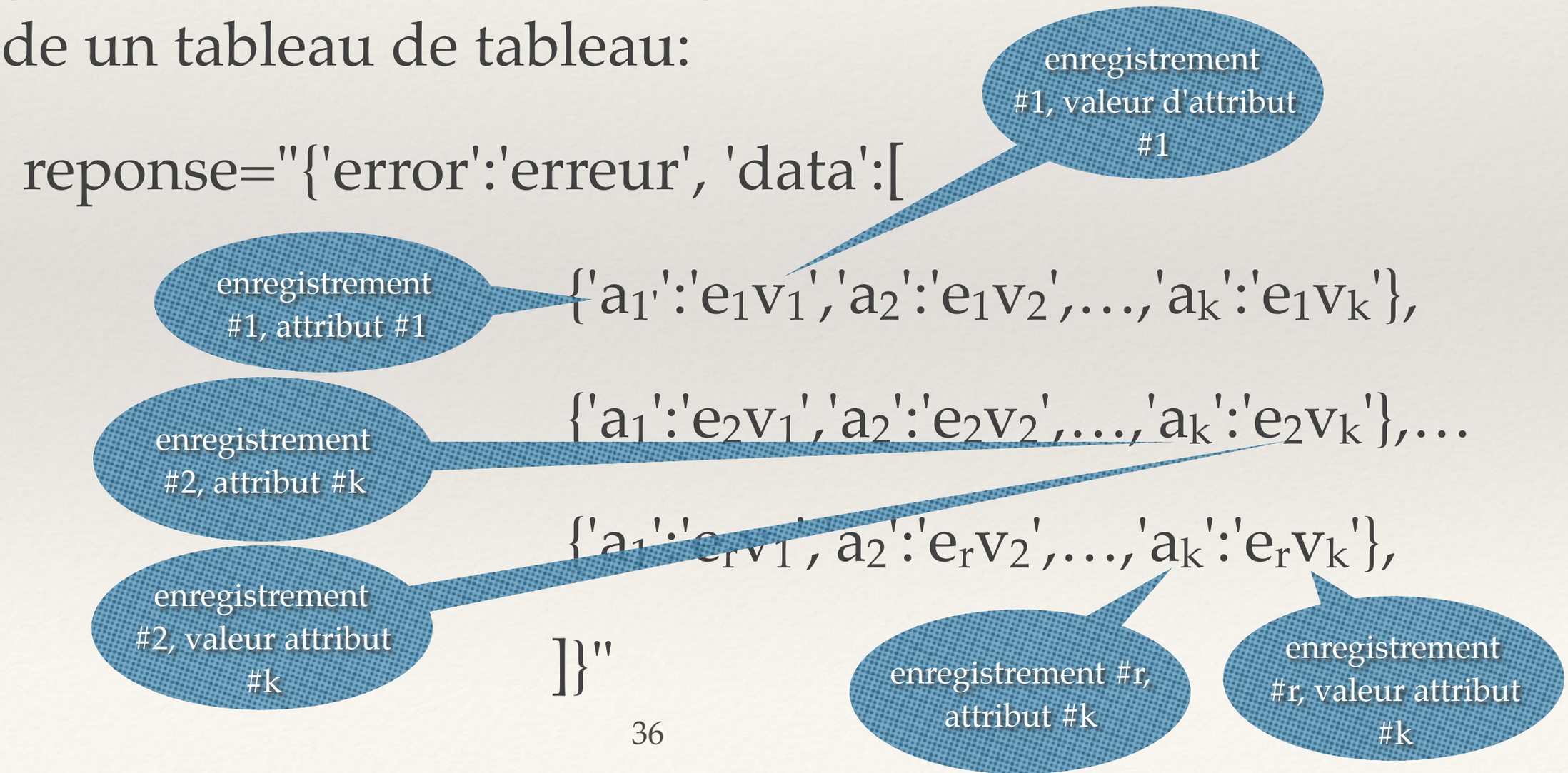
Les meilleurs frappeurs de circuits

```
..
<head>
  <title>IFT6800 Baseball</title>
  <meta charset="utf-8">
  <script src="jquery.js"></script>
  <script>
  var postData = {};
  postData["db"] = "dift6800_baseball";
  postData["query"] = "CC.yearId, CC.MaxCC, Master.nameFirst, Master.nameLast
    from Batting,
         (select yearID, max(HR) as MaxCC from Batting where lgID='NL' group by yearId having yearId>=1950)
               as CC,
         Master
    where Master.playerId=Batting.playerId and Batting.HR=CC.MaxCC AND Batting.yearId=CC.yearId
    order by CC.yearId ASC";
  $.post(
    "http://www-ens.iro.umontreal.ca/~dift6800/baseball/db.php",
    postData,
    function(reponse,status){
      console.log(status);
      console.log(response);
    }
  );
  </script>
</head>
<body>
  <h1>SQL en javascript.</h1>
</body>
...
```

La requête transmise ne contient pas le mot `select` au début. C'est de cette façon que le serveur est configuré.

