

JAVASCRIPT

JS

Contenido

★ JSON

- `JSON.parse()`
- `JSON.stringify()`

★ AJAX

- `xmlHttpRequest`
- `onreadystatechange`
- `open()`
- `send()`

JSON

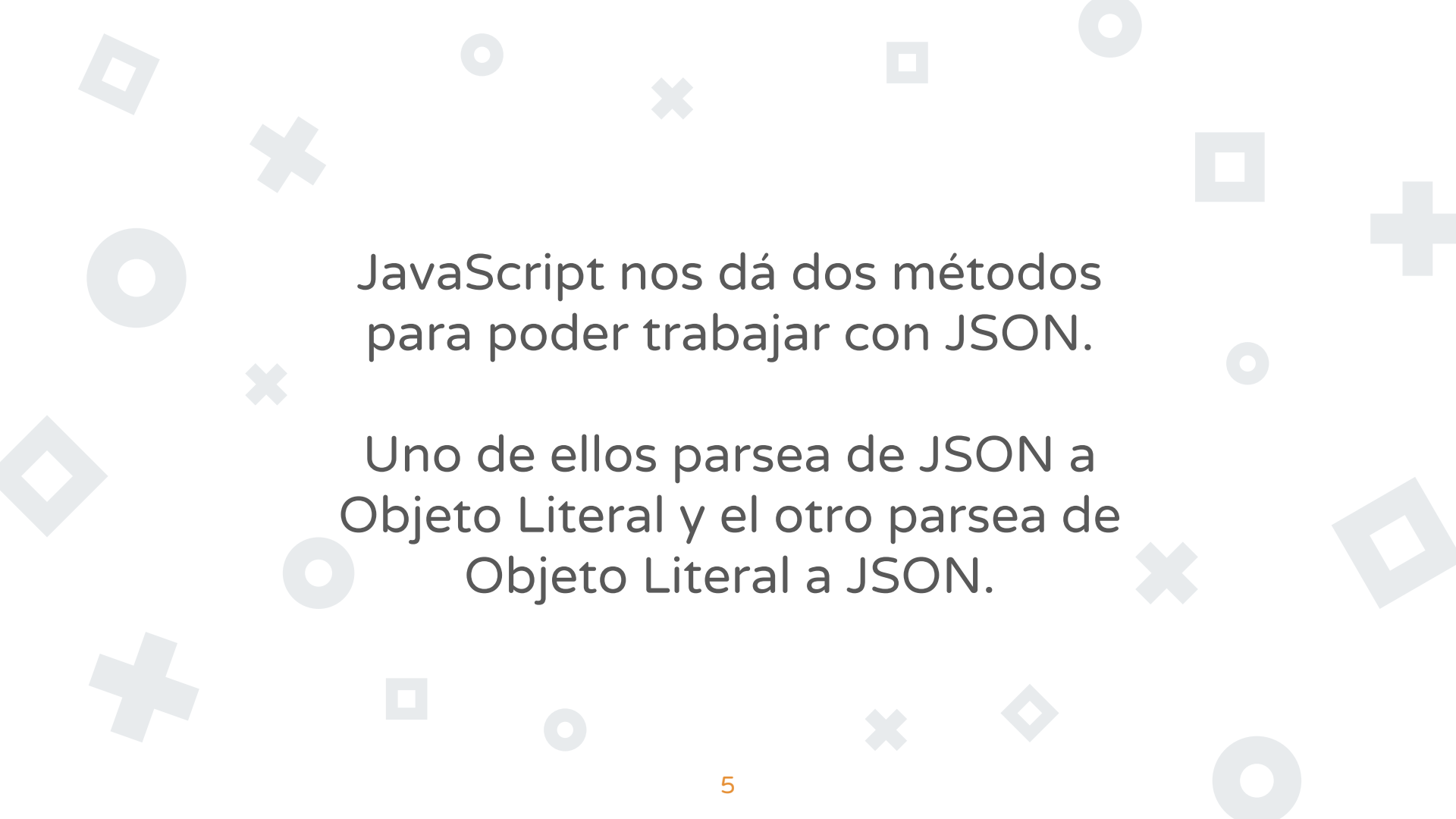
(JavaScript Object Notation)

Es un formato de intercambio de datos que deriva de la notación de objetos literales de JS.

```
var ejemploJSON = '{"prop1": "String", "prop2": 1, "prop3": [],  
"prop4": true}';
```

```
// -----
```

```
var adaLovelaceEnJSON = '{"nombre": "Ada", "apellido": "Lovelace",  
"profesion": "Programadora"}';
```

The background is white and decorated with various light gray geometric shapes scattered across the page. These shapes include squares (some with smaller squares inside), circles (some with smaller circles inside), and plus signs. The shapes are of different sizes and are positioned at various angles, creating a subtle pattern.

