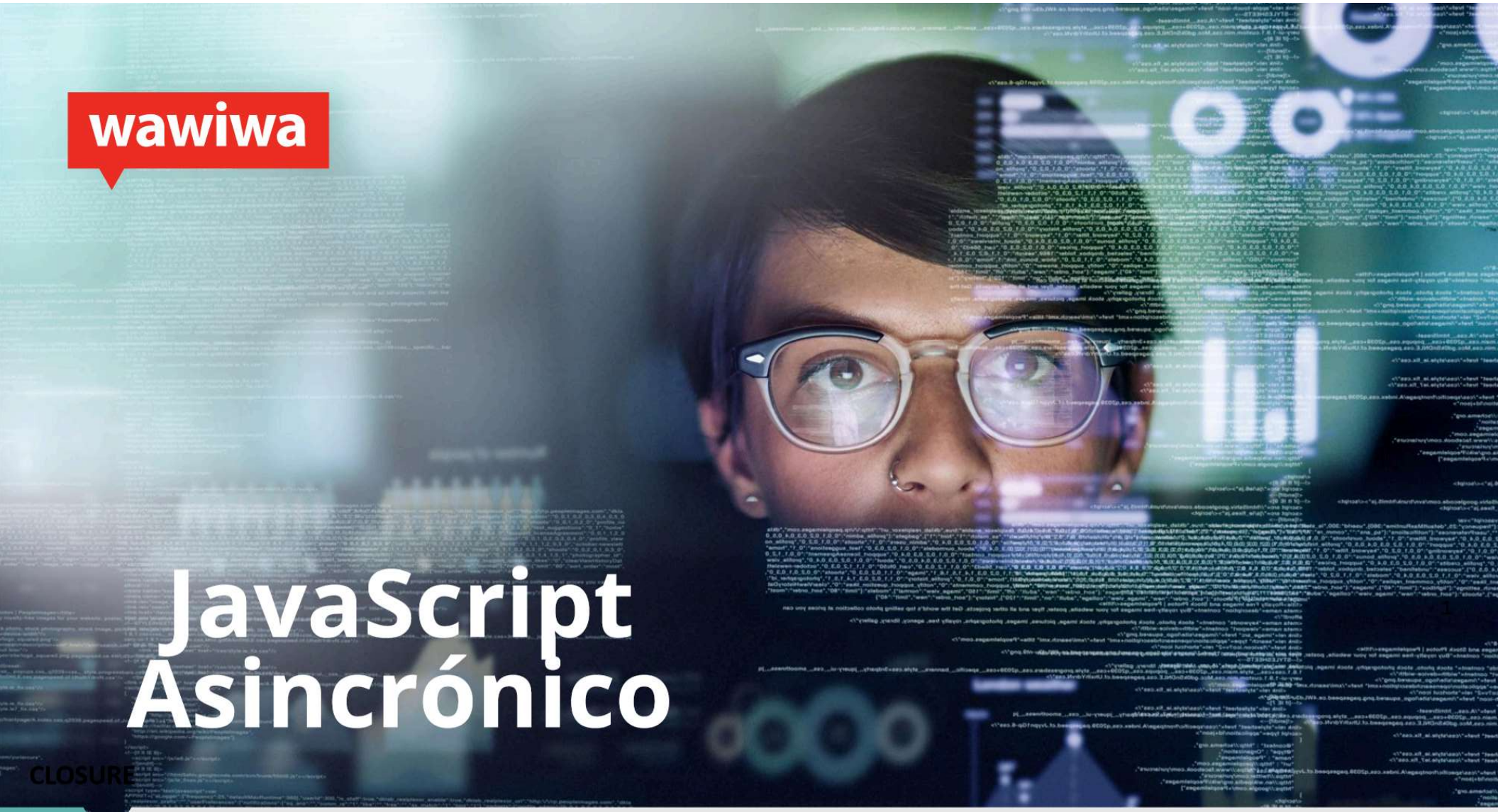


wawiwa

Javascript asincrónico

El objetivo de JavaScript asincrónico es manejar tareas de larga duración que se ejecutan en segundo plano.

