TEMA 5:

TECNOLOGIES WEB

EI1042 - TECNOLOGÍAS Y APLICACIONES WEB

EI1036 - TECNOLOGÍAS WEB PARA LOS SISTEMAS DE INFORMACIÓN

(2022/2023)

PROFESSORA: DRA. DOLORES Mª LLIDÓ ESCRIVÁ

Universitat Jaume I.

ÍNDICE

1. HTTP ASÍNCRONO

MODELO ASÍNCRONO

Asincronía: acción que no tiene lugar en total correspondencia temporal con otra acción. (Fuente: Wikipedia).

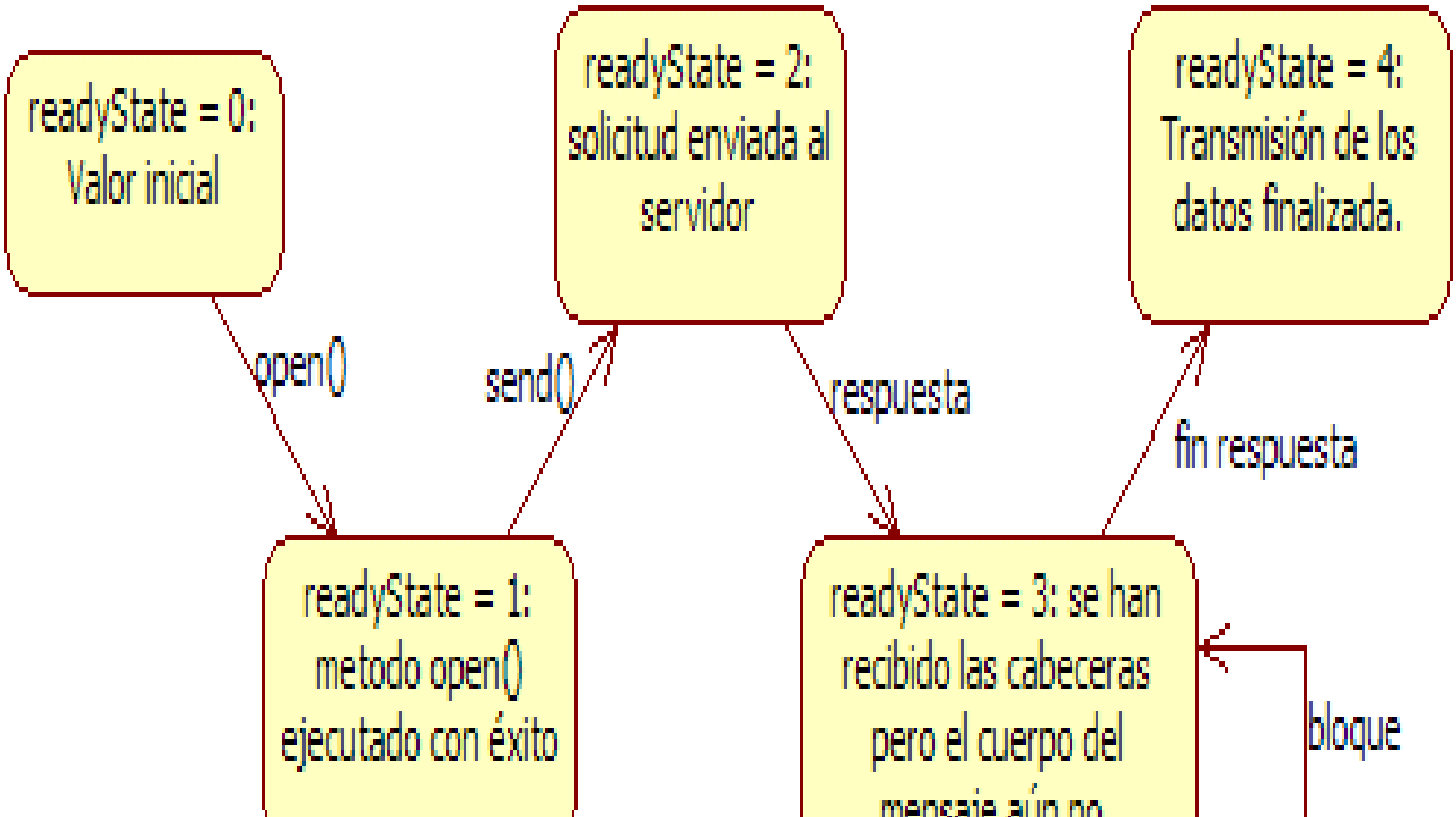
¿Ejemplo de asincronía?

