Ajax

Ajax

- AJAX
  - Asynchronus JavaScript + XML
  - También a sido descrito como “Dynamic HTML and remote scripting”
  - Interfaces para clientes enriquecidos
    - Es más complicado que diseñar una página web
  - “Enriquecido” se refiere al modelo de interacción con el cliente
    - Puede soportar una amplia variedad de métodos de entrada
    - Puede responder intuitivamente y oportunamente
  - Web browser
    - Son clientes contactando a servidores web
    - Tiene algunas funcionalidades: botón “back”, history, tabs...

Ajax

- Páginas web
  - Se consideran como una aplicación
- 4 principios que definen AJAX
- (1) El browser recibe y mantiene una aplicación, no contenido
  - Aplicaciones clásicas
    - En las aplicaciones web clásicas el browser sólo es un “terminal tonto”
    - El servidor mantiene toda la información: sesión de usuario
    - El browser recibe los documentos y los despliega, cada vez que recibe uno nuevo hace lo mismo
  - Aplicaciones AJAX mueven lógica de la aplicación hacia el browser

Ajax

- (1) El browser recibe y mantiene una aplicación, no contenido
  - El browser recibe un documento más complejo, con bastante código Javascript
- (2) El servidor entrega datos, no contenido
  - El servidor solo envía datos relevantes, no repetitivos
  - Aplicación Ajax puede hacerlo por medio de trozos de código javascript, stream de texto o XML
  - Se reduce el uso de ancho de banda comparado con las aplicaciones tradicionales

Ajax

- (3) Interacción del usuario con la aplicación puede ser fluida y continua
  - Web browser provee dos mecanismos de entrada de datos: hyperlinks y formularios HTML
  - Mientras la página esta siendo enviada al servidor, el usuario está en el limbo
    - La antigua página se mantiene visible por mientras
    - El browser permite que el usuario siga haciendo click en los controles que se mantienen a la vista
  - Con Ajax se pueden conectar eventos para acciones del usuario
    - “Drag and drop” similar a las aplicaciones de escritorio
    - El servidor trabaja en conjunto con el usuario, ante cada evento que realiza en la interfaz

Ajax

- (4) Este es código real, necesita disciplina
  - Aplicaciones web clásicas limitan el uso de Javascript para validaciones y controles muy simples
    - Javascript ganó una reputación de trivial
  - Una aplicación Ajax es completamente diferente
    - El código generado es utilizado desde que el usuario lanza la aplicación hasta que la cierra
      - No debe bajar la performance ni memory-leaks
  - Se debe escribir código con alta perfomance y mantenible
    - Se debe usar la misma disciplina que en el desarrollo en el lado del servidor
  - Aplicación Ajax es una aplicación funcional compleja
    - Se comunica eficientemente con el servidor

Ajax

- Implementación
  - La clave es el objeto XMLHttpRequest
  - Actualmente todos los browser utilizan XMLHttpRequest, IE5 e IE6 utilizan ActiveXObject
  - Se debe averiguar que tipo de objeto soporta el browser
  - Este objeto realiza la comunicación con el servidor
  - Antes de enviar datos al servidor, se debe trabajar con las propiedades de XMLHttpRequest

Ajax

- Propiedades de XMLHttpRequest
  - onreadystatechange
    - Permite definir la función que recibirá y procesará los datos desde el servidor
    - Esta función es almacenada en esta propiedad para ser invocada automáticamente

```
xmlhttp.onreadystatechange=function()
{
    // We are going to write some code here
}
```

  - readyState
    - Mantiene el estado de la respuesta del servidor
    - Cada vez que cambia el estado de readystate, la función onreadystatechange es ejecutada

Ajax

- readyState: Posibles estados

State	Description
0	The request is not initialized
1	The request has been set up
2	The request has been sent
3	The request is in process
4	The request is complete

- Se necesita un if en onreadystatechange para analizar si la respuesta está completa:

```
xmlhttp.onreadystatechange=function()
{
if(xmlhttp.readyState==4)
  {
  // Get data from the server's response
  }
}
```

Ajax

- responseText
  - Los datos enviados desde el servidor se pueden recuperar desde el responseText:

```
xmlhttp.onreadystatechange=function()
{
if(xmlhttp.readyState==4)
  {
  document.myForm.time.value=xmlhttp.responseText;
  }
}
```