Ejemplo : obtener datos del servidor - llamada Ajax

Sincrónico es un código que se ejecuta línea por línea en orden.

```
let x = 6;

let y = 4;

let sum = x+y;

console.log(sum);
```

wawiwa

Javascript asincrónico

```
<p id="details"></p>
<button onclick="details()">click</button>

function details() {
    const p = document.getElementById("details")

    // asincrónico
    // El código principal no es bloqueado
    setTimeout(() => p.innerText="Asaf", 3000);
    p.style.background="blue";
}
```

wawiwa

Javascript asincrónico

```
<img src="" class="cat">
<button onclick="details()">click</button>

function details() {
    const img = document.querySelector(".cat");

    // img.src es asincrónico
    img.src = "cat.png";
    img.addEventListener('load', () => {
        img.classList.add("someclass");
    });
}
```

Con ajax podemos:

1.) Leer datos del servidor
2.) Actualizar una página sin recargarla
3.) Enviar datos al servidor en segundo plano

wawiwa

API

API : application programming Interface (Interfaz de Programación de Aplicaciones): Pieza de software que puede ser usada por otra pieza de software, para permitir a las aplicaciones hablar entre ellas.

web api : Es una aplicación que se ejecuta en un servidor, que recibe solicitudes de datos y envía datos como respuesta.

Existen muchos api's - datos de clima, vuelos, divisas, google play, google maps y mas.

Formato de los datos de la API

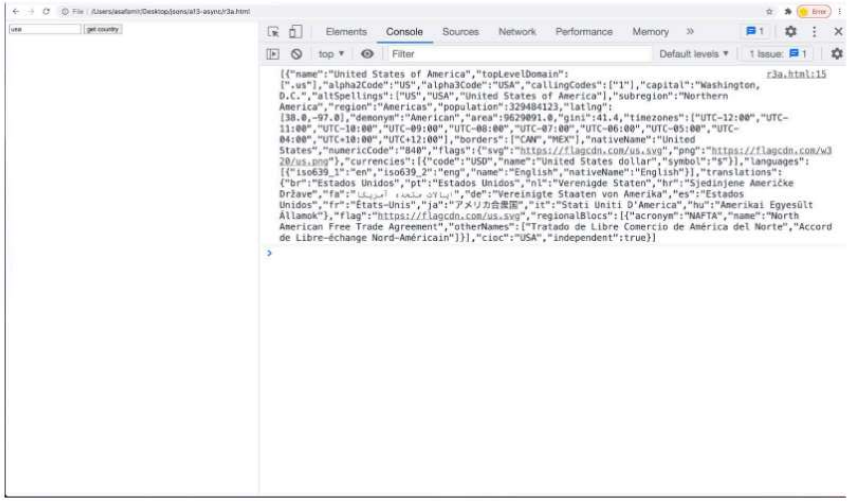
Ajax - la x en ajax significa xml pero hoy en día ya nadie usa datos xml.

La mayoría de las APIs hoy en día usan json api. Json es el formato de datos más popular hoy en día, porque es un objeto convertido en string.

Por eso es tan facil mandar a través de la web y usar Javscript una vez que los datos llegan.

wawiwa

Ejemplo de Ajax el método viejo



wawiwa

Ejemplo de Ajax el método viejo

```
<input type="text" id="country-input">
<button onclick="getCountry()">get country</button>
<p id="country"></p>
<p id="region"></p>
<p id="population"></p>
<p id="language"></p>
<p id="currency"></p>
<img src="" id="flag">
```

wawiwa

Ejemplo de Ajax el método viejo

```
const img = document.getElementById("flag");
const country = document.getElementById("country");
const region = document.getElementById("region");
const population = document.getElementById("population");
const currency = document.getElementById("currency");
const language = document.getElementById("language");
const countryInput = document.getElementById("country-input");
```

wawiwa

Ejemplo de Ajax el método viejo

```
// el método viejo todavía existe
function getCountry(country) {
    const req = new XMLHttpRequest();

    // lo que pasa en segundo plano
    req.open('GET', 'https://restcountries.com/v2/name/' + countryInput.value);
    req.send();
    req.addEventListener('load', function() {
        const [data] = JSON.parse(this.responseText);
        console.log(data);

        img.src = data.flag;
        country.innerText = "country: " + data.name;
        region.innerText = "region: " + data.region;
        population.innerText = "population: " + data.population;
        currency.innerText = "currency: " + data.currencies[0].name;
        language.innerText = "language: " + data.languages[0].name;
    });
}
```

wawiwa

Promise (promesa)

Que es una promesa?

Una promesa es un objeto Javascript que vincula el código que produce datos y el código que los consume.

Puede tomar tiempo y debe esperar al resultado.

El objeto Promise representa la finalización (o fallo) del evento a una operación asincrónica y el valor del resultado.

¿Por qué y cuando los usamos?

