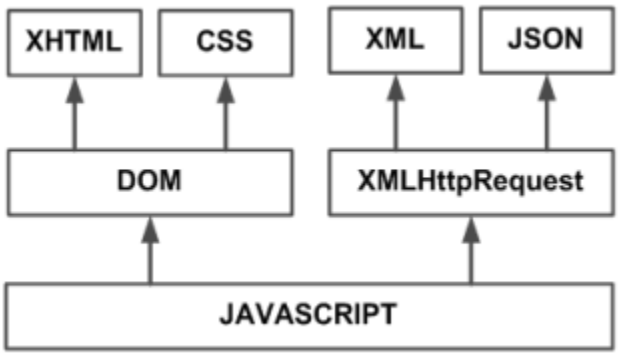
Introducció a AJAX

El terme AJAX és l'acrònim de **Asynchronous JavaScript And XML**, que es podria traduir per "JavaScript asíncron i XML"

No és una tecnologia per si mateixa sinó vàries tecnologies independents que s'uneixen per obtenir un comportament no esperat.

Les tecnologies que formen AJAX són:

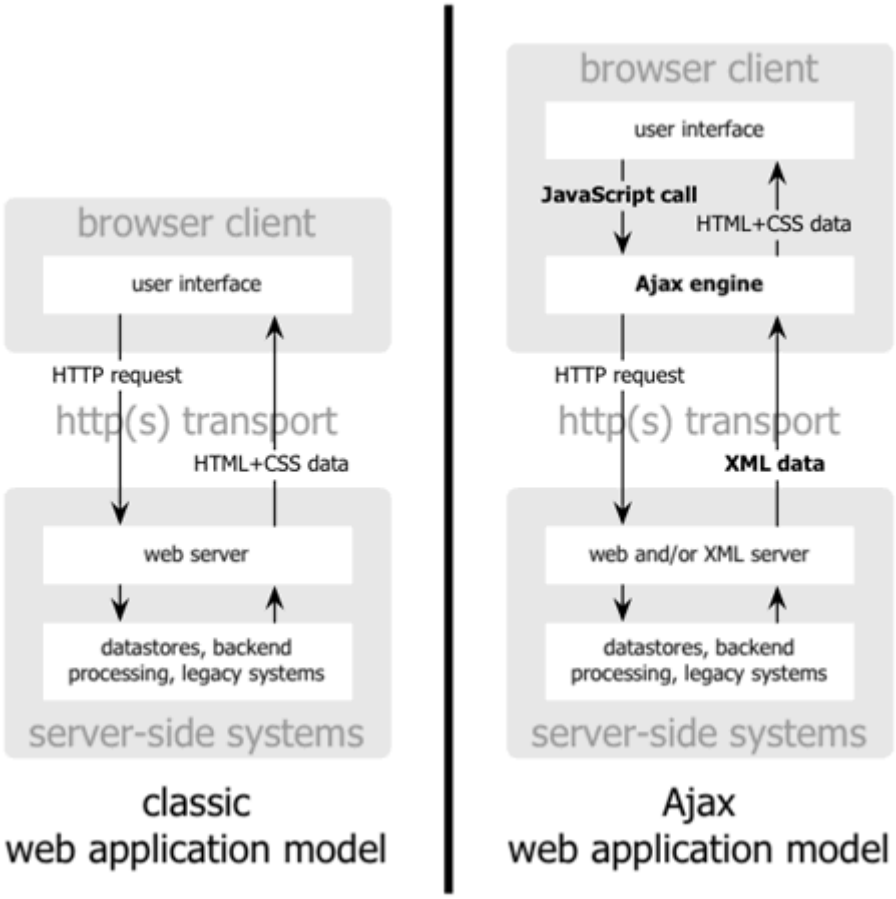
- **XHTML** i **CSS**, per crear una presentació basada en estàndards.
- **DOM**, per a la interacció i manipulació dinàmica de la presentació
- **XML**, **SXLT** i **JSON**, per a l'intercanvi i la manipulació de la informació.
- **XMLHttpRequest**, per a l'intercanvi asíncron d'informació
- **JavaScript**, per unir totes aquestes tecnologies.



