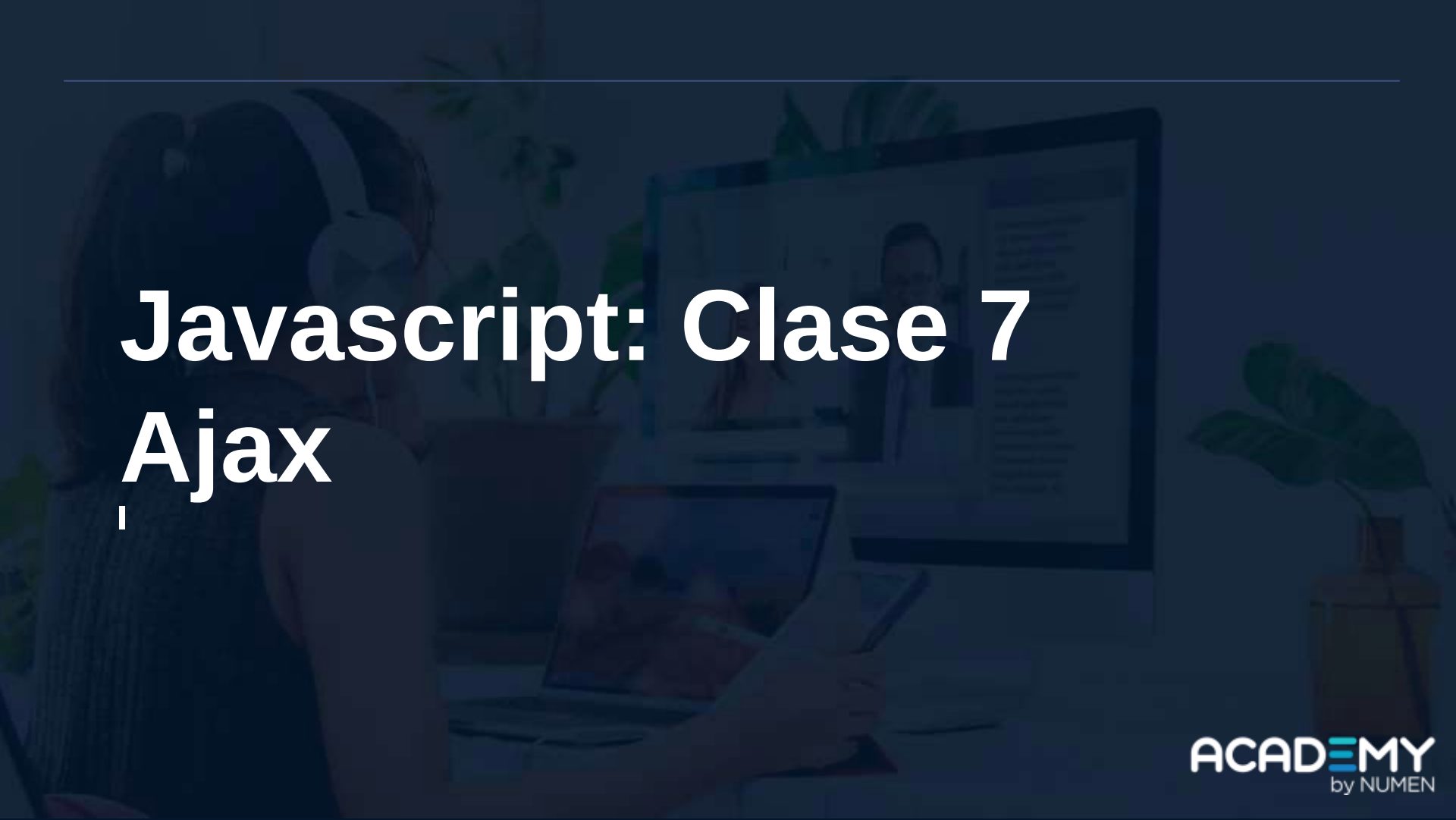
ACADEMY by NUMEN

FULL STACK

Comenzamos en unos minutos


ACADEMY
by NUMEN

