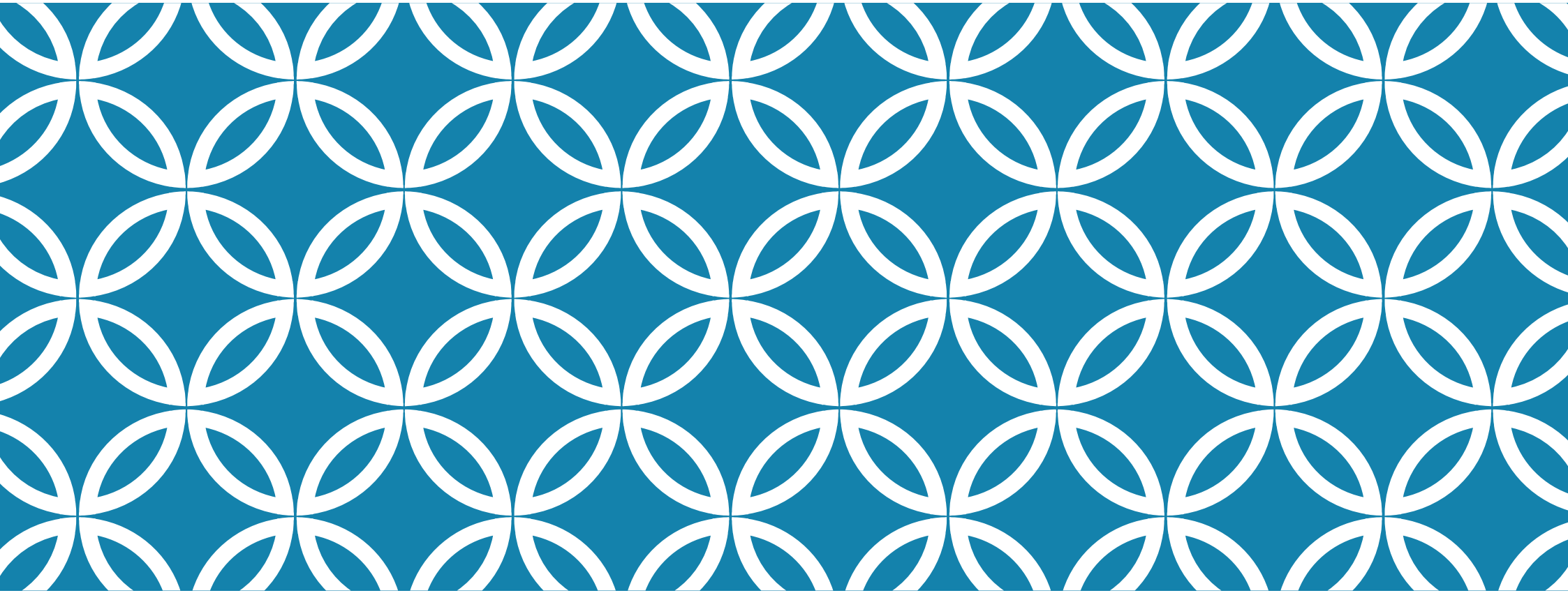


TECHNOLOGIE AJAX

Site Web dynamique

AJAX: (ASYNCHRONOUS JAVASCRIPT AND XML) ET UN CONCEPT PERMETTANT DE FAIRE DES APPELS ASYNCHRONES AU SERVEUR DEPUIS LE CLIENT. LORS DE CES APPELS, LE SERVEUR RETOURNERA DU XML QUI SERA « RÉCUPÉRÉ » PAR JAVASCRIPT ET TRAITÉ.

AINSI, ON PEUT ENVISAGER MODIfiER LE CONTENU D'UNE PAGE HTML LORS D'UNE INTERACTION DE L'UTILISATEUR (PAR EXEMPLE, LORS D'UN CLIC SUR UN BOUTON) SANS AVOIR À LA RECHARGER ENTIÈREMENT.

CECI PEUT ÊTRE UTILE LORSQUE L'ON VEUT EXÉCUTER UN SCRIPT PHP SANS RECHARGER LA PAGE EN ENTIÈRE (POUR PAR EXEMPLE INTERROGER LA BASE DE DONNÉES AU FUR ET À MESURE D'UNE SAISIE DANS UNE TEXTBOX).

EN EFFET, LE LANGAGE JAVASCRIPT EST EXÉCUTÉ DU CÔTÉ CLIENT ET NON SERVEUR, C'EST LÀ QUE TOUT SE JOUE, POUR EXÉCUTER UN SCRIPT PHP, LE CODE JAVASCRIPT FERA APPEL AU SCRIPT PHP, RÉCUPÈRERA LES INFORMATIONS RETOURNÉS PAR CELUI-CI ET LES AFFICHERA. CE CODE SERA EXÉCUTÉ DU CÔTÉ CLIENT, DONC AUCUN RECHARGEMENT DE LA PAGE

Asynchronous **J**avaScript **A**nd **X**ML

Ajax permet de modifier partiellement la page affichée par le navigateur pour la mettre à jour sans avoir à recharger la page entière.