Els desenvolupadors d'aplicacions AJAX requereixen un coneixement avançat de totes aquestes tecnologies.

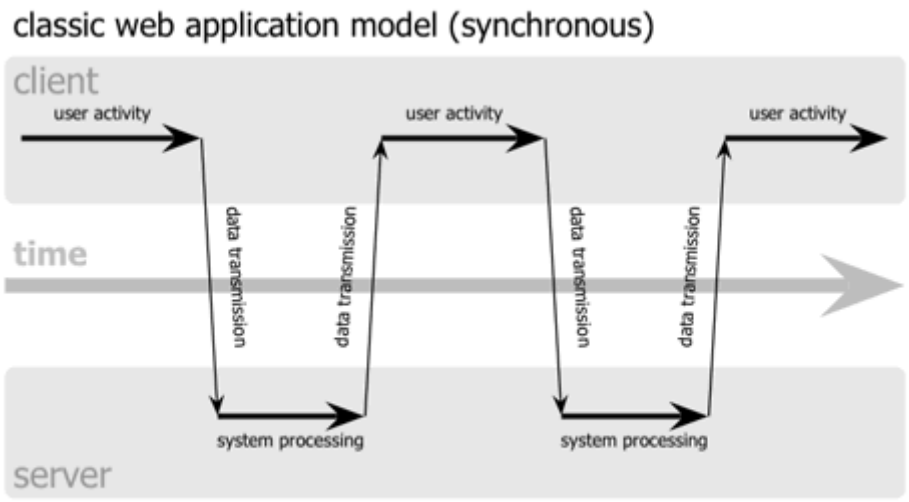
En aplicacions web tradicionals, les accions de l'usuari (fer clic en un botó, seleccionar un valor en una llista, etc) desencadenen crides al servidor. Una vegada processada la petició de l'usuari, el servidor retorna una nova pàgina HTML al navegador de l'usuari.

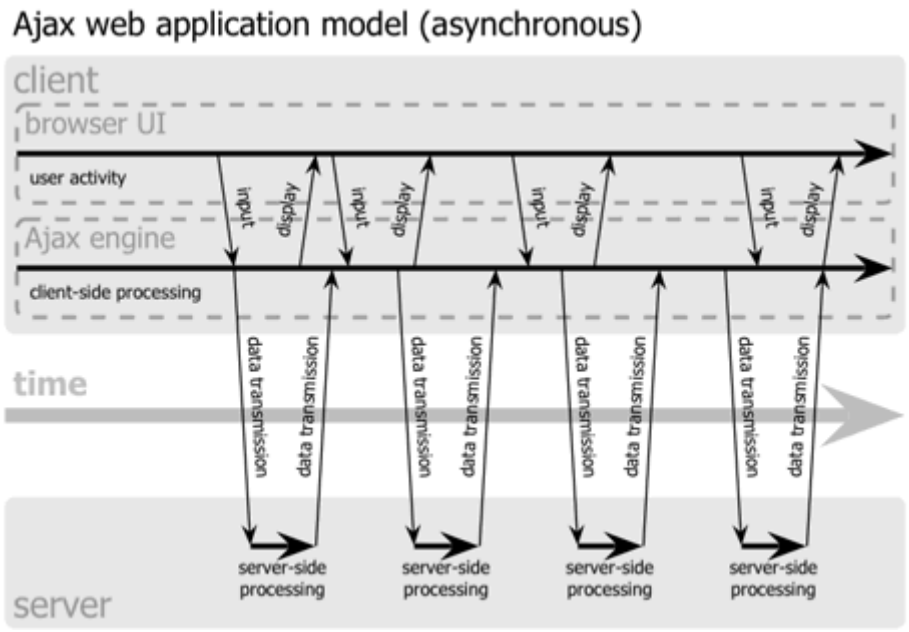
La següent imatge mostra la diferència de funcionament del model clàssic d'aplicacions web i el model proposat per AJAX.