Javascript: Clase 7
Ajax

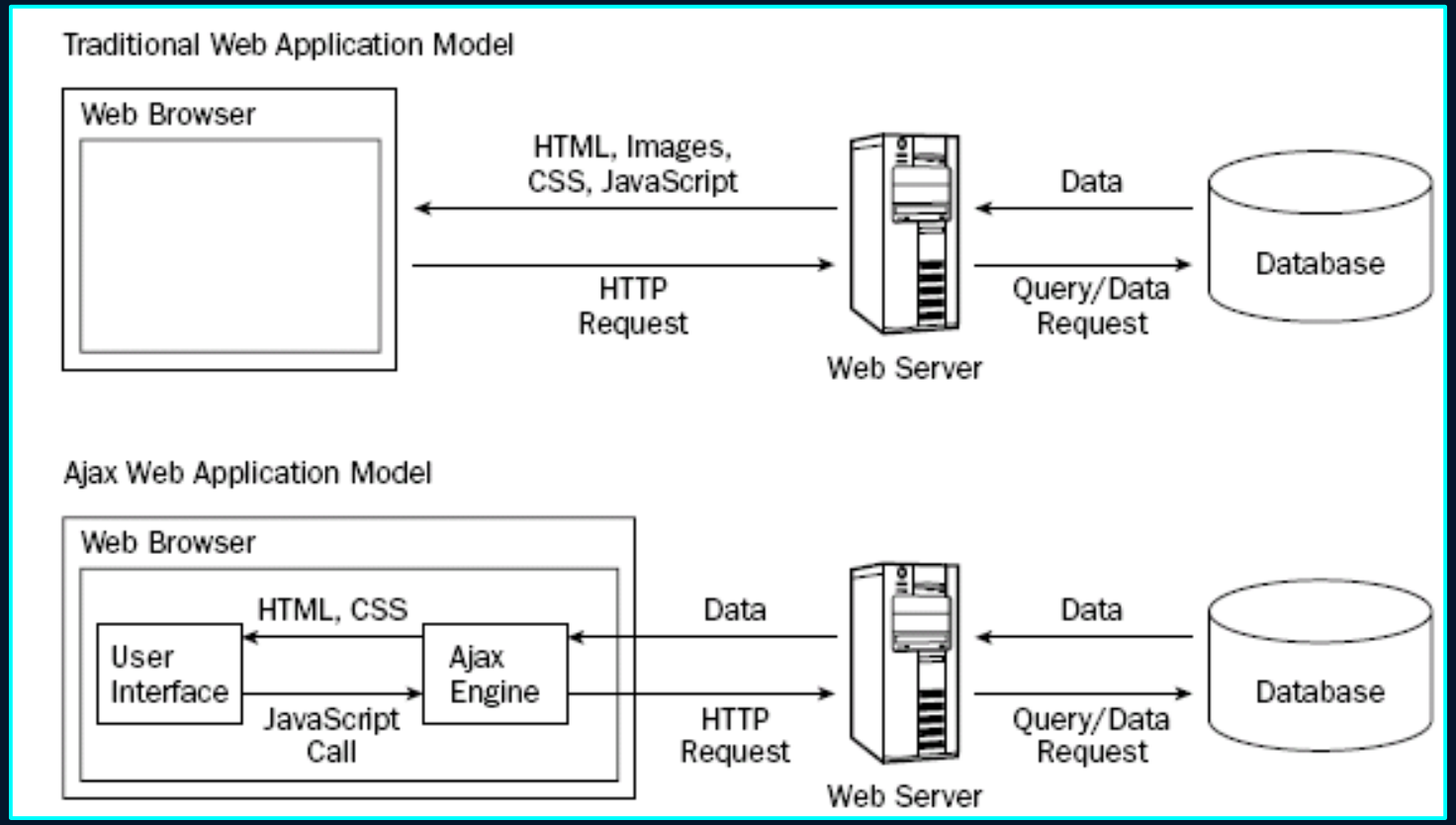


ACADEMY by NUMEN

¿Qué es Ajax y para qué sirve?