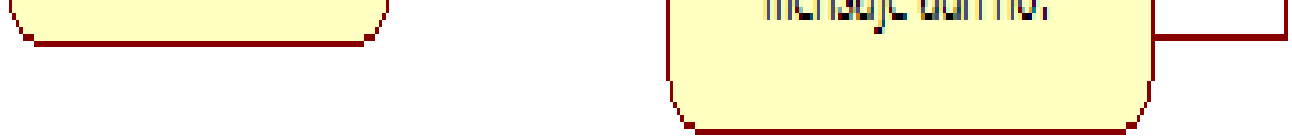
AJAX: JESSE JAMES GARRETT

- El término AJAX se presentó en 2005 por primera vez en el artículo "Ajax A New Approach to Web Applications"
- AJAX es un acrónimo de **Asynchronous JavaScript + XML** , que se puede traducir como "JavaScript asíncrono + XML".

ESTADOS DEL MOTOR DE AJAX

API XMLHttpRequest





INTERACCIÓN

- AJAX mejora la interacción del usuario con la aplicación, evitando las recargas constantes de la página.
- En el **cliente** requiere un motor del Ajax que se conecta con el servidor de forma asíncrona,(API XMLHttpRequest).
- En el **servidor** no se requiere nada en particular, simplemente debe devolver el recurso que se le solicita.

EJEMPLO: PETICIÓN URL SÍNCRONA CON AJAX

```
var Ajax1=new XMLHttpRequest();
Ajax1.open(“GET”,” http://www.example.com”);
Ajax1.send();
node=document.getElementByID("central");
node.innerHTML=Ajax1.responseText;
```

- ¿Hay algun problema si es síncrono?
- ¿Y si es asíncrono?

EJEMPLO: CARGA HIPERENLACES DE FORMA ASÍNCRONA

https://asociacionpiruleta.cloudaccess.host/PHP/T5/AsincroAjax.html

```
function cargaAjax(src_url, l) {
  var Ajax1 = new XMLHttpRequest();
  Ajax1.addEventListener("readystatechange", function () {
    if (this.readyState === 4) {
      if (this.status < 400) {
        l.innerHTML = Ajax1.responseText;
      }
    }
  });
  Ajax1.open("GET", src_url);
  Ajax1.send();
}
function ready() {
  enlace = document.querySelector("nav a");
  src_url = enlace.getAttribute("href");
  enlace.addEventListener("click", function (event) {
    event.preventDefault();
    cargaAjax(event.target.src, event.target.parentElement);
  });
}
document.addEventListener("DOMContentLoaded", function () {
  ready();
});
```

HTTP ASÍNCRONO

Soluciones en Javascript:

- API XMLHttpRequest: Eventos. Motor AJAX
- API FETCH: Promesas.

PROMESAS- PROMISE

- Las promesas son un Patrón de diseño para controlar la ejecución de un determinado cómputo del cual no sabemos: ni cómo, ni cuándo se nos va a devolver un determinado valor.
- Una promesa es un objeto que por medio de una máquina de estados podemos controlar cuándo un valor está disponible o no.
- Los métodos de las promesas devuelven promesas, permitiendo que las promesas se puedan encadenar.
- EmacScript 7

MÉTODOS PROMESAS

promise = new Promise(function(resolve[, reject]) {});

- promise.resolve (Obligatorio). Si la promesa devuelve el valor deseado , se ejecuta este método.
- promise.reject (Opcional). Método que se ejecuta cuando la promesa no devuelve el valor deseado.