La tècnica clàssica per crear aplicacions web funciona correctament, però no crea una bona sensació a l'usuari ja que quan es realitzen peticions al servidor, l'usuari ha d'esperar a que es recarregui tota la pàgina web amb els canvis demanats. Si l'aplicació ha de realitzar peticions contínuament, el seu ús pot esdevenir molt molest.

Amb la tecnologia AJAX, la interacció entre l'usuari i l'aplicació millora considerablement ja que s'eviten les recàrregues constants de la pàgina perquè l'intercanvi d'informació amb el servidor es realitza de manera asíncrona (és a dir, en segon pla). Les aplicacions construïdes en AJAX eliminen la recàrrega constant de pàgines mitjançant la creació d'un element intermedi entre l'usuari i el servidor. Aquesta nova capa (màquina Ajax) millora la resposta de l'aplicació i s'encarrega de tota la comunicació d'anada i tornada amb el servidor. Es pot comprovar que les peticions Http al servidor se substitueixen per peticions JavaScript que es fan a l'element encarregat d'AJAS. Aquestes peticions, són molt més simples i no requereixen la intervenció del servidor, amb la qual cosa la resposta és immediata. Si aquesta interacció requereix una resposta del servidor, aquesta es fa de manera asíncrona, amb la qual cosa la interacció de l'usuari tampoc no es veu interrompuda per haver de recarregar la pàgina o esperar la resposta del servidor.





Hi ha una llarga llista d'aplicacions basades an AJAX:

- **Gestors de correu electrònic**: Gmail, Yahoo mail, Windows Live Mail
- **Cartografía**: Google Maps, Yahoo Maps, Windows Live Local
- **Aplicacions web i productivitat**: Google Docs, Zimbra, Zoho
- **Altres**

Primers passos amb AJAX

Breu història d'AJAX

La història d'AJAX està relacionada amb un objecte de programació anomenat **XMLHttpRequest**. La primera vegada que es va fer servir aquest objecte va ser a l'any 2000, amb productes com l'Exchange 2000, Internet Explorer 5 i Outlook Web Access.

En 1998, **Alex Hopmann** i el seu equip desenvolupaven la futura versió d'Exchange 2000. El punt feble d'aquest servidor de correu electrònic era el seu client via web, anomenat OWA Outlook Web Access. Durant el desenvolupament del client web, es van avaluar dues opcions: un client format només per pàgines HTML estàtiques que es recarregaven constantment i un client realitzat tot amb HTML dinàmic o DHTML. Vistes les dues opcions, Hopmann es va decantar per la versió basada en DHTML, perquè a aquesta darrera opció li faltava alguna cosa que evités haver d'enviar contínuament els formularis amb dades al servidor.

Les primeres versions de la tecnologia, a la qual va anomenar XMLHTTP, van demostrar que eren un èxit, però de res serviria si els navegadors del moment no la incloïen. Malgrat que faltaven poques setmanes per el llançament de la darrera versió beta d'Internet Explorer 5 prèvia al seu llançament final, Alex va aconseguir que la seva tecnologia s'inclogués en la llibreria MSXML que inclou Internet Explorer.

Cal dir que el nom de l'objecte (XMLHTTP) es va triar com a excusa per incloure'l en la llibreria MSXML d'Internet Explorer, ja que aquest objecte està molt més relacionat amb HTTP que amb XML.