Récupérer le résultat en javascript

- Le résultat de la requête est retourné dans la variable `reponse` de la fonction callback définie dans la requête POST.
  - Il s'agit d'une variable qui range une chaîne de caractères qui encode un tableau de tableau:



Récupérer les valeurs

- Il faut travailler un peu pour récupérer le résultat de la requête SQL à partir de la chaîne de caractères retournée.
- En premier lieu, nous pouvons créer un objet avec la méthode `JSON.parse(réponse)` qui produit un objet javascript avec les propriétés error et data de la requête SQL qui a produit la chaîne réponse:
  - `var obj = JSON.parse(reponse);`
  - `obj.error` est `"le message d'erreur"` et si aucune erreur alors `obj.error=""`
  - `obj.data` est $[\{'a_1':'e_1v_1','a_2':'e_1v_2',\ldots,'a_k':'e_1v_k'\},\ldots,\{'a_1':'e_rv_1','a_2':'e_rv_2',\ldots,'a_k':'e_rv_k'\}]$
- Pour aller chercher les données:
  - `var donnees = obj.data;` et nous avons
  - `donnees[i]` est $\{'a_{1'}:'e_iv_1','a_2':'e_iv_2',\ldots,'a_k':'e_iv_k'\}$
  - `donnees[i]` est un tableau associatif avec comme propriétés les attributs $a_1,a_2,\ldots,a_k$. La valeur de l'attribut a_j pour l'enregistrement i est:
    - `donnees[i][`a_j`]` est $"e_iv_j"$.

dbscript3.html

Charger les données dans un tableau

- Voici une fonction qui accepte `donnees=obj.data` et qui remplit un tableau entre les balises `<table id="id">...</table>` :

```
function genereTableau(donnees, id){
    var nb = donnees.length;
    var htmltable="<tr>";
    for(var attr in donnees[0]){
        htmltable=htmltable+"<th>"+attr+"</th>";
    }
    htmltable=htmltable+"</tr>";
    for(var x=0;x<nb;x++){
        htmltable=htmltable+"<tr>";
        for(var a in donnees[x]){
            htmltable=htmltable+"<td>"+donnees[x][a]+"</td>";
        }
        htmltable=htmltable+"</tr>";
    }
    $("#"+id).html(htmltable);
};
```

`htmltable` contiendra une chaîne avec le code html pour le tableau résultant

Le tableau contient un nom d'attribut pour chaque colonne

Pour chaque résultat: chaque ligne de la table

nouvelle ligne

la valeur de l'attribut a est ajoutée comme une nouvelle cellule.

Pour chaque attribut du x-ième résultat

fin de ligne

Remplie le tableau d'indice `id` avec la chaîne `htmltable`

dbscript3.html

Mettre les choses ensemble

- Voici maintenant le code qui fait la requête SQL et qui lance la fonction `genereTableau(donnees,"tableau")` avec les données produites par la requête SQL:

```
var postData = {};
postData["db"] = "dift6800_baseball";
postData["query"] = "CC.yearId, CC.MaxCC, Master.nameFirst, Master.nameLast from Batting, (select yearID,
max(HR) as MaxCC from Batting where lgID='NL' group by yearId having yearId>=1950) as CC, Master where
Master.playerId=Batting.playerId and Batting.HR=CC.MaxCC AND Batting.yearId=CC.yearId order by CC.yearId
ASC";
$.post(
    "http://www-ens.iro.umontreal.ca/~dift6800/baseball/db.php",
    postData,
    function(reponse,status){
      console.log(status);
      var obj = JSON.parse(reponse);
       if(obj.error==""){
            genereTableau(obj.data, "table");
       }else{
            alert("Erreur:"+obj.error);
       }
    });
```

Indique qu'il faut remplir la table avec `id="table"`.

Choisir un année à partir de laquelle les champions frappeurs sont donnés

- Ajoutons un sélectionneur d'années qui détermine la première année pour laquelle nous voulons les champions frappeurs de circuits. Les choix depuis 1980,1970, 1960 et 1950 sont offerts.
- Ajoutons un bouton "lancer la requête" qui lance la requête avec l'année sélectionnée.
- Les fonctionalités de JQuery vont nous permette de réaliser cette nouvelle page à l'aide de l'ancienne.
- Le titre de la section contenant le tableau sera également mis à jour avec l'année sélectionnée.
- Profitons-en pour ajouter les styles CSS `monstylesimple.css` et `tablestyle.css`. Le tableau et la page seront alors formatés conformément.

dbscript4.html

Le bouton, le sélecteur et le titre de section

Le sélectionneur avec id="annee"

L'identificateur pour le titre de section qui sera mis à jour.