Para encadenar promesas en caso de éxito o fracaso tenemos estos métodos:

- promise.then(onFulfilled,onRejected) : Método para indicar que hacer cuando una promesa devuelve el valor deseado. **Resolve**
- promise.catch(onRejected): Método para indicar que hacer cuando una promesa devuelve el valor no deseado. **Reject**

EJEMPLO DEFINICIÓN Y USO DE UNA FUNCIÓN QUE ES UNA PROMESA

```
function isDinnerTime() {
  return new Promise(function (resolve, reject) {
    setTimeout(function () {
      const now = new Date();
      if (now.getHours() >= 22) {
        resolve("yes");
      } else {
        reject("no");
      }
    }, 1000);
  });
}
isDinnerTime()
  .then((data) => console.log("success: " + data))
  .catch((data) => console.log("error: " + data));
```

2. API FETCH

- es una API más simple y limpia que XHTMLRequest.
- No envia ni recibe ninguna **cookie**.
- Utiliza las promesas de JavaScript.
- El objeto **Fetch** permite solicitar recursos definidos y devuelve un objeto **Response** que es una promesa.

fetch(resource[, initRequest])*

FETCH GET

```
fetch("./api/some.json")
  .then(function (response) {
    if (response.status !== 200) {
      console.log(
        "Looks like there was a problem. Status Code: " + resp
      );
      return;
    } // Examine the text in the response
    response.json().then(function (data) {
      console.log(data);
    });
  })
  .catch(function (err) {
    console.log("Fetch Error :", err);
  });
```

https://asociacionpiruleta.cloudaccess.host/PHP/T5/AsincroFetch.html

`response.status !== 200` Es un error. ¿no se debería recogerse con catch?

OBJETO FETCH

- resource(recurso de la petición):
  - una cadena con la **URL** .
  - o un objeto **Request**. Objeto que contiene los datos para la solicitud de un recurso para la petición Fetch.
- initRequest: Objeto JSON de inicialización de la petición, cuyos valores sobrescriben los que existen en el objeto Request. Es optativo. Estos parámetros pueden ser:
  - Cadena **method**: request HTTP method:Por defecto GET.
  - Objeto **headers**: request HTTP headers.
  - Objeto **body**: request HTTP body.
  - Cadena **mode**: cors, no-cors, same-origin, navigate.

OBJETO REQUEST

request = new Request([recurso][,requestInit]])

```
const request = new Request('https://www.mozilla.org/favicon.i

const url = request.url;
const method = request.method;
const credentials = request.credentials;
```

OBJETO HEADERS

- **Headers** : Objeto que Representa los encabezados de la respuesta/solicitud, lo que le permite consultar y tomar diferentes acciones en función de los resultados.

```
var myHeaders = new Headers();
myHeaders.append('Content-Type', 'text/xml');
myHeaders.get('Content-Type') // should return 'text/xml'
```

OBJETO BODY

Objeto proporciona métodos relacionados con el contenido de la respuesta/solicitud, lo que le permite declarar cuál es su tipo y cómo debe manejarse.

```
body: "foo=bar&lorem=ipsum";
```

RESPONSE

- Representa la respuesta a una solicitud.
- Es la promesa que devuelve fetch.
- Devolverá un error sólo cuando hay un error de red. No el código de error HTTP como 404 o 500.

Métodos:

- response.ok : true (false) si el estado esta entre 200-299.
- response.status: Código HTTP de respuesta.

Tipos de respuesta:

- response.arrayBuffer(): El objeto ArrayBuffer se usa para representar un buffer genérico, de datos binarios crudos (raw) con una longitud específica
- response.blob():Un objeto Blob representa un objeto tipo fichero de datos planos inmutables
- response.json()
- response.text()

FETCH: REQUEST GET BLOB

Ejemplo: Carga una imagen

https://asociacionpiruleta.cloudaccess.host/PHP/T5/ImageSend.html

```
function asyncCall() {
  var myImage = document.querySelector("#mi_imagen");
  fetch("https://upload.wikimedia.org/wikipedia/commons/7/77/
    .then(function (response) {
      if (!response.ok) {
        throw new Error(response.statusText);
      }
      response.blob().then(function (data) {
        var objectURL = URL.createObjectURL(data);
        myImage.src = objectURL;
      });
    })
    .catch(function (error) {
      console.log(error);
    });
}
```

- ver carga imagen asíncrona
- ver envio formulario asíncrono

3. SERVICIO WEB

Los Servicios Web son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web y que intercambian datos entre sí con el objetivo de ofrecer servicios.