La meva primera aplicació AJAX

Farem en primer moment una petita aplicació que ens permeti descarregar un arxiu del servidor i mostrar-ne el contingut sense necessitat de recarregar la pàgina:

Partirem d'una pàgina bàsica que tindrem situada en el nostre servidor web amb les següents dades:

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>Salutacions des d'AJAX</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">


    </head>

    <body>
        <h1>Creació dinàmica d'elements com a resposta d'una petició AJAX</h1>
    </body>
</html>
```

A aquesta pàgina li afegirem el següent codi JavaScript:

```
<script type="text/javascript">
function descarregarArxiu() {
  /*Obtenim una instància de l'objecte XMLHttpRequest (depenent del navegador es fa com a
  new XMLHttpReques o com a new ActiveXObject("Microsoft.XMLHTTP")*/
  if(window.XMLHttpRequest) {
    peticio_http = new XMLHttpRequest();
  }
  else if(window.ActiveXObject) {
    peticio_http = new ActiveXObject("Microsoft.XMLHTTP");
  }

  // Preparem la funció de resposta (la que s'executarà cada vegada que canviï l'estat)
  peticio_http.onreadystatechange =   function() {
      console.log(peticio_http.readyState);
    if(peticio_http.readyState === 4) {
      if(peticio_http.status === 200) {
          var newP = document.createElement("p");
          newP.setAttribute("id","httpText");
          newP.innerHTML=peticio_http.responseText;
          document.body.appendChild(newP);
      }
    }
  };

  // Per últim realitzem la petició HTTP (Hem de )
  peticio_http.open('GET', 'http://localhost:8080/projecte/Salutacions.txt', true);
  peticio_http.send(null);
}
window.onload = descarregarArxiu;

</script>
```

Com funciona aquest codi?

Cada vegada que es carrega la pàgina s'executa el codi que descarrega l'arxiu Salutacions.txt que es troba al servidor i es mostra a la pàgina sense necessitat de recarregar-la tot. Una vegada s'ha recuperat el contingut de l'arxiu, només es genera un paràgraf nou, el carrega amb el text que hem recuperat de l'arxiu i l'afegeix al cos del nostre document.

Però analitzem amb més profunditat aquest codi:

Anàlisi detallat de l'aplicació

L'aplicació AJAX del nostre exemple es compon de quatre grans blocs:

- Instanciar l'objecte XMLHttpRequest.
    - Totes les aplicacions realitzades amb AJAX han d'instanciar en primer lloc l'objecte XMLHttpRequest, que és l'objecte clau que permet realitzar comunicacions asíncrones (en segon pla) amb el servidor, sense necessitat de recarregar tota la pàgina de nou.
    - La implementació de l'objecte XMLHttpRequest depèn de cada navegador, amb la qual cosa cal emprar una discriminació senzilla en funció de navegador en què s'executa el codi:
    - Si el navegador compleix amb els estàndards actuals crearem l'objecte com a un nou objecte del tipus XMLHttpRequest.
    - Si el navegador és obsolet, aquest objecte resultarà ser un objecte del tipus ActiveXObject (és molt estrany trobar-los actualment)

```
if(window.XMLHttpRequest) {
  peticio_http = new XMLHttpRequest();
}
else if(window.ActiveXObject) {
  peticio_http = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- Preparar la funció de resposta.
    - Una vegada hem obtingut la instància de l'objecte XMLHttpRequest, preparem la funció que s'encarrega de processar la resposta del servidor. Ho podem fer amb la propietat **onreadystatechange** de l'objecte, la qual ens permet indicar mitjançant una funció anònima o una referència a una funció independent definida quin és el codi que cal executar en funció de la resposta que ens doni el servidor.
    - El codi mostrat a continuació indica que, cada vegada que l'aplicació rebi una resposta del servidor ha d'executar la funció anònima que es mostra. Cal vigilar amb el codi d'aquesta funció perquè el servidor passa per diferents estats al llarg del procés, de manera que aquesta funció s'executarà diverses vegades. Ho veurem quan analitzem l'execució de la funció resposta.

```
peticio_http.onreadystatechange =   function() {
    console.log(peticio_http.readyState);
  if(peticio_http.readyState === 4) {
    if(peticio_http.status === 200) {
        var newP = document.createElement("p");
        newP.setAttribute("id","httpText");
        newP.innerHTML=peticio_http.responseText;
        document.body.appendChild(newP);
    }
  }
};
```