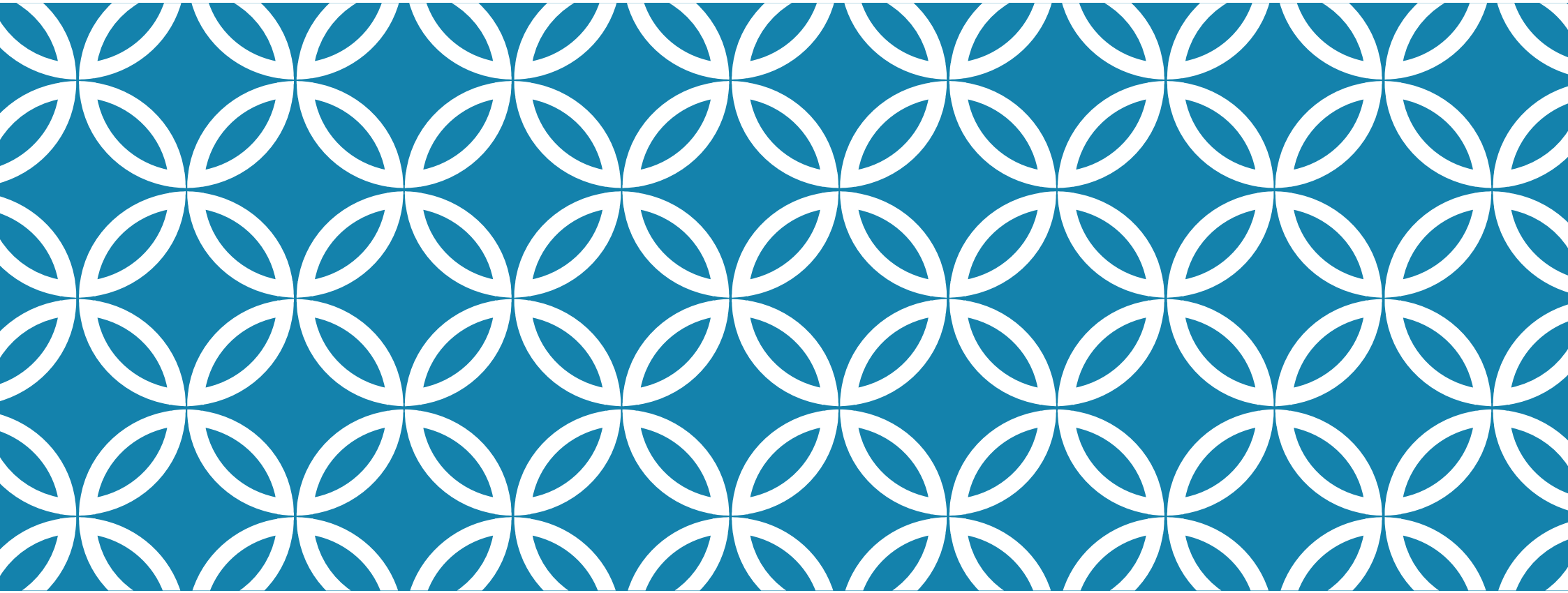
Utilisation ***conjointe*** d'un ensemble de technologies couramment utilisées sur le Web :

- HTML (ou XHTML) et CSS pour la mise en forme
- DOM et JavaScript pour afficher et interagir dynamiquement avec l'information présentée
- XML, XSLT et l'objet XMLHttpRequest pour échanger et manipuler les données de manière asynchrone avec le serveur web.
- L'objet XMLHttpRequest lit des données ou fichiers sur le serveur de façon asynchrone.

Application AJAX :

AJAX utilise l'objet XMLHttpRequest pour échanger des données avec un serveur. Cela signifie qu'il est possible de mettre à jour des parties d'une page Web, sans recharger la page entière.

- Envoyer des requêtes au serveur HTTP pour ne récupérer que les données nécessaires
- Utilisation la requête HTTP XMLHttpRequest
- Utilisation la puissance des feuilles de style (CSS) et de Javascript côté client pour interpréter et mettre en forme la réponse du serveur
- Permet au navigateur de modifier partiellement la page pour la mettre à jour sans avoir à la recharger
- Applications plus réactives, meilleure ergonomie



La propriété **readyState** contient le statut de XMLHttpRequest.

La propriété **onreadystatechange** définit une fonction à exécuter lorsque le readyState est modifié.