JavaScript nos dá dos métodos
para poder trabajar con JSON.

Uno de ellos parsea de JSON a
Objeto Literal y el otro parsea de
Objeto Literal a JSON.

Formato JSON

JSON.stringify()

El método stringify permite pasar un objeto literal de JavaScript al formato JSON.

```
var miAutito = {marca: "Fiat", modelo: 1985, color: "Verde"};
```

```
JSON.stringify(miAutito);
```

```
// '{"marca": "Fiat", "modelo": 1985, "color": "Verde"}'
```

Formato JSON

JSON.parse()

El método parse toma un string en formato JSON y lo transforma en un objeto literal.

```
var toObject = JSON.parse('{ "marca": "Fiat", "modelo": 1985,  
"color": "Verde" }')
```

```
// {marca: "Fiat", modelo: 1985, color: "Verde"}
```

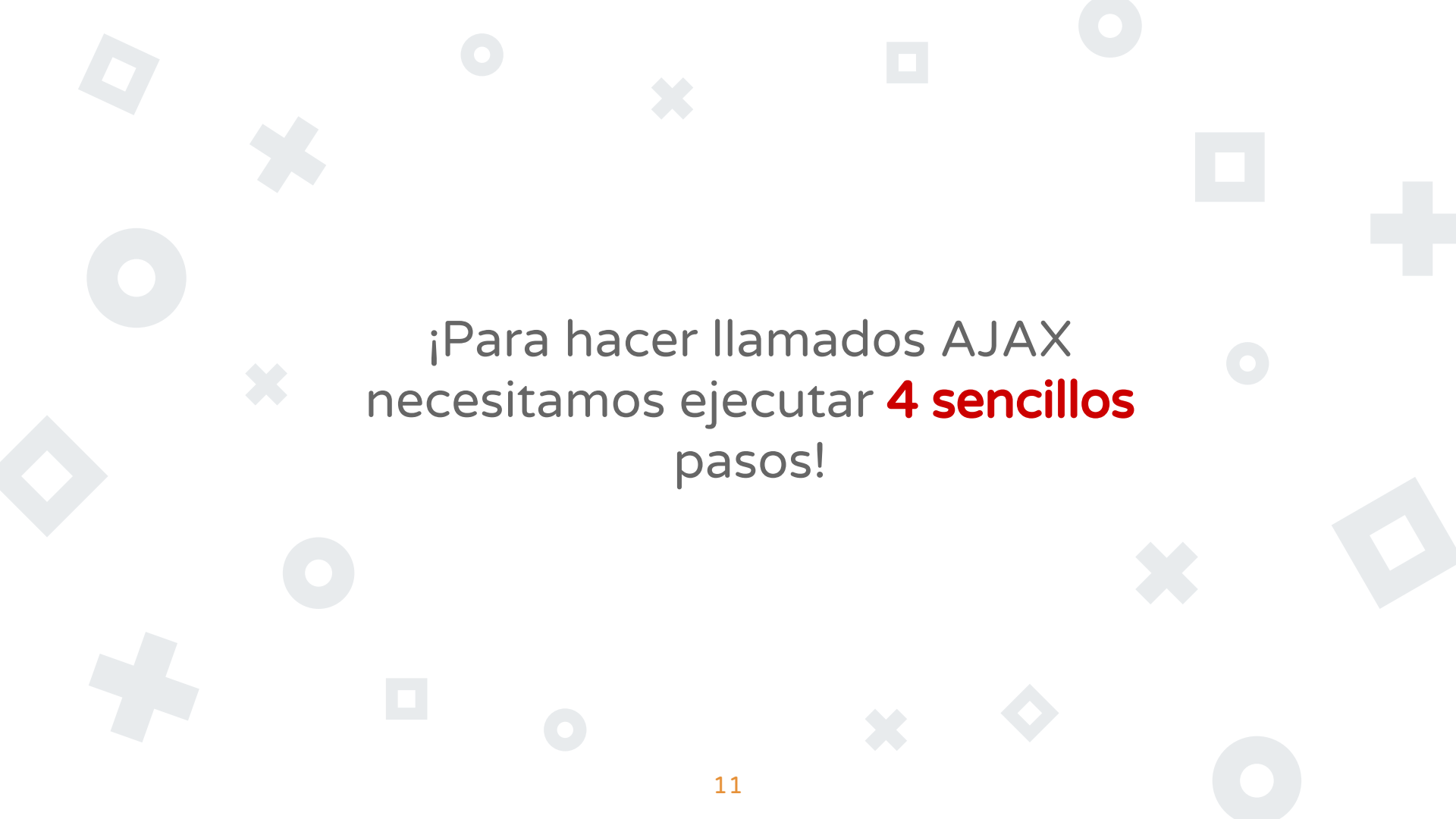


AJAX

Llamados Asincrónicos

Asynchronous JavaScript and XML

Nos permite hacer peticiones HTTP (GET y POST) sin tener que salir del documento HTML actual. Es por eso que podemos decir que AJAX es una técnica para crear webs dinámicas.



¡Para hacer llamados AJAX
necesitamos ejecutar **4 sencillos**
pasos!

AJAX

1. Instanciar el objeto XMLHttpRequest

Al instanciar el objeto, y guarda el mismo en una variable, ahora sobre dicha variable vamos a poder ejecutar sobre ésta los métodos a los que el objeto nos da acceso.

```
var ajaxCall = new XMLHttpRequest();
```

```
// ajaxCall, ahora guarda la instancia del objeto XMLHttpRequest
```

```
// A dicha variable le vamos a solicitar varios métodos
```

AJAX

2. onreadystatechange

Este evento recibe como valor una función. Dentro de la función tendremos que verificar si el ESTADO y el STATUS está OK.

```
ajaxCall.onreadystatechange = function() {  
    if (ajaxCall.readyState === 4 && ajaxCall.status === 200) {  
        console.log(ajaxCall.response);  
    }  
};
```

`ajaxCall.readyState`

0 = no se inicializó

1 = está cargando

2 = ya se envió el pedido

3 = está descargando la respuesta

4 = terminó.

`ajaxCall.status`

