AJAX

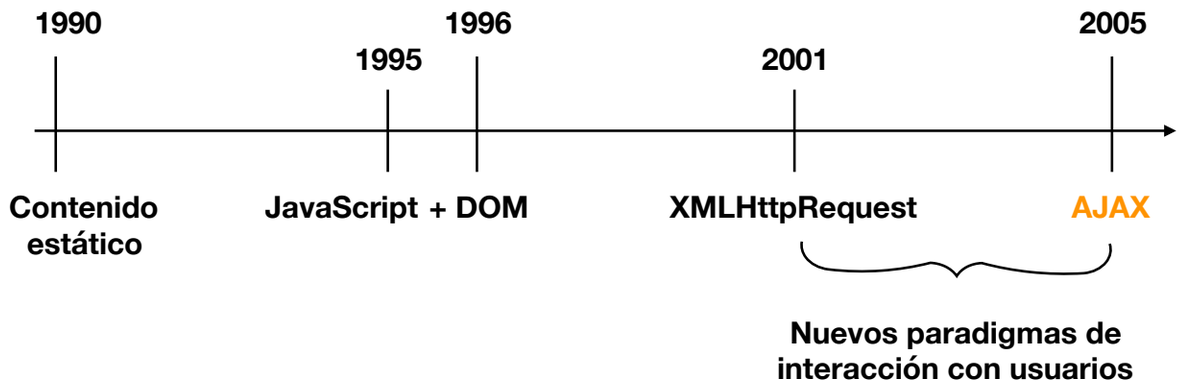
Laboratorio de Desarrollo Web

ITIC Yvone Sánchez Reyes

Agenda

- ¿Por qué Ajax?
- JSON + AJAX
  - JSON: objetos y arreglos
- AJAX: definición y componentes
- Ejemplo: *Permitir que AJAX modifique el contenido*
- ¿Qué es lo que sucede?
  - XMLHttpRequest: métodos y propiedades
  - Asíncrono vs. Síncrono
  - GET vs. POST
- Nota en Access Across Domains
- Recursos

¿Por qué AJAX?



Sitio web

App web

The Document Object Model (DOM) connects web pages to scripts or programming languages. Usually that means JavaScript, although modelling HTML, SVG, or XML documents as objects is not part of the JavaScript language, as such.
The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them you can change the document's structure, style, or content. Nodes can also have event handlers attached to them; once an event is triggered, the event handlers get executed. - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

Asynchronous JavaScript And XML

AJAX (Asynchronous JavaScript And XML)

JavaScript y XML Asíncronos

- No es un lenguaje de programación
- Es un modelo de interacción con usuarios
- Enviar o recibir información sin recargar la página

w3schools. *AJAX Introduction*, edited by Refsnes Data, 2 Mar. 2007, https://www.w3schools.com/js/js_ajax_intro.asp. Accessed 5 Mar. 2018.

JSON + AJAX

Java
Script
Object
Notation

JSON (JavaScript Object Notation)

Notación de Objectos de JavaScript

- Formato ligero de intercambio de datos
- Enviar, recibir y empaquetar información

```
1
2
3   var persona = {
4       "nombre" : "Marinette",
5       "apellido" : "Dupain-Cheng"
6   }
7
8   var gustos = ["coser", "Adrien", "parkour"];
9
10  var personas = [
11      {
12          "nombre" : "Marinette",
13          "apellido" : "Dupain-Cheng"
14      },
15      {
16          "nombre" : "Adrien",
17          "apellido" : "Agreste"
18      }
19  ];
20
21
22
```

Objetos

Arreglos

AJAX

- **XMLHttpRequest**, objeto incorporado en el navegador para solicitar datos del servidor
- **JavaScript** + **DOM**, para mostrar o usar datos
- XML, texto plano o **JSON**, para transportar datos

w3schools. *AJAX Introduction*, edited by Refsnes Data, 2 Mar. 2007, https://www.w3schools.com/js/js_ajax_intro.asp. Accessed 5 Mar. 2018.