- Realitzar la petició al servior.
    - Una vegada hem preparat l'aplicació per a rebre la resposta del servidor, realitzem la petició HTTP al servidor.
    - En el nostre cas es tracta d'una petició senzilla de tipus GET, sense enviar cap paràmetre al servidor. Aquesta petició es crea mitjançant el mètode open en el qual incloem el tipus de petició (**GET**), quina URL estem sol·licitant (**'http://localhost:8080/projecte/Salutacions.txt'**) i un tercer paràmetre de tipus booleà amb el qual indiquem si aquesta petició serà asíncrona (**true**) o síncrona (**false**). En el nostre cas, volem fer la petició asíncrona, és per això que afegim un "**true**" al final
    - Una vegada creada la petició, l'enviem de facto al servidor mitjançant el mètode send(). És amb aquest mètode amb el que enviem els paràmetres necessaris per executar la petició. En el nostre cas, com que es tracta de descarregar un arxiu i no cal cap informació més, hi posem "**null**"

```
// Per últim realitzem la petició HTTP (Hem de )
peticio_http.open('GET', 'http://localhost:8080/projecte/Salutacions.txt', true);
peticio_http.send(null);
```

- Executar la funció de resposta.
    - Finalment, quan es rep la resposta del servidor, l'aplicació executa de forma automàtica la funció establerta anteriorment.
    - En primer lloc comprova que s'ha rebut la resposta del servidor. Això es fa mitjançant el valor de la propietat **readyState**. Com hem dit abans, aquesta propietat va variant al llarg de la petició :
        - UNSET (0): L'objecte XMLHttpRequest ha estat creat però encara no cridat el mètode open.
        - OPENED (1): La petició ha estat cridada (**open**)
        - HEADERS_RECEIVED (2): La petició ha estat enviada (**send**) i les capçaleres estan disponibles.
        - LOADING (3): Descarregant; l'atribut **responseText** conté dades parcials.
        - DONE (4): L'operació està completada.
    - Una vegada l'operació està completa, cal comprovar si la resposta ha estat vàlida i correcta (comprovant si el codi d'estat HTTP retornat és 200 (**OK**)). Altres possible valors de retorn són 403 (**Forbidden**), 404 (**Not Found**), etc. Podeu veure més codis d'error a: https://www.w3schools.com/tags/ref_httpmessages.asp
    - Si tot ha anat correctament, ja podem accedir a la resposta del servidor mitjançant la propietat **responseText**.

```
peticio_http.onreadystatechange =   function() {
    console.log(peticio_http.readyState);
  if(peticio_http.readyState === 4) {
    if(peticio_http.status === 200) {
        var newP = document.createElement("p");
        newP.setAttribute("id","httpText");
        newP.innerHTML=peticio_http.responseText;
        document.body.appendChild(newP);
    }
  }
};
```

Activitat 1:

Donada la pàgina web adjunta, afegir el codi JavaScript necessari per tal que:

1. Quan es carregui la pàgina, el quadre de text ha de mostrar per defecte la URL de la pròpia pàgina.
2. Quan es polsa el botó "Mostrar Contingut", ha de descarregar mitjançant peticions AJAX el contingut corresponent a la URL introduïda per l'usuari. El contingut de la resposta rebuda del servidor s'ha de mostrar en la zona de "Contingut de l'arxiu"

3. En la zona “Estats de la petició” ha de mostrar en tot moment l’estat en què es troba la petició (No inicialitzada, carregant, completada, etc)
4. Mostrar el contingut de totes les capçaleres de la resposta del servidor en la zona “Capçaleres HTTP de la resposta del servidor”
5. Mostrar el codi i text d’estat de la resposta del servidor en la zona “Codi d’estat”