

# JAVASCRIPT



**JS**

# Contenido

- ★ ¿Qué es y para qué usamos JS?
- ★ Repaso
- ★ Objeto Literal en JS
- ★ D.O.M
- ★ Método style

# ¿Qué es y para qué usamos JS?



¡Nace una leyenda!

En 1995 Brendan Eich lo creó en  
tan solo 10 días para Netscape.



# ¿Qué es y para qué?

## ¿Qué es?

- Es un lenguaje de programación basado en objetos.
- Imperativo e interpretado.
- No tipado.
- Dialecto de ECMAScript (ES).

# ¿Qué es y para qué?

## ¿Para Qué?

- Permite crear, leer, actualizar y eliminar el contenido de los documentos HTML.
- Aporta dinamismo a la UI para hacer más agradable la UX.
- Permite validar datos y verificar los mismos antes de enviarlos al servidor.
- Por qué es el lenguaje que más adeptos ha adquirido en los últimos 3 años.

Fuente (<https://insights.stackoverflow.com/survey/2018/#technology>)



# REPASO

Conceptos básicos de programación usados  
también en Javascript

# Repaso - Tipo de datos

- ★ String
- ★ Array
- ★ Number
- ★ Boolean

- ★ Object
- ★ Functions
- ★ Null
- ★ Undefined



# Repaso - truthy & falsy

truthy	falsy
true	false
[]	null
1	0
// cualquier otro dato	undefined
	NaN // Not a Number
	"" // string vacío

# Repaso - Operadores

- `=` // asignación
- `+` // suma
- `-` // resta
- `*` // multiplicación
- `/` // división
- `%` // módulo
- `++` // incremento
- `--` // decremento
- `==` // igual simple
- `===` // igual estricto
- `!=` // distinto
- `!==` // distinto estricto
- `>` // mayor
- `<` // menor
- `>=` // mayor o igual
- `<=` // menor o igual
- `&&` // and
- `||` // or
- `!` // negación

# Repaso - if - if / else - if ternario

```
if (true) {  
    // when is true  
}
```

```
if (true) {  
    // when is true  
} else {  
    // when is false  
}
```

# Repaso - if - if / else - if ternario

`(test) ? expressionA : expressionB;`

`(test)` : condición a evaluar

`expressionA` : resultado cuando es verdadero

`expressionB` : resultado cuando es falso

# Repaso - if - if / else - if ternario

```
var isMember = true;
```

```
(isMember) ? "$12.50" : "$17.80";
```

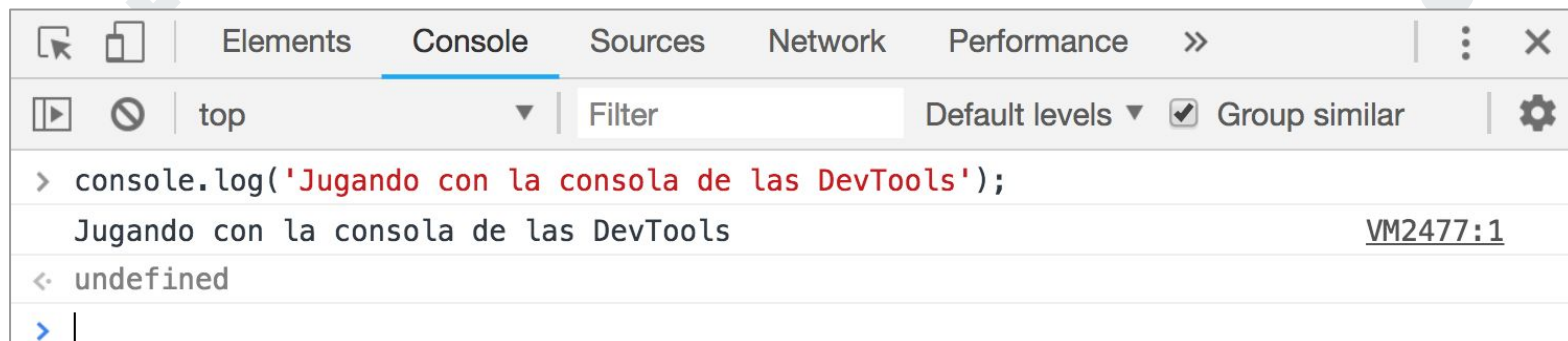
```
// Retorna "$12.50"
```

# ¿cómo debugear?



¡Donde ver los datos!