Use XMLHttpRequest (XHR) objects to interact with servers. You can retrieve data from a URL without having to do a full page refresh. XHR can be used to retrieve any type of data, not just XML, and it supports protocols other than HTTP (including file and ftp).
If your communication needs to involve receiving event data or message data from a server, consider using server-sent events through the EventSource interface. For full-duplex communication, WebSockets may be a better choice. - https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

```html
<!DOCTYPE html>
<html>
<head>
    <title>JSON y AJAX</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
</head>
<body>
    <header>
        <h1>JSON y AJAX</h1>
        <h3>Permitir que AJAX modifique el contenido</h3>
        <button id="btn" type="button" onclick="loadData()">Mostrar
        información sin recargar la pantalla</button>
    </header>

    <div id="loaded-info"></div>

    <script type="text/javascript">
        function loadData() {
            // crear una petición
            // procesar la petición
            // modificar contenido HTML
            // enviar petición
        }
    </script>
</body>
</html>
```

AJAX

```html
<script type="text/javascript">
    function loadData() {
        // crear una petición
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "https://yvone.github.io/DAW/info-1.json");

        // procesar la petición
        xhr.onload = function() {
            changeText(this);
        };
        // enviar petición
        xhr.send();
    }

    function changeText(response) {
        // modificar contenido
        var data = JSON.parse(response.responseText);
        console.log( data[0] );
    }
</script>
```

JSON.parse

```
function changeText(response) {
    var contenedor = document.getElementById("loaded-info");
    var htmlString = "";

    // modificar contenido
    var data = JSON.parse(response.responseText);
    data.forEach(function(persona) {
        htmlString += '<p>' + persona.apellido + ' ' + persona.nombre
            + ' nació el ' + persona.fechaNac;

        htmlString += ' y le gusta ' + persona.perfil.gustos.join(" y
            ");

        htmlString += ' y no le gusta ' + persona.perfil.disgustos.
            join(" ni ");

        htmlString += '.</p>'
    });

    contenedor.insertAdjacentHTML('beforeend', htmlString);
}
```

```
function loadData() {
    // crear una petición
    var xhr = new XMLHttpRequest();
    var btn = document.getElementById("btn");

    xhr.open("GET", "https://yvone.github.io/DAW/info-1.json");

    // procesar la petición
    xhr.onload = function() {
        changeText(this);

        btn.classList.add("bye-bye");
    };
    // enviar petición
    xhr.send();
}
```

Mensajes de estado HTTP

1xx: Información
2xx: Exitoso

200: "OK"

3xx: Re dirección
4xx: Error en cliente

403: "Forbidden"
404: "Not Found"

5xx: Error en servidor

503: "Service Unavailable”