Le bouton qui lance la requête.

```
<body>
  <h1 id="titre">Le plus de circuits par saison depuis 1950</h1>
  Choisir l'année à partir de laquelle vous voulez les
  champions cogneurs de circuits:
  <select id="annee">
       <option value="1950">1950</optio>
       <option value="1960">1960</option>
       <option value="1970">1970</option>
       <option value="1980">1980</pption>
  </select>
  <br>
  <button id="lance">Lance la requête</button>
  <br><br>
  <table id="table">
  </table>
</body>
```

dbscript4.html

Réagir au bouton

Lorsque le bouton est pressé alors le code de `function()` est exécutée.

```
var s = "CC.yearId, CC.MaxCC, Master.nameFirst, Master.nameLast
         from Batting,
            (select yearID, max(HR) as MaxCC from Batting where lgID='NL'
             group by yearId having yearId>=";
var r = ") as CC, Master where Master.playerId=Batting.playerId
                  AND Batting.HR=CC.MaxCC AND Batting.yearId=CC.yearId order by CC.yearId ASC";
$(document).ready(function(){
  $("#lance").click(function(event){
      var a = $("#annee").val();
      var h = "Le plus de circuits par saison depuis "+a;
      $("#titre").text(h);
       poste(s+a+r);
  });
});
```

obtient l'année sélectionnée par l'utilisateur

Le texte pour le titre de section mis à jour avec l'année sélectionnée.

Met à jour le titre de section avec l'année sélectionnée par l'utilisateur.

Fonction qui poste la requête au serveur roulant le script `db.php`

La requête SQL mise à jour avec l'année sélectionnée

dbscript4.html

La fonction qui poste la requête

Voici la fonction `poste(requete)` qui lance une requête entrée par l'usager dans une plage texte et qui imprime son résultat de la même façon que précédemment:

une fonction avec le code qui lançait la requête dans les exemples précédents. `requete` est une chaîne de caractères qui contient la requête SQL.

```
function poste(requete){
   var postData = {};
   postData["db"] = "dift6800_baseball";
   postData["query"] = requete;
   //La requête AJAX suit, faisant appel au backend db.php qui se trouve dans le même répertoire
   $.post(
        "http://www-ens.iro.umontreal.ca/~dift6800/baseball/db.php",
        postData,
        function(reponse,status){
          console.log(status);
          var obj = JSON.parse(reponse);
          if(obj.error==""){
               genereTableau(obj.data, "table");
          }else{
               alert("Erreur:"+obj.error);
          }
        }
    );
};
```

Modifier le rendu

dbscript5.html

N'imprime qu'une seule fois les années doublons

N'imprime qu'une seule fois le nbre de circuits pour les années doublons

```
function genereTableau(donnees, id){
    var nb = donnees.length;
     var nbattributs = donnees[0].length;
     var htmltable="<tr>";
     for(var attr in donnees[0]){
          htmltable=htmltable+"<th>"+attr+"</th>";};
     htmltable=htmltable+"</tr>";
     var doublon = "";
     var maxCC = true;
     for(var x=0;x<nb;x++){
          htmltable=htmltable+"<tr>";
          for(var a in donnees[x]){
               var dxa = donnees[x][a];
               if(a=="yearId"){
                    if(dxa==doublon){
                         dxa="";
                         maxCC = false;
                    }else{
                         doublon=dxa;
                         maxCC= true;}
               }
               if((a=="MaxCC") && !maxCC){
                    dxa="";}
               htmltable=htmltable+"<td>"+dxa+"</td>";
          }
          htmltable=htmltable+"</tr>";
    }
    $("#"+id).html(htmltable);
};
```

dbscript6.html

inalement: Une interface HTML à SQL

Voici le bout de code qui réagit au bouton en lançant la fonction `poste(requete)` avec la requête `requete` qui est écrite entre les balises `<textarea id="req"></textarea>` :

La même fonction `poste(requete)` que précédemment.

```
$(document).ready(function(){
     $("#lance").click(function(event){
          poste($("#req").val());
     });
 });
```

Un bouton avec `id="lance"` lance la requête inscrite dans le champs `textarea` avec `id="req"` sur un clic!

La requête inscrite dans le champs `textarea`.

dbscript6.html

Le corps de l'interface

```
<body>
  <h1>Entrez une requête baseball de votre choix</h1>
  <p>Par la suite, la requête ne doit pas contenir le mot select au début. Autrement dit, la
requête ne doit pas contenir le premier select, mais le reste de la requête est transmis
intégralement sans le ";" final.</p>

  Entrez votre requête SQL pour le baseball:<br>

  <textarea id="req" maxlength="5000" rows="10" cols="50">CC.yearId, CC.MaxCC,
Master.nameFirst, Master.nameLast  from Batting, (select yearID, max(HR) as MaxCC from
Batting where lgID='NL' group by yearId having yearId>=1950) as CC, Master  where
Master.playerId=Batting.playerId and Batting.HR=CC.MaxCC AND Batting.yearId=CC.yearId order
by CC.yearId DESC</textarea>

  <br>
  <button id="lance">Lance la requête</button>
  <br><br>
  <table id="table">
  </table>
</body>
```

Voici la plage texte avec la requête que nous avons vue

Le bouton qui lance la requête.