## La mejor manera es usando las DevTools y el `console.log()`



## Repaso - if - if / else - if ternario

```
var isMember = true;
```

```
var price = (isMember) ? "$12.50" : "$17.80";
```

```
console.log("Precio a cobrar: " + price);
```

```
// Precio a cobrar: 12.50 (en la consola)
```





¿Cómo vinculamos **JS** con **HTML**?

# Vinculación Interna

```
<body>  
  ...  
  <script>  
    var mySuperHeroe = "Ada Lovelace";  
    console.log(mySuperHeroe); // "Ada Lovelace"  
  </script>  
</body>
```

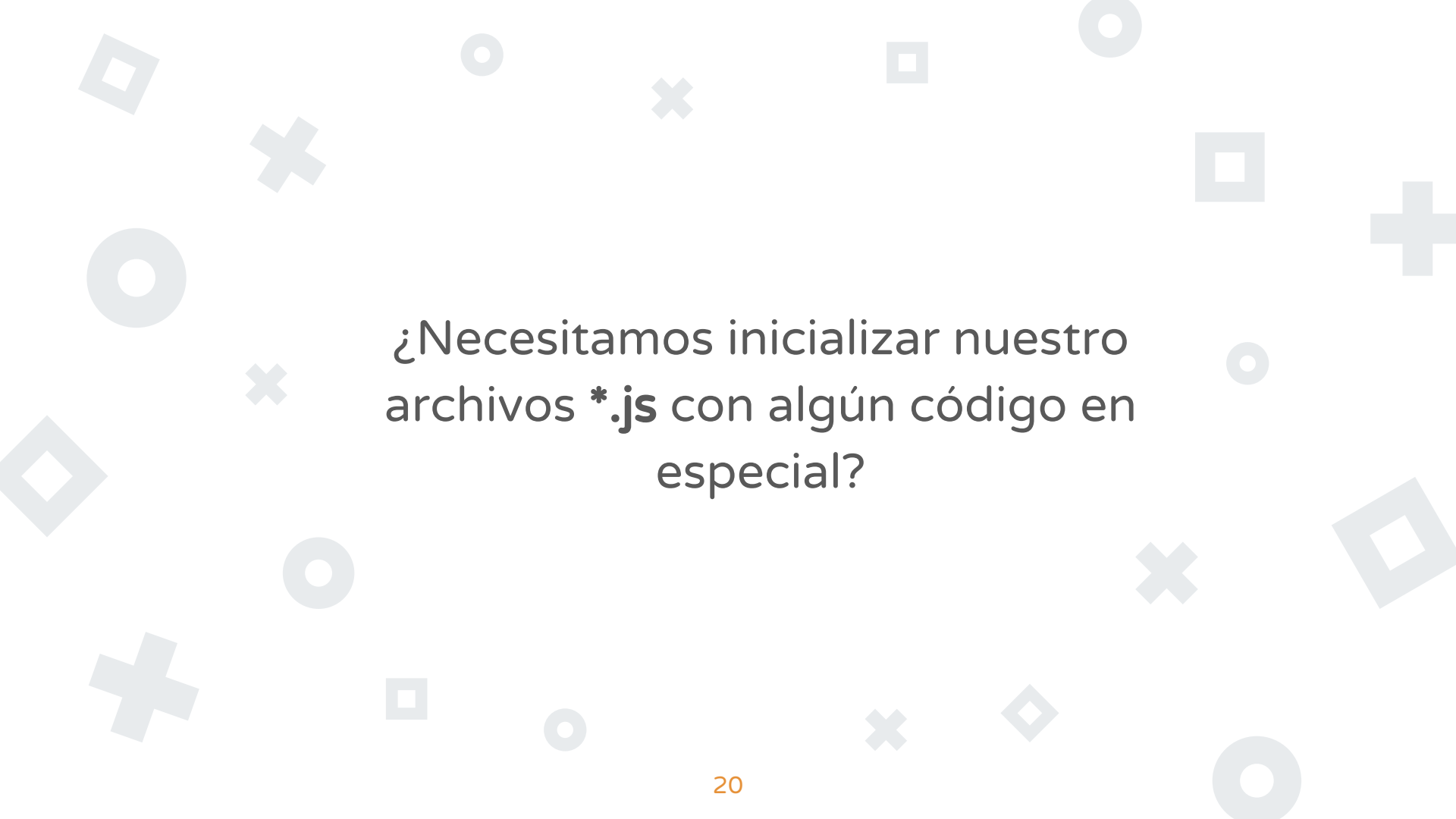
# Vinculación Externa

```
<body>
```

```
...
```

```
<script src="js/main.js"></script>
```

```
</body>
```



¿Necesitamos inicializar nuestro  
archivos **\*.js** con algún código en  
especial?