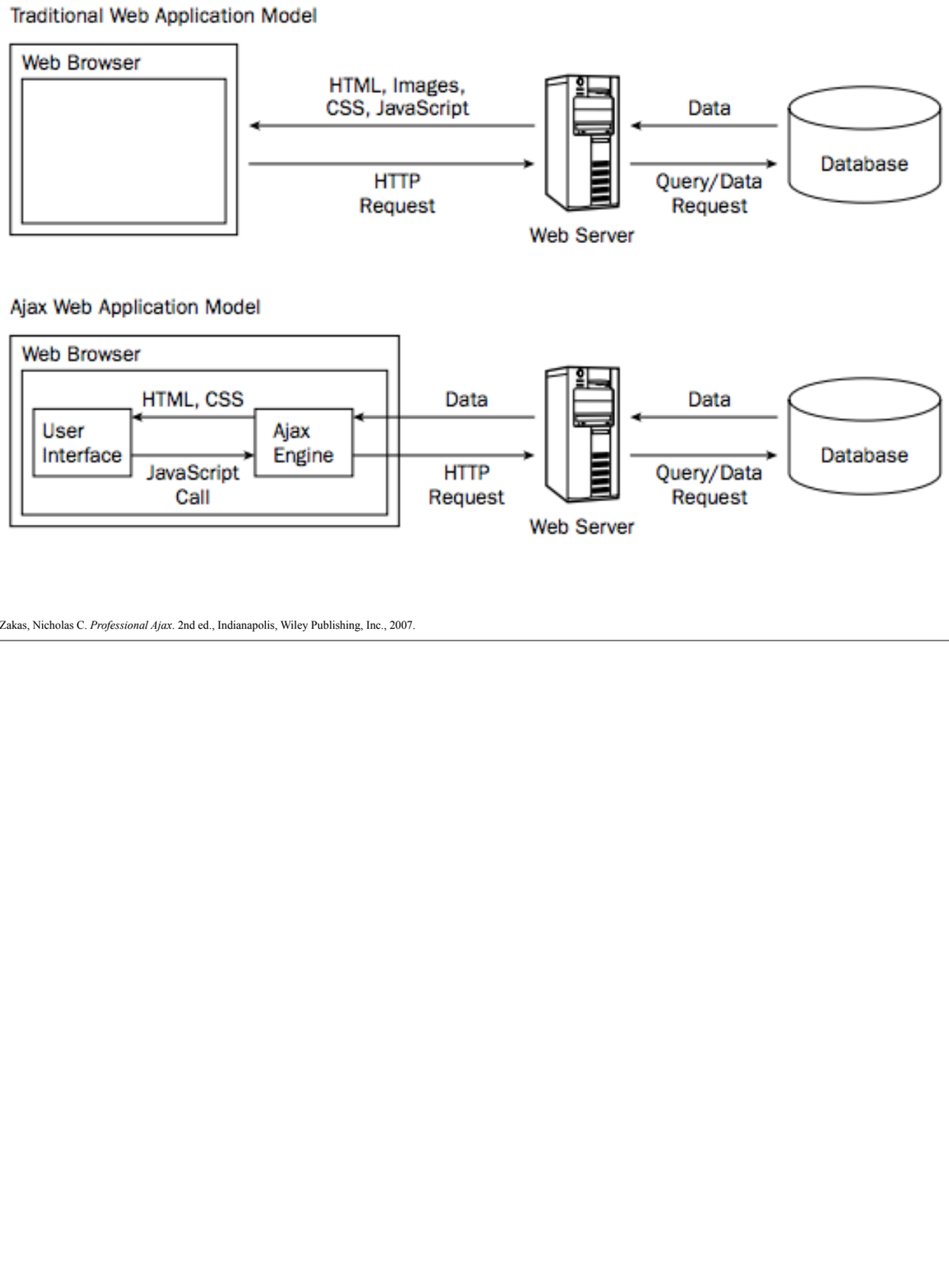
"HTTP response status codes." *MDN web docs*, mozilla, 2 Mar. 2018, https://developer.mozilla.org/en-US/docs/Web/HTTP/status. Accessed 5 Mar. 2018.

```js
function loadData() {
    // crear una petición
    var xhr = new XMLHttpRequest();
    var btn = document.getElementById("btn");

    xhr.open("GET", "https://yvone.github.io/DAW/info-1.json");

    // procesar la petición
    xhr.onload = function() {
        if(this.readyState == 4 && this.status == 200) {
            changeText(this);

            btn.classList.add("bye-bye");
        } else {
            alert("Error " + this.status);
        }
    };

    xhr.onerror = function() {
        alert("La conexión ha fallado");
    };

    // enviar petición
    xhr.send();
}
```

HTTP status

¿Qué es lo que sucedió?

1. Un evento ocurre
2. Crear un objeto XMLHttpRequest
3. Enviar una petición HTTP al servidor
4. Procesar la petición HTTP
5. Crear una respuesta HTTP y mandar la información al navegador
6. Procesar la respuesta HTTP
7. Actualizar el contenido de la página



Zakas, Nicholas C. *Professional Ajax*. 2nd ed., Indianapolis, Wiley Publishing, Inc., 2007.

Objeto XMLHttpRequest

Enviar una petición al servidor y procesar una respuesta

```
17          function loadData() {
18              var xhr = new XMLHttpRequest();
19              var btn = document.getElementById("btn");
20
21              xhr.open("GET", "https://yvone.github.io/DAW/info-1.json");
22
23              xhr.onload = function() {
24                  if(this.readyState == 4 && this.status == 200) {
25                      changeText(this);
26                      btn.classList.add("bye-bye");
27                  } else {
28                      alert("Error " + this.status);
29                  }
30              };
31
32              xhr.onerror = function() {
33                  alert("La conexión ha fallado");
34              };
35
36              xhr.send();
37          }
```

2

6

7

3

Kesteren, Anne V., Julian Aubourg, 송정기, and Hallvord R. Steen. "XMLHttpRequest Level 1." 2016, https://www.w3.org/TR/XMLHttpRequest/. Accessed 5 Mar. 2018.