Las Promises son la elección ideal para manejar operaciones asíncronas de la manera más simple. Pueden manejar operaciones asíncronas múltiples fácilmente y proveer un mejor manejo que solamente callbacks y eventos.

11

Imagina una función, createAudioFileAsync(), que genera de forma asincrónica un archivo de sonido dado un registro de configuración y dos funciones de devolución de llamada, una llamada si el archivo de audio se crea con éxito y la otra llamada si ocurre un error.

wawiwa

Ejemplo de Promise

```
function successCallback(result) {
    console.log("Audio file ready at URL: " + result);
}

function failureCallback(error) {
    console.error("Error generating audio file: " + error);
}

function audioSettings () { /* ... */ }

createAudioFileAsync(audioSettings, successCallback, failureCallback);
```

or

```
createAudioFileAsync(audioSettings).then(successCallback, failureCallback);
```


12

wawiwa

Fetch Api

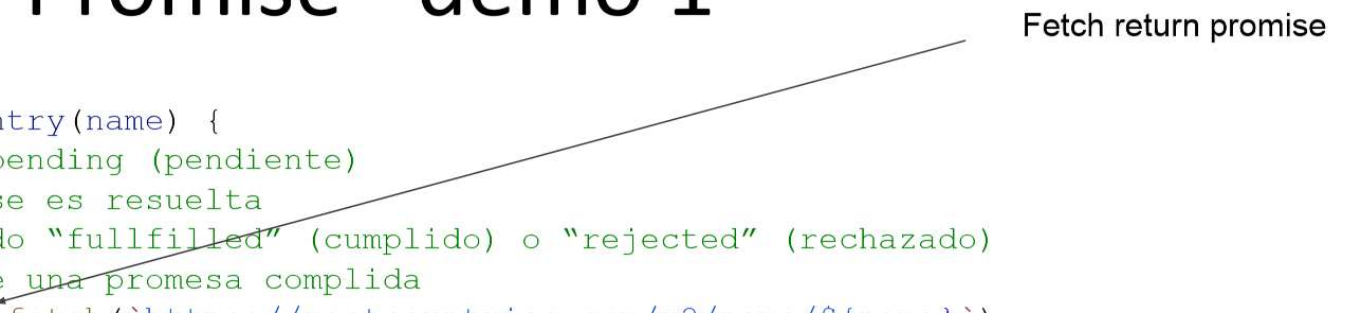
La Fetch API nos provee una interfaz para obtener recursos (incluyendo a través de la red). Va a parecer conocido a todos los que han usado XMLHttpRequest, pero la nueva API proporciona una configuración de características más poderosas y flexibles.

El método fetch() toma un argumento obligatorio que es la ruta del recurso que quieres obtener. Retorna una **Promise** que que resuelve el Response ("respuesta") al request ("solicitud")

El objeto Promise representa la finalización (o fracas) del evento de una operación asíncrona y el valor de su resultado.

wawiwa

Ejemplo de Promise - demo 1




```
function getCountry(name) {
    // fase 1: pending (pendiente)
    // la promise es resuelta
    // con estado "fullfilled" (cumplido) o "rejected" (rechazado)
    // manejo de una promesa complida
    const req = fetch(`https://restcountries.com/v2/name/${name}`)
        .then(response => {
            console.log(response);
            // los datos están en el body
            // se necesita llamar al json
            return response.json(); // disponible en cualquier valor de resultado.
            // La función response.json() es asíncrona. Entonces retorna una promesa
        })
        .then(data => console.log(data));
}

getCountry('usa');
```

wawiwa

Ejemplo de Promise - muestra demo 2

wawiwa

Promesa de función de flecha - explicación en demo 3

```
const div = document.querySelector(".flags");
div.style.opacity = 1;

function renderCountry(data) {
    const html =
        `<div>
            <img width="200" height="100" src=" ${data.flag}" />
            <p id="country">${data.flag}</p>
            <p id="region">${data.region}</p>
            <p id="population">${data.population}</p>
            <p id="language">${data.languages[0].name}</p>
            <p id="currency">${data.currencies[0].name}</p>
        </div>`

    div.insertAdjacentHTML('beforeend', html);
}

function getCountry(name) {
    fetch(`https://restcountries.com/v2/name/ ${name}`)
        .then(response => response.json())
        .then(data => renderCountry(data[0]));
}

getCountry('usa');
```