Podemos clasificar los servicios web en:

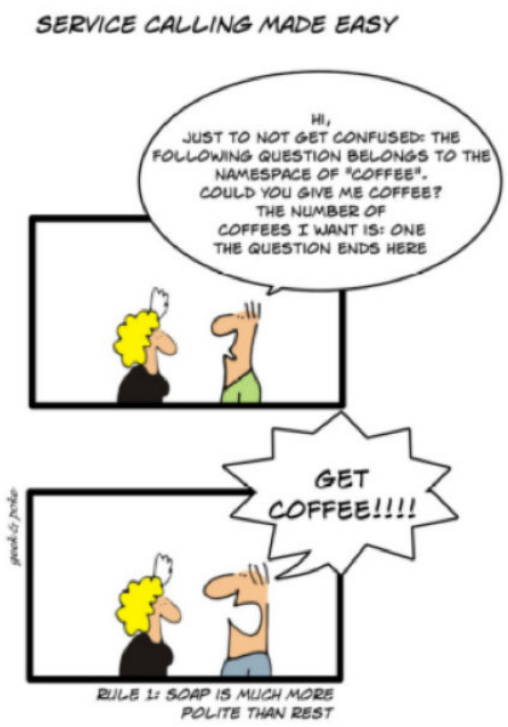
- Servicios Web **REST**, el propósito principal del servicio es manipular las representación de los recursos WEB (XML,JSON) usando un conjunto uniforme de operaciones sin estado(Métodos).
- Servicios Web **SOAP**: en el cual el servicio se expone con un conjunto arbitrario de operaciones.(XML)

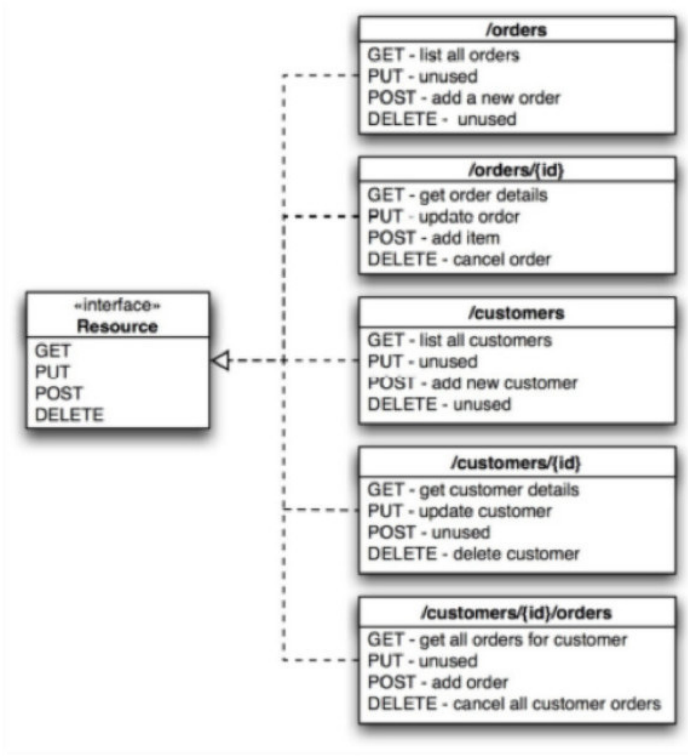
REST API O SERVICIO WEB REST

Transferencia de Estado Representacional

Los sistemas REST utilizan estos métodos para identificar operaciones sobre los distintos recursos

- GET (read): Se utiliza para consultar, leer y en definitiva acceder a un recurso
- POST (create): Envía datos para crear un recurso.
- PUT (update): Utilizado para editar un recurso.
- DELETE (delete): Es la opción para eliminar un recurso





(http://geek-and-poke.com/)

WORDPRESS RESTAPI

http://v2.wp-api.org/ https://developer.wordpress.org/rest-api/

Existe un Endpoint especial, llamado wp-json, al que se accede mediante la ruta raíz o inicial y que devuelve toda la información relativa a la API REST

https://asociacionpiruleta.cloudaccess.host/wp-json/

Guia de referencia: http://v2.wp-api.org/reference/posts/

https://asociacionpiruleta.cloudaccess.host/wp-json/wp/v2/posts

https://asociacionpiruleta.cloudaccess.host/wp-json/wp/v2/posts/2

La api Rest de Wordpress permite realizar operaciones CRUD pueden afectar a cualquiera de los elementos de información de nuestro sitio WordPress:

- Entradas
- Páginas
- Usuarios
- Multimedia
- Taxonomías
- CustompostTypes

RUTAS BASE

Elemento	Ruta base
Posts	/wp/v2/posts
Post Revisions	/wp/v2/revisions
Categories	/wp/v2/categories
Tags	/wp/v2/tags
Pages	/wp/v2/pages