- Es un conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asíncrona, procesando cualquier solicitud al servidor en segundo plano.
- Gestiona el contenido dinámico de un sitio web y permite la interacción dinámica del usuario.
- No es necesario recargar y redibujar la página web completa, con lo que todo es más rápido.
- Los pasos que antes podía ser necesario dar cargando varias páginas web pueden quedar condensados en una sola página que va cambiando gracias a Ajax y a la información recibida del servidor.

Modelo de aplicación Web



JSON


```json
{
    "localizaciones": [
        {
            "latitude": 40.416875,
            "longitude": -3.703308,
            "city": "Madrid",
            "description": "Puerta del Sol"
        },
        {
            "latitude": 40.417438,
            "longitude": -3.693363,
            "city": "Madrid",
            "description": "Paseo del Prado"
        },
        {
            "latitude": 40.407015,
            "longitude": -3.691163,
            "city": "Madrid",
            "description": "Estación de Atocha"
        }
    ]
}
```

Códigos de estado de respuesta HTTP

1. Respuestas informativas (`100` – `199`),
2. Respuestas satisfactorias (`200` – `299`),
3. Redirecciones (`300` – `399`),
4. Errores de los clientes (`400` – `499`),
5. y errores de los servidores (`500` – `599`).

Más información en https://developer.mozilla.org/es/docs/Web/HTTP/Status

Estados de petición

Valor	Descripción
`0`	No inicializado (objeto creado, pero no se ha invocado el método `open`)
`1`	Cargando (objeto creado, pero no se ha invocado el método `send`)
`2`	Cargado (se ha invocado el método `send`, pero el servidor aún no ha respondido)
`3`	Interactivo (se han recibido algunos datos, aunque no se puede emplear la propiedad `responseText`)
`4`	Completo (se han recibido todos los datos de la respuesta del servidor)

Objeto XMLHTTPRequest

XMLHttpRequest es un **objeto que p**roporciona una forma fácil de obtener información de una URL sin tener que recargar la página completa. A través del console.log veremos que propiedades y métodos cuelgan de este objeto.

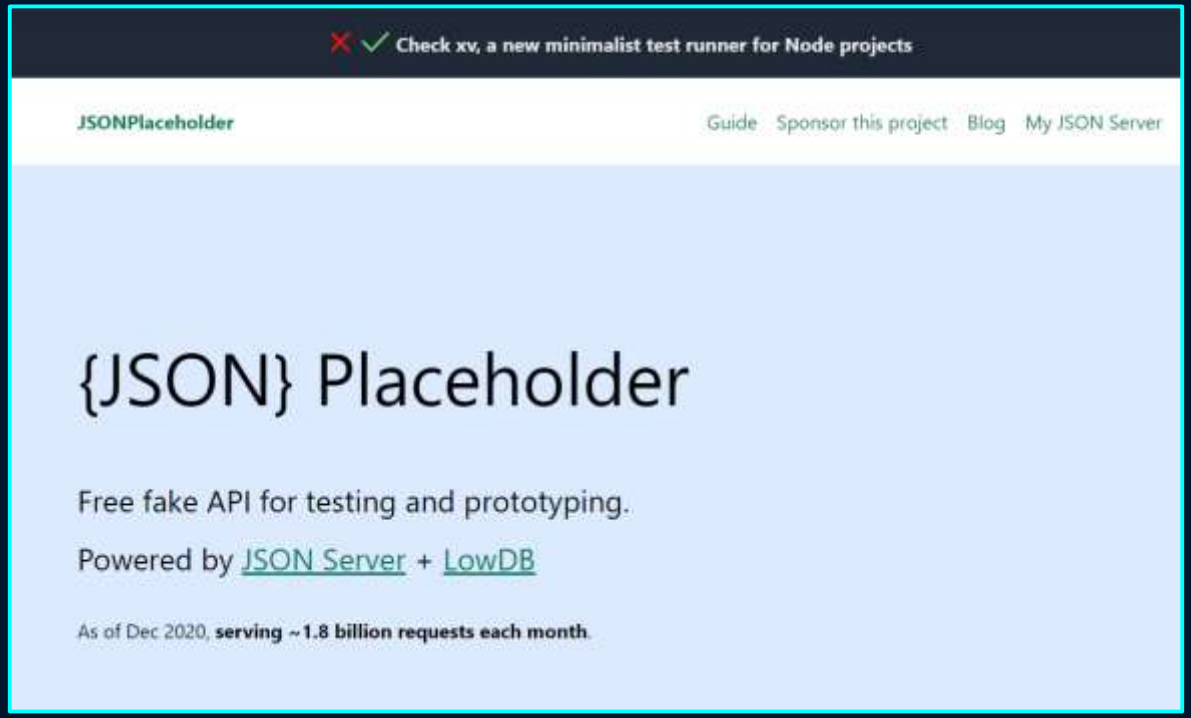
```
<!-- index.html -->
<ul id="xhr">

</ul>
```

```
// index.js
const xhr = new XMLHttpRequest(),
    $xhr = document.getElementById("xhr"),
    $fragment = document.createDocumentFragment();

console.log(xhr)
```