wawiwa

Encadenando promesas Chaining promises

Ejemplo de Promise - muestra demo 4

wawiwa

Errores en promises

Ejemplo de Promise - muestra demo 5

wawiwa

Errores en promises

Ejemplo de Promise - muestra demo 6

wawiwa

finally (“finalmente”) en promises

Ejemplo de Promise - muestra demo 7

wawiwa

Más errores

Ejemplo de Promise - muestra demo 8

wawiwa

Crear Promises

```
const promise = new Promise((resolve, reject) => {
    // do thing (possibly asy) - smile, cry, .....bla
    if ( /* every thing is ok - she marry u*/true ) {
        resolve("Every thing work");
    }
    else if (/* she doesnt want u*/false) {
        reject(Error("she broke my heart"));
    }
});

promise.then(result => console.log(result),
    err => console.log(err));
```

wawiwa

Crear un promise

Ejemplo de Promise - muestra demo 9

wawiwa

Crear un “wait” promise

Ejemplo de Promise - muestra demo 10

wawiwa

Promises - Bueno saberlo

```
const promise = new Promise((resolve, reject) => resolve(1));

promise.then(val => {
    console.log(val);
    return val + 2;
})
.then(val => {
    console.log(val);
    return val + 5;
})
.then(val => console.log(val)); // ????
```

wawiwa

Promises - bueno saberlo

```
function get(url) {
      // Retorna una nueva promise.
      return new  Promise(function(resolve, reject) {
            // Hace lo usual de XHR
            const req = new XMLHttpRequest();

            req.open('GET', url, true);

            req.onload = () => {
                  // Esto es llamado incluso en errores como 404 etc
                  // entonces se checa el estado
                  if (req.DONE && req.status == 200) {
                        // Resuelve el promise con el texto del resultado
                        resolve(req.response);
                  }
                  else {
                        // De otra manera rechaza con un texto de estado/
                        // que esperamos que sea un error significativo
                        reject(Error(req.statusText));
                  }
            };

            // Maneja errores de la red
            req.onerror = () => reject(Error("Network Error"));

            // Hace la solicitud
            req.send();
      });
}

const url = 'http://ilemon.mobi/site1/b.php' ;
get(url).then(res => console.log(res));
```

wawiwa

Promises - bueno saberlo

```
function get(url) {
      // Retorna una nueva promise.
      return new  Promise(function(resolve, reject) {
            // Hace lo usual de XHR
            const req = new XMLHttpRequest();

            req.open('GET', url, true);

            req.onload = () => {
                  // Esto es llamado incluso en errores como 404 etc
                  // entonces se checa el estado
                  if (req.DONE && req.status == 200) {
                        // Resuelve el promise con el texto del resultado
                        resolve(req.response);
                  }
                  else {
                        // De otra manera rechaza con un texto de estado/
                        // que esperamos que sea un error significativo
                        reject(Error(req.statusText));
                  }
            };

            // Maneja errores de la red
            req.onerror = () => reject(Error("Network Error"));

            // Hace la solicitud
            req.send();
      });
}

const url = 'http://ilemon.mobi/site1/b.php' ;
get(url).then(res => console.log(res));
```

wawiwa

Promises - bueno saberlo

```
function get(url) {
    return new  Promise(function(resolve, reject) {
        const req = new XMLHttpRequest();

        req.open('GET', url, true);

        req.onload = function() {
            req.status == 200 ?
                resolve(req.response) :
                reject(Error(req.statusText));
        };

        req.onerror = () => reject(Error("Network Error"));

        // Hace la solicitud
        req.send();
    });
}
```

wawiwa

Promises - bueno saberlo

```
function getJson(url) {
   return get(url).then(JSON.parse);
}

const url = 'http://ilemon.mobi/site1/b.php';

// get json tambien retorna una promise
getJson(url).then(res => console.log(res));
```

wawiwa

Promises - bueno saberlo

```
function get(url) {
    return new  Promise((resolve, reject) => {
        // Hace lo usual de XHR
        const req = new XMLHttpRequest();

        req.open('GET', url, true);

        req.onload = () => {
            if (req.status == 200) {
                // reslueve la promise con el resultado
            }
            else {
                reject(Error(req.statusText));
            }
        };

        req.onerror = () => reject(Error("Network Error"));

        // Hace la solicitud
        req.send();
    });
}
```

```
function getJson(url) {
    return get(url).then(JSON.parse);
}

const url =
    'http://api.open-notify.org/astros.json';

getJson(url).then(res =>
    console.log(res));
```


wawiwa
¿Preguntas?