Elemento	Ruta base
Comments	/wp/v2/comments
Taxonomies	/wp/v2/taxonomies
Media	/wp/v2/media
Users	/wp/v2/users
Post Types	/wp/v2/types
Post Statuses	/wp/v2/statuses
Settings	/wp/v2/settings

4. GUIAS ACCESIBILIDAD

- UAAG: User Agent Accessibility Guidelines (for developers of Web browser,)
- WCAG: Web Content Accessibility Guidelines (site designers)
- ATAG : Authoring Tool Accessibility Guidelines (HTML editors)
- WAI-ARIA : Accessible Rich Internet Applications.

https://w3c.github.io/silver/guidelines/#relationship-to-other-w3c-guidelines WCAG3

https://olgacarreras.blogspot.com/2021/01/wcag-30-novedades-del-ultimo-borrador.html

https://www.usableyaccesible.com/recurso_misvalidadores.php

https://www.tawdis.net/ Validador URL de accesibilidad WCA2

NORMATIVAS ACCESIBILIDAD EUROPEA

Norma UNE 139803:2012 (PDF) , es equivalente a las WCAG 2.0

LEGISLACIÓN ESPAÑOLA

- Ley 34/2002, de 11 de julio, de Servicios de la Sociedad de la Información y de Comercio Electrónico. LSSI (B.O.E. de 12-7-02). : Se fijaba por primera vez la obligación de que las páginas web de la Administración Pública española fueran accesibles Ley 9/2017, de 8 de noviembre, de Contratos del Sector Público, por la que se transponen al ordenamiento jurídico español las Directivas del Parlamento Europeo y del Consejo 2014/23/UE y 2014/24/UE, de 26 de febrero de 2014. se incluyen requisitos de *accesibilidad universal y diseño universal o diseño para todas las personas*.

- La publicación de la Directiva (UE) 2016/2102 del Parlamento Europeo y del Consejo, de 26 de octubre de 2016, sobre la accesibilidad de los sitios web y aplicaciones para dispositivos móviles de los organismos del sector público, que deberá ser transpuesta a la legislación española antes de septiembre de 2018, y que establece los requisitos de accesibilidad de los sitios web y

apps del sector público. Esta norma es el estándar europeo que especifica los requisitos funcionales de accesibilidad de los productos y servicios TIC: los requisitos de accesibilidad de los sitios web (equivalentes al nivel **AA de las WCAG 2.0**), del hardware, del software, de los documentos, etc.

5. SEGURIDAD

La seguridad supone un coste económico y de eficiencia.

- El riesgo cero no es práctico
- Hay diversas formas de mitigar el riesgo
- No se puede gastar un millón para proteger un céntimo

TIPOS DE SEGURIDAD:

1.Seguridad en el Cliente:

- Aparece incrustado en un documento HTML. Un cliente de correo o un navegador que cargue el documento lo ejecutará en la máquina cliente.

2.Seguridad en el Servidor

- Revisar periódicamente los ficheros de log (access_log y error_log en Apache) para detectar posibles ataques.

3.Seguridad en la Comunicación:

- conectar con un sitio web protegido con SSL

4.Seguridad en la Aplicación:

- Almacenar los datos sensibles de forma encriptada

SEGURIDAD: TOP 10

HTTPS://WWW.OWASP.ORG

THE OPEN WEB APPLICATION SECURITY PROJECT (OWASP).

- Informe2017 https://www.owasp.org/images/b/b0/OWASP_Top_10_2017_RC2_Fin
- Top 10
- A1-Injection
- A2-Broken Authentication and Session Management
- A3-Cross-Site Scripting (XSS)
- A4-Broken Access Control
- A5-Security Misconfiguration
- A6-Sensitive Data Exposure
- A7-Insufficient Attack Protection
- A8-Cross-Site Request Forgery (CSRF)
- A9-Using Components with Known Vulnerabilities
- A10-Underprotected APIs

ENLACES DE INTERÉS

- http://www.w3.org/TR/XMLHttpRequest/
- https://developer.mozilla.org/es/docs/XMLHttpRequest/Using_XMLH
- https://fetch.spec.whatwg.org/
- https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Utilizando
- http://www.etnassoft.com/2016/10/10/estudiando-la-nueva-api-fetc
  de-xhr-en-el-nuevo-javascript/
- https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Fu

EI1036_42: Teoría T5