JSON Placeholder



Enlace:
https://jsonplaceholder.typicode.com/

Pasos para consumir datos

```
// index.js
// Paso 1 - crear una instancia del objeto XMLHTTPRequest
const xhr = new XMLHttpRequest(),
    $xhr = document.getElementById("xhr"),
    $fragment = document.createDocumentFragment();

// Paso 2 - Asignar el o los eventos
xhr.addEventListener("readystatechange", function (e) {});

// Paso 3 - Abrir la petición
xhr.open("GET", "https://jsonplaceholder.typicode.com/users")

// Paso 4 - Enviar la petición
xhr.send();
```

Propiedades para el objeto XMLHTTPRequest

Propiedad	Descripción
`readyState`	Valor numérico (entero) que almacena el estado de la petición
`responseText`	El contenido de la respuesta del servidor en forma de cadena de texto
`responseXML`	El contenido de la respuesta del servidor en formato XML. El objeto devuelto se puede procesar como un objeto DOM
`status`	El código de estado HTTP devuelto por el servidor (`200` para una respuesta correcta, `404` para "No encontrado", `500` para un error de servidor, etc.)
`statusText`	El código de estado HTTP devuelto por el servidor en forma de cadena de texto: "OK", "Not Found", "Internal Server Error", etc.

Programando la función del evento: Validaciones

Lo primero que tenemos que hacer es validar que, por un lado solo nos devuelva la información cuando el estado de petición esté listo (readyState), y por el otro lado validar que el código de respuesta de la petición sea exitoso (200-299).

```
// index.js
xhr.addEventListener("readystatechange", function (e) {
    // Validando que sólo devuelva datos si el estado es ready
    if(xhr.readyState !== 4) return;

    // Validando que el código de respuesta de la petición sea exitoso
    if(xhr.status >= 200 && xhr.status < 300) {
        console.log('exito')
    } else {
        console.log('error')
    }
});
```

Programando la función del evento: Caso exitoso

Ahora programaremos lo que sucederá en caso de que la petición sea exitosa.

```
// index.js
xhr.addEventListener("readystatechange", function (e) {
    // if(xhr.readyState !== 4) return;

    if(xhr.status >= 200 && xhr.status < 300) {
        let json = JSON.parse(xhr.responseText)

        json.forEach((el) => {
            const $li = document.createElement('li');
            $li.innerHTML = `${el.name} -- ${el.email} -- ${el.phone}`;
            $fragment.appendChild($li);
        });
      $xhr.appendChild($fragment)
    // } else {
    //     console.log('error')
    // }
});
```

Programando la función del evento: Caso erroneo

```
// index.js
xhr.addEventListener("readystatechange", function (e) {
    // if(xhr.readyState !== 4) return;

    if(xhr.status >= 200 && xhr.status < 300) {
        // let json = JSON.parse(xhr.responseText)

        // json.forEach((el) => {
        //     const $li = document.createElement('li');
        //     $li.innerHTML = `${el.name} -- ${el.email} -- ${el.phone}`;
        //     $fragment.appendChild($li);
        // });
    } else {
        let message = xhr.statusText || "Ocurrio un error";
        $xhr.innerHTML = `Error ${xhr.status}: ${message}`
    }
});
```

ACADEMY
by NUMEN