La propriété **status** et la propriété **statusText** contiennent le statut de l'objet XMLHttpRequest.

Fonctionnement

L'objet

XMLHttpRequest

- Attributs

Attributs	Valeurs possibles
readyState	0: non initialisé. 1: connexion établie. 2: requête reçue. 3: réponse en cours. 4: terminé.
Status	200 est ok 404 si la page n'est pas trouvée.
responseText	contient les données chargées dans une chaîne de caractères.
responseXml	contient les données chargées sous forme xml, les méthodes de DOM servent à les extraire.
Onreadystatechange	propriété activée par un évènement de changement d'état. On lui assigne une fonction.

```
var x = new XMLHttpRequest();
```

```
x.onreadystatechange = function() {
  if (x.readyState == 4 && x.status == 200) {
    // Code à executer quand la réponse est prête
  }
};
x.open("GET", "mesInfos.txt", true);
x.send();
```

Le « `readyState` » correspond au status de l'`XMLHttpRequest`. En particulier sa valeur sera :

- **0**, si la requête n'a pas (encore) été initialisée ;
- **1**, si une connexion au server a été établié ;
- **2**, si la requête a été réçue ;
- **3**, si la requête est en traitement et
- **4**, si la réponse à la requête est prête.

Le « `status` », lui, prend comme valeur :

- **200**, si tout va bien et
- **404**, si la page n'a pas été trouvée.

Pour acceder à la réponse du serveur on utilisera la méthode « `responseText()` ».

```
y = x.responseText;
```

responseText
Description : Réponse retournée par le serveur, au format texte.

responseXML
Description : Réponse retournée par le serveur, au format XML.

Fonctionnement ' objet equest

- **Méthodes**

Pour envoyer une demande à un serveur, nous utilisons les méthodes « open() » et « send() » de nôtre XMLHttpRequest:

```
x.open("GET", "monTexte.txt", true);
 x.send();
```

Méthodes	Paramètres
open(mode, url, boolean)	**mode:** type de requête, GET ou POST **url:** l'endroit ou trouver les données, un fichier avec son chemin sur le disque. **boolean:** true (asynchrone) / false (synchrone).
send("chaine")	null pour une commande GET.

```
$.ajax({
  type: 'GET', // Type de la requête : GET ou POST
  url: 'monTexte.txt', // URL à appeler
  dataType:'text' // S'il n'est pas spécifié, le navigateur va penser que le code
                  // est en XML
  success: function(responseText){ // Fonction de callback
  responseText // contient les données renvoyées (comme avant)
  }
});
```

CHARGER DU CONTENU

POSSIBILITÉ DE NE CHARGER QU'UNE PARTIE DU FICHIER (MÊME SI TOUT LE FICHIER EST RÉCUPÉRÉ DANS CE CAS, PUIS TRAITÉ POUR EN EXTRAIRE LA PARTIE VOULUE)

// version sans arguments :

// appelle fichier.html et met le contenu dans div

```
$("div").load("fichier.html");
```

// version avec arguments : appelle fichier.php en transmettant nom=Omar en POST

```
$("div#content").load("fichier.php", {"nom":« Omar"});
```

// version ne récupérant que l'objet d'id monid

```
$("div").load('test.html #monid');
```

AVEC GET ET POST

```
// récupère le fichier fichier.php
// puis exécute une fonction
// GET : paramètres dans l'URL :
$.get(
 "fichier.php",
{"nom":« Omar"},
function(data){
 alert(data);
});
```

AVEC GET ET POST

```
// POST : paramètres dans l'entête du message
// (caractères spéciaux, taille importante, etc.)
$.post(
"fichier.php",
{"nom":« Omar"},
function(data){
alert(data);
});
```

EXEMPLE

LE BUT DE CET EXEMPLE EST BASIQUE : ON CLIQUE SUR UN BOUTON, CELUI-CI UTILISE AJAX POUR EXÉCUTER UN SCRIPT PHP, LE RÉSULTAT EST AFFICHÉ SOUS FORME D'ALERT().

```
Xmlhttp=new XMLHttpRequest();
*/        function test(){
// On défini ce qu'on va faire quand on aura la réponse
          xmlhttp.onreadystatechange = function(){
// On ne fait quelque chose que si on a tout reçu et que le serveur est ok
if(xmlhttp.readyState == 4 && xmlhttp.status == 200)
{                          alert(xmlhttp.responseText);
              }          }
xmlhttp.open("GET",« f_ajax.php",true);
xmlhttp.send(null);          }
```



```
<input type='button' value='ajax!!' onclick='test()' />
```

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></
script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.get("demo_test.php", function(data, status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Envoyer une requête HTTP GET à une page et obtenir le résultat
</button>

</body>
</html>
```