El estado de la respuesta al pedido. Por ejemplo:

`status = 200 // pedido exitoso`

`ajaxCall.response`

La respuesta al pedido como texto, o null si el pedido no fue exitoso o todavía no se envió. Solo lectura.

AJAX

3. open()

open() inicializa el pedido asíncrono.

```
ajaxCall.open(method, url, async);
```

```
// method = String - GET o POST
```

```
// url = String - Url a donde se hace / envía el pedido
```

```
// async = Boolean - true pasa asíncrono ó false para síncrono. Default true
```

AJAX

3. send()

send() envía el pedido. Si va por GET, **NO** recibe parámetros.

```
ajaxCall.send();
```




Recapitulando:

¡En conjunto, los **4 pasos** se ven así!

```
var ajaxCall = new XMLHttpRequest();

ajaxCall.onreadystatechange = function() {
    if (ajaxCall.readyState === 4 && ajaxCall.status === 200) {
        console.log(ajaxCall.response);
        // do stuff with var ajaxCall.response
    }
};

ajaxCall.open(
    "GET",
    "https://jsonplaceholder.typicode.com/users",
    true
);

ajaxCall.send();
```



Y:

¿Qué pasaría si deseamos
enviar data con vía POST?

```
var ajaxCall = new XMLHttpRequest();

ajaxCall.onreadystatechange = function() {
    if (ajaxCall.readyState === 4 && ajaxCall.status === 200) {
        console.log(ajaxCall.response);
        // do stuff if ajaxCall.response is OK
    }
};

ajaxCall.open("POST", "http://localhost/post.php", true);

ajaxCall.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

var data = "nombre=Ada&&apellido=Lovelace&&curso=FullStack";

ajaxCall.send(data);
```

ajaxCall.setRequestHeader

Añade las cabeceras HTTP necesarias. Siempre es necesario para pasar datos por POST.

data

Es una variable cualquiera que pasará a \$_POST de PHP un array asociativo (clave / valor). Esta variable se debe definir tipo String así:

"clave=valor&&otraClave=otroValor&&otraclaveMas=OtroValorMás"



¡A practicar!

Práctica Integradora



```
// To Do  
console.log("Practice Time");
```