Ajax

- Enviando solicitudes al servidor
  - Se utiliza los métodos “send()” y “open()”
  - Método open recibe 3 argumentos:
    - Método para enviar la solicitud (GET o POST)
    - URL del programa que se ejecutará en el servidor
    - Indicar si el método se debe manejar de manera asíncrona

```
xmlhttp.open("GET","ajax.py",true);
xmlhttp.send(null);
```

    - Finalmente se debe decidir cuando la funcionalidad Ajax se debe ejecutar desde el HTML
      - Lo típico es cuando se gatilla algún evento

Ajax

- Aplicación con Flask

```
from flask import Flask, make_response, render_template
import subprocess

app = Flask(__name__)

@app.route('/ajax', methods = ['GET'])
def ajax():
    output = subprocess.run("/usr/bin/top -l 1", shell=True, capture_output=True, text=True)
    resp = make_response(output.stdout)
    resp.headers["Content-type"] = "text/plain; charset=UTF-8"
    resp.headers["Cache-Control"] = "no-cache"
    return resp

@app.route('/', methods = ['GET'])
def index():
    return render_template('index.html')

app.run(debug=True)
```

Ajax: sin dependencias

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function llamadaAjax() {
  document.getElementById("myDiv").innerHTML = "";
  document.getElementById("waiting").style.display = "block";
  xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange = function(){
      if (xmlhttp.readyState==4 && xmlhttp.status==200) {
          document.getElementById("waiting").style.display = "none";
          document.getElementById("myDiv").innerHTML = "<pre>"+xmlhttp.responseText+"</pre>";
      }
  }
  xmlhttp.open("GET","http://localhost:5000/ajax",true);
  xmlhttp.send();
}
</script>
</head>
<body>
<h2>Ejemplo de Ajax</h2>
<p>Listado de procesos</p>
<button type="button" onclick="llamadaAjax()">Mostrar listado!</button>
<div id="myDiv"></div>
<div id="waiting" style="display:none;"><img src="static/dog-waiting-gif.gif" alt="zzz"></div>
</body>
</html>
```

Ajax: dependencia remota

```
<!DOCTYPE html>
<html>
<head>
<script src="http://yui.yahooapis.com/3.6.0/build/yui/yui-min.js"></script>
<script type="text/javascript">
function llamadaAjax(){
YUI().use('node-load', function (Y) {
    // Replace the contents of the #content node with content.html.
    Y.one('#myDiv').load('http://localhost:5000/ajax');
});
}
</script>
</head>
<body>
<h2>Ejemplo de Ajax</h2>