# Vinculación Externa

```
// main.js
```

```
window.onload = function () {  
    var mySuperHeroe = "Ada Lovelace";  
    console.log(mySuperHeroe); // "Ada Lovelace"  
}
```

// **onload**, es un evento que permite que todo el script se ejecute cuando se haya cargado por completo el objeto **document** dentro del objeto **window**. En unas clases más adelante profundizaremos sobre los eventos.

**¡A practicar!**

Ejercicios Rompehielo  
del 1 al 5



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



# OBJETOS LITERALES

Representación en código de un  
elemento de la vida real

# Objetos literales

## ¿Qué es?

Un objeto de JavaScript es un bloque de código que tiene **propiedades**, las cuales a su vez tienen un valor determinado.

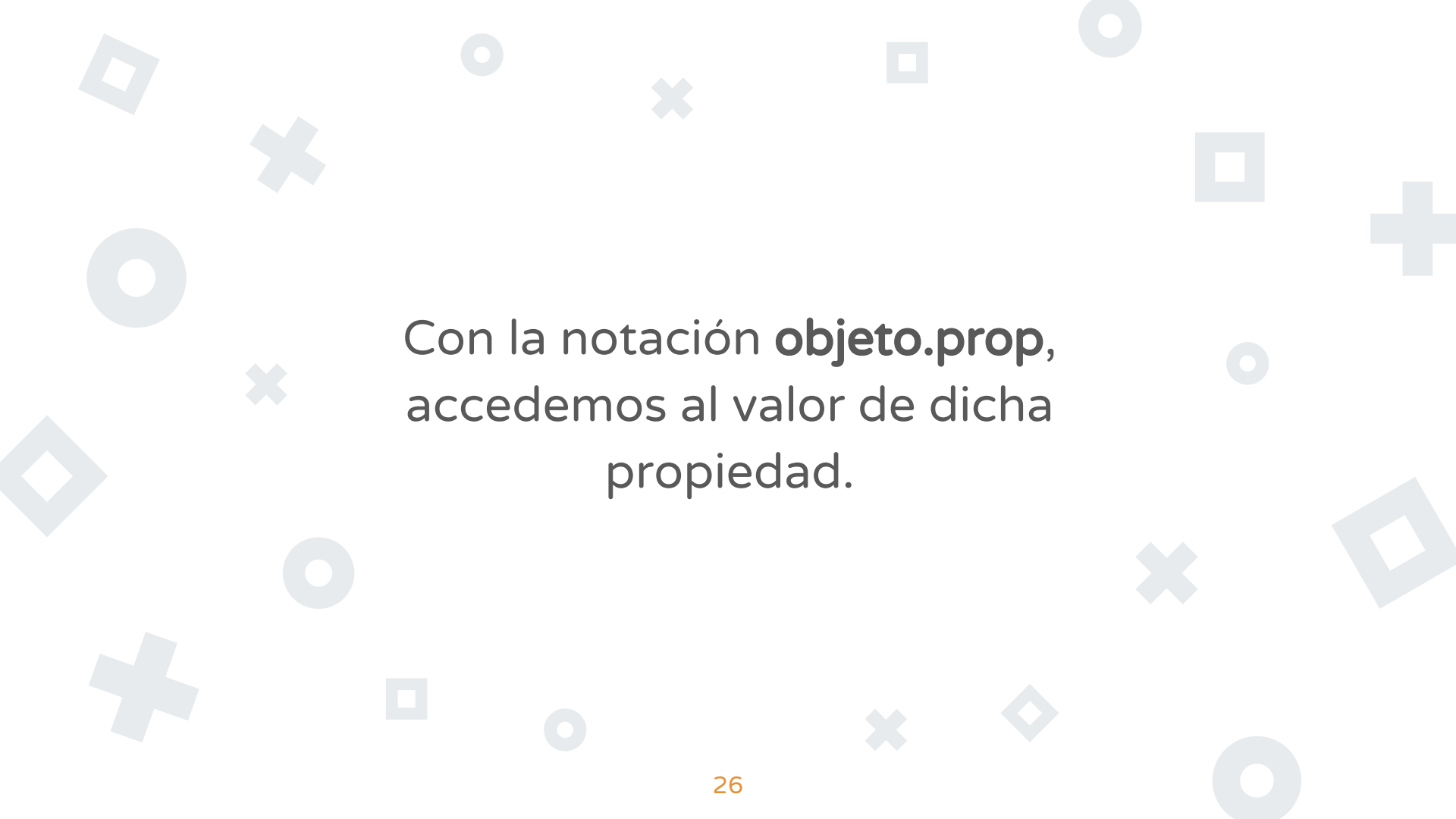
Si una propiedad recibe como valor una **función**, a esto lo llamamos un **método**.