Métodos

Método	**Descripción**
new XMLHttpRequest()	Crea un nuevo XMLHttpRequest object
abort()	Cancela la petición actual
open(*method,url,async,user,psw*)	Especifica la petición *method*: el tipo de petición: GET o POST *url*: indica la ubicación del archivo *async*: true (asíncrona) or false (síncrona) *user*: nombre de usuario, opcional *psw*: contraseña, opcional
send()	Envía la petición al servidor. Utilizado para peticiones tipo GET
send(*string*)	Envía la petición al servidor. Utilizado para peticiones tipo POST
setRequestHeader(header, value)	Agrega encabezados HTTP a la petición *header*: especifica el nombre del encabezado *value*: especifica el valor del encabezado

Propiedades

Propiedad	Descripción
onreadystatechange	Define una función que será ejecutada cada que la propiedad readyState cambie Nota: se ejecuta cuatro veces, una por cada cambio en readyState (1-4)
readyState	Almacena el estado de la petición XMLHttp. 0: petición no inicializada 1: conexión establecida con el servidor 2: petición recibida 3: procesando petición 4: petición terminada y respuesta lista
status	Regresa el estado (número) de la petición 200: "OK" 403: "Forbidden" 404: "Not Found”
statusText	Regresa el estado (texto) Ej. “OK”, “Not Found”
responseText	Regresa la respuesta como string
responseXML	Regresa la respuesta como XML

Asíncrono vs. Síncrono

- **Asíncrono:** Permite que JavaScript no espere la respuesta del servidor y pueda ejecutar otros scripts y ocuparse de la respuesta cuando ya este lista.

onReadyStateChange

- **Síncrono:** No son recomendadas pues JavaScript tendrá que dejar de ejecutar cualquier otro script y esperar que la respuesta del servidor este lista. Sí el servidor esta ocupado o es lento, la aplicación puede congelarse o detenerse.

tests rápidos

GET vs. POST

GET

```
1  xhttp.open("GET", "ajax_info.txt", true);
2  xhttp.send();

...

40 xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
41 xhttp.send();
```

POST

```
1  xhttp.open("POST", "ajax_test.asp", true);
2  xhttp.setRequestHeader("Content-type", "application/x-www-form-
   urlencoded");
3  xhttp.send("fname=Henry&lname=Ford");
```

```
Content-Type: text/html; charset=utf-8
Content-Type: multipart/form-data; boundary=something
Content-Type: application/x-www-form-urlencoded
```

GET vs. POST

Obtener o Modificar

GET es más simple y rápido sin embargo se debe usar POST cuando:

- No se desea consultar información en caché y se desea actualizar un archivo o base de datos en servidor
- Se envía una gran cantidad de datos al servidor. POST no tiene límite en tamaño
- Se envía una entrada (input) del usuario que puede contener caracteres especiales. POST es más robusto y seguro que GET

Access Across Domains

Por seguridad, los navegadores actuales no permiten acceso a través de dominios ajenos. Es decir, la página web y el archivo xml/texto plano/json deben estar en el mismo servidor.

Nota: Para consultar archivos de servidores externos necesitamos “pedir permiso” y “obtener permiso” del servidor.

Hossain, Monsur. "Using CORS." *HTML5 Rocks*, 29 Oct. 2013, https://www.html5rocks.com/en/tutorials/cors/. Accessed 5 Mar. 2018.

Zakas, Nicholas C. "Cross-domain Ajax with Cross-Origin Resource Sharing." *NCZOnline*, 25 May 2010, https://www.nczonline.net/blog/2010/05/25/cross-domain-ajax-with-cross-origin-resource-sharing/. Accessed 5 Mar. 2018.

Recursos

- Nicholas Zakas, *Professional Ajax*
- W3org https://www.w3.org/TR/XMLHttpRequest/
- MDN Web docs https://developer.mozilla.org/
- W3 w3schools.com
- Google project, HTML5 Rocks https://www.html5rocks.com/en/tutorial/cors/