<p>Listado de procesos</p>
<button type="button" onclick="llamadaAjax()">Mostrar listado!</button>
<pre id="myDiv"></pre>
</body>
</html>
```

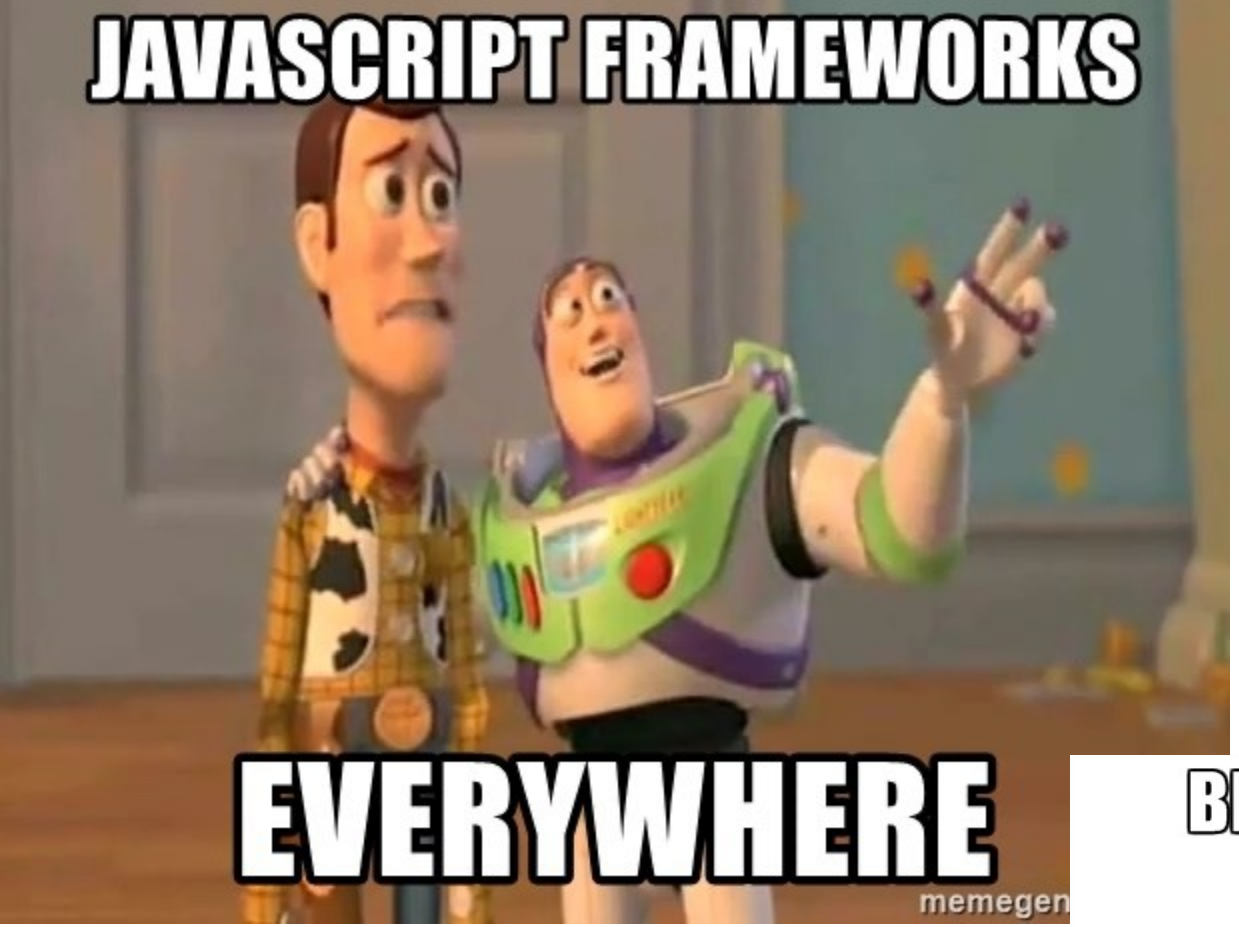
Ajax: dependencia local

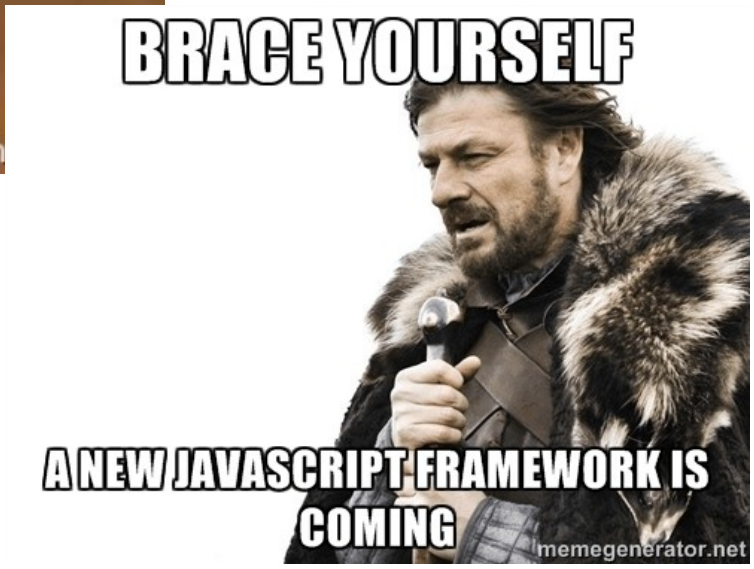
```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="static/jquery-3.5.1.min.js"></script>
<script type="text/javascript">
function llamadaAjax(){
        $.ajax({
                url: "http://localhost:5000/ajax"
        }).done(function(data) {
                $("#myDiv").empty();
                $( "#myDiv" ).append("<pre>"+ data +"</pre>");
        });
}
</script>
</head>
<body>
<h2>Ejemplo de Ajax</h2>

<p>Listado de procesos</p>
<button type="button" onclick="llamadaAjax()">Mostrar listado!</button>
<div id="myDiv"></div>
</body>
</html>
```

Ajax

- Ejemplos
  - W3schools:
    - https://www.w3schools.com/xml/tryit.asp?filename=tryajax_first
  - Google Maps
    - http://maps.google.com
- Herramientas
  - Vue, React, Angular, ...


JAVASCRIPT FRAMEWORKS
EVERYWHERE
memegen


BRACE YOURSELF
A NEW JAVASCRIPT FRAMEWORK IS COMING
memegenerator.net