# Objetos literales

```
var student = {  
  name: "Juana",  
  lastName: "Heinz",  
}
```

Ahora ¿cómo accedemos al nombre del estudiante?

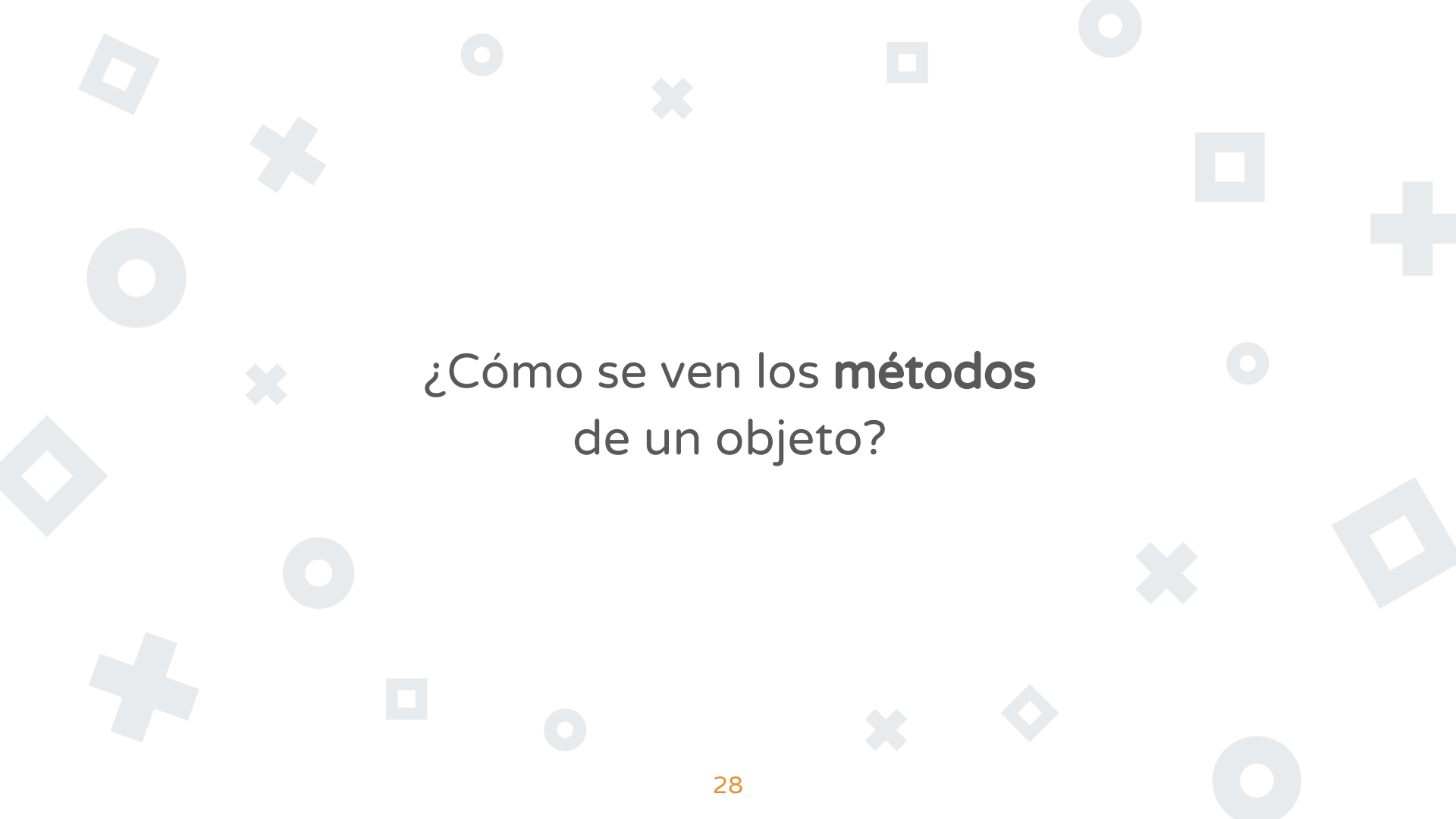


Con la notación **objeto.prop**,  
accedemos al valor de dicha  
propiedad.

# Objetos literales

```
var student = {  
  name: "Juana",  
  lastName: "Heinz",  
}
```

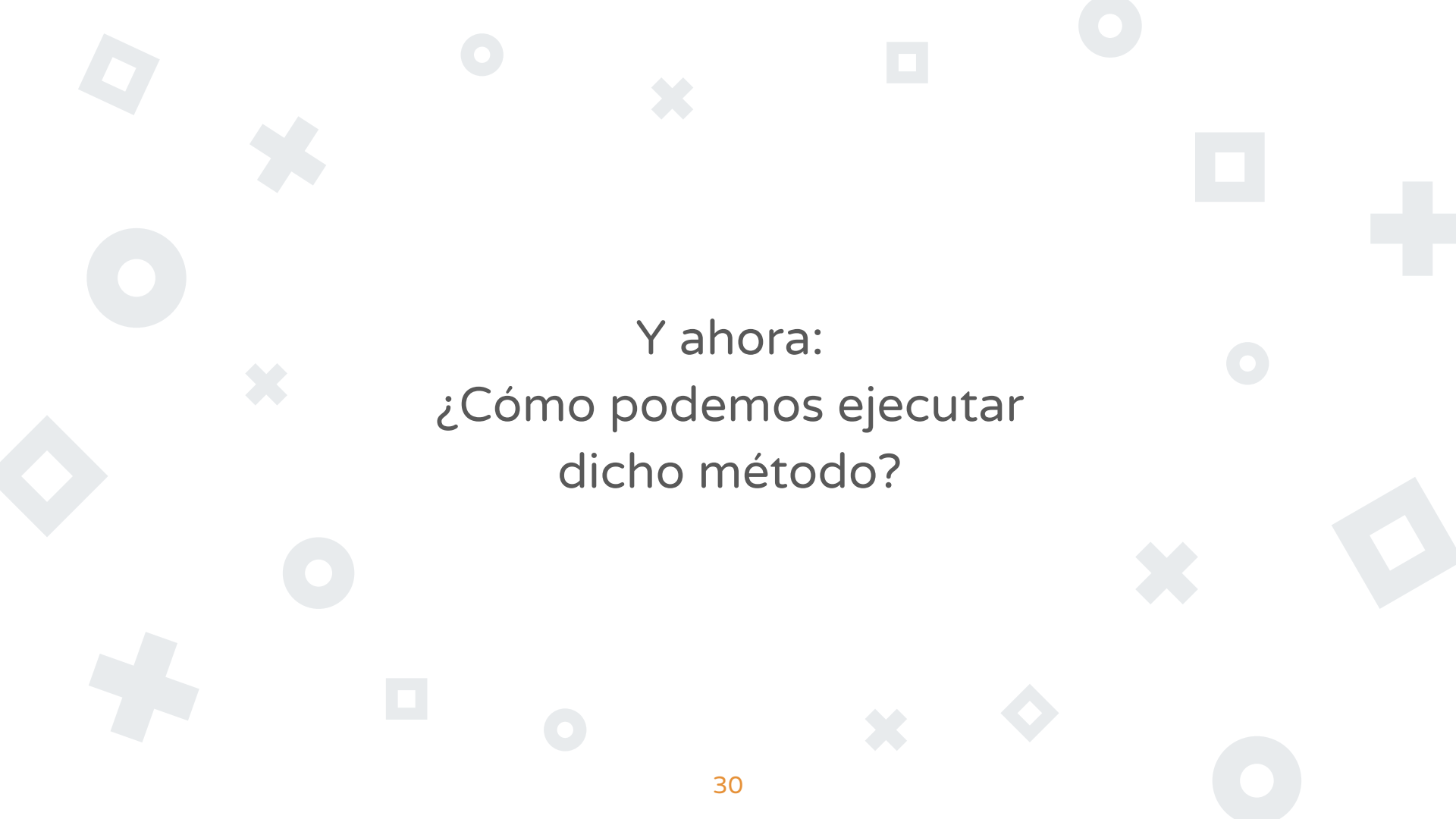
```
console.log(student.name); // "Juana"
```



¿Cómo se ven los **métodos**  
de un objeto?

# Objetos literales

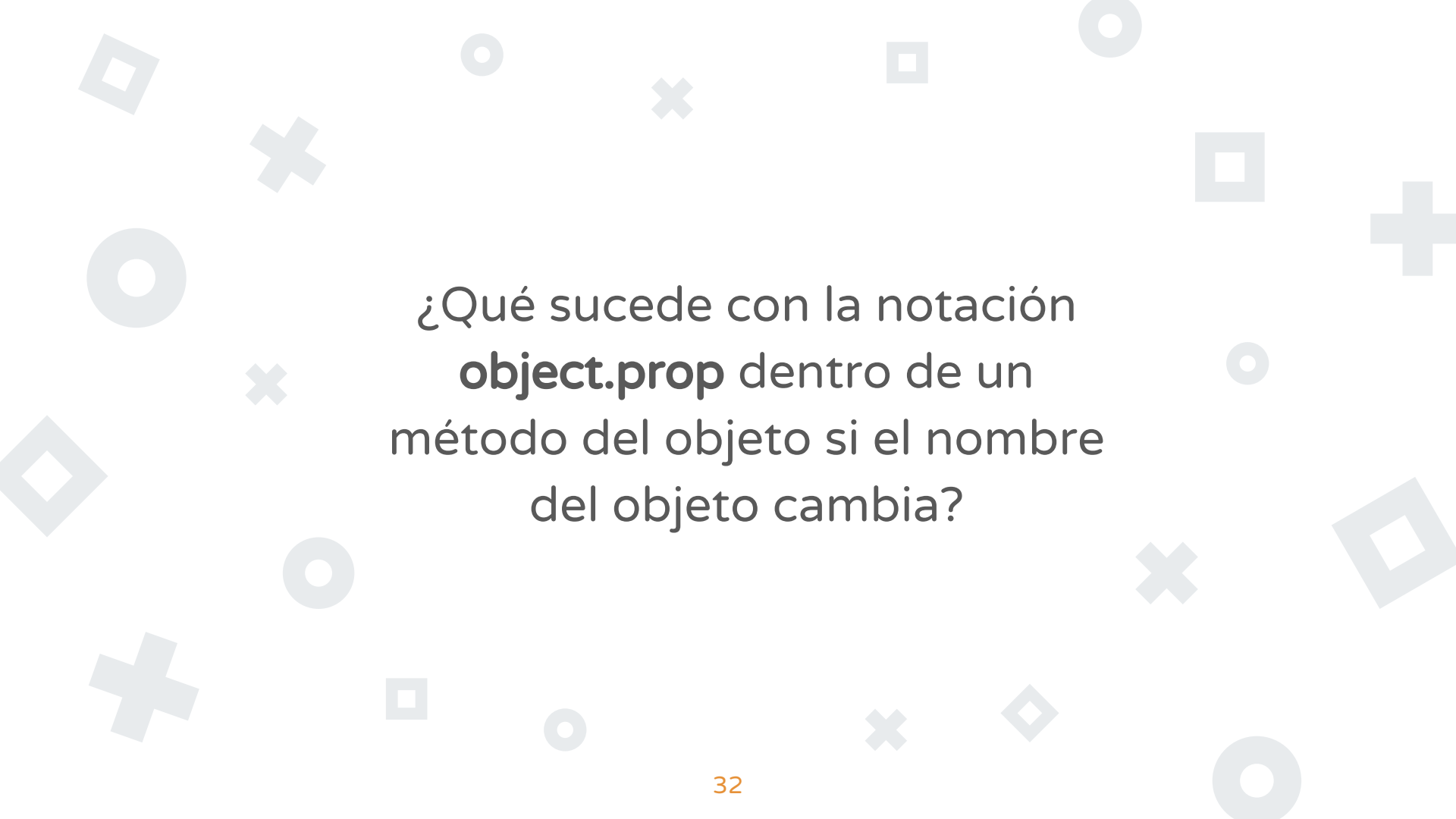
```
var student = {  
  name: "Juana",  
  lastName: "Heinz",  
  fullName: function () {  
    return student.name + " " + student.lastName;  
  }  
}
```



Y ahora:  
¿Cómo podemos ejecutar  
dicho método?

# Objetos literales

```
var student = {  
  name: "Juana",  
  lastName: "Heinz",  
  fullName: function () {  
    return student.name + " " + student.lastName;  
  }  
}  
console.log(student.fullName()); // "Juana Heinz"
```



¿Qué sucede con la notación  
**object.prop** dentro de un  
método del objeto si el nombre  
del objeto cambia?



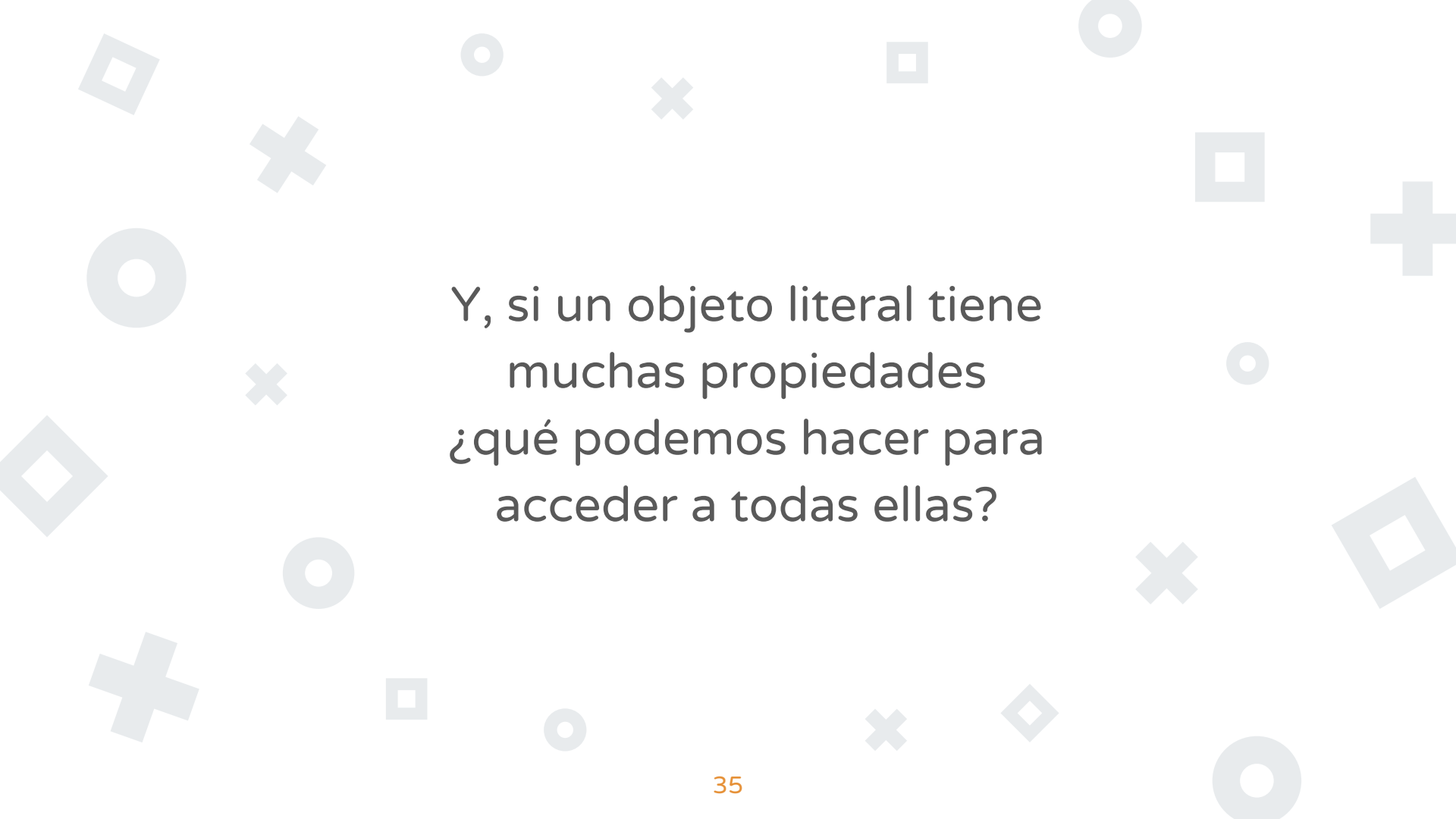
# Objetos literales

```
var juanitaEstudiante = {  
  name: "Juana",  
  lastName: "Heinz",  
  fullName: function () {  
    return this.name + " " + this.lastName;  
  }  
}  
  
console.log(student.fullName()); // "Juana Heinz"
```



La palabra reservada **this** hace referencia al mismo objeto en sí.

Se usa bastante incluso en otros ámbitos que no sean objetos literales.

The background of the slide is white and decorated with various light gray geometric shapes. These include squares, circles, and crosses. Some of these shapes are hollow, while others are solid. They are scattered across the slide, with a higher concentration around the edges, creating a subtle border effect. The shapes vary in size and orientation, adding a modern, minimalist aesthetic to the presentation.

Y, si un objeto literal tiene  
muchas propiedades  
¿qué podemos hacer para  
acceder a todas ellas?

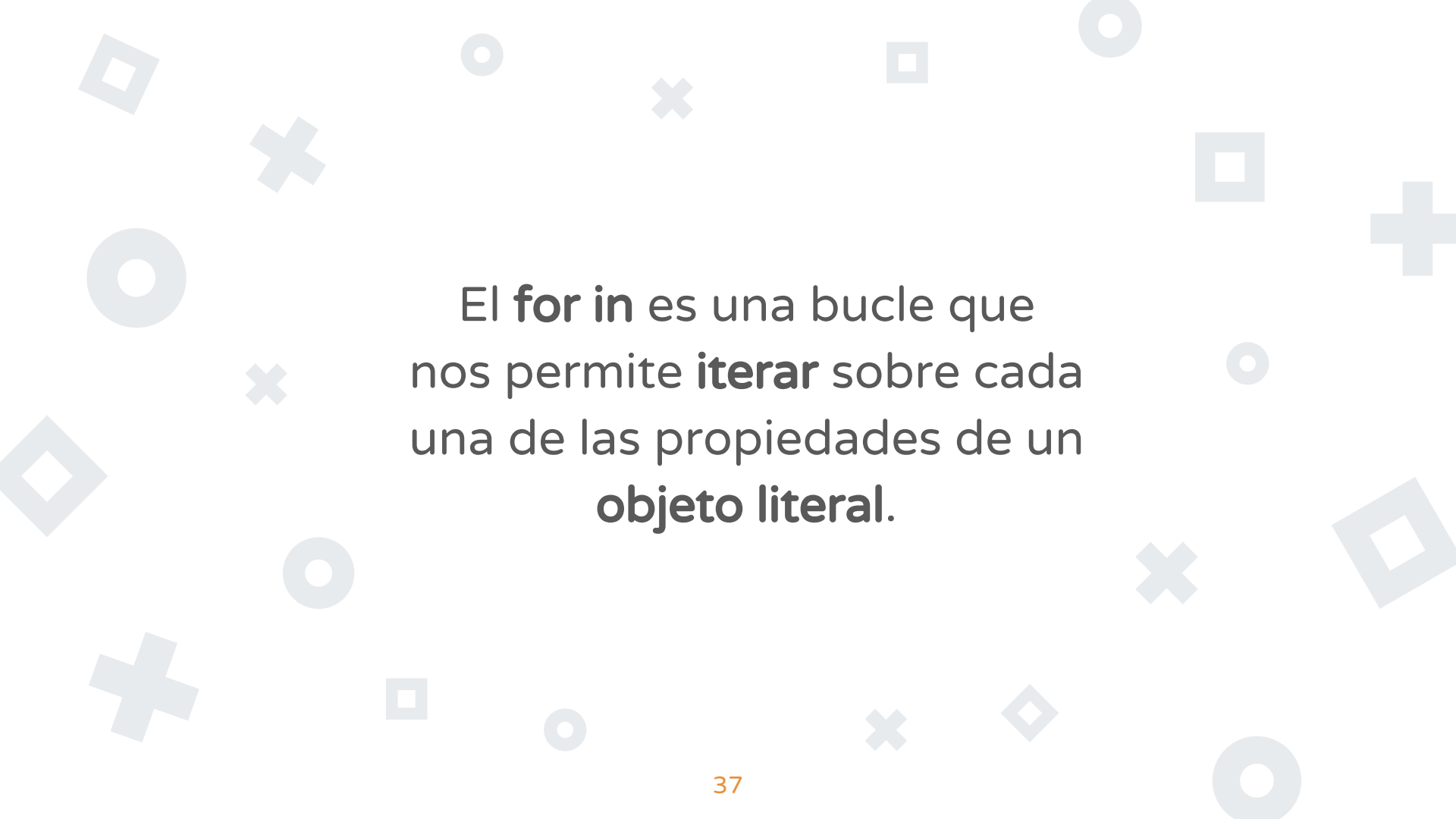
# Objetos literales

```
for (var key in student) {  
    console.log(student[key]);  
}
```

```
// "Juana"
```

```
// "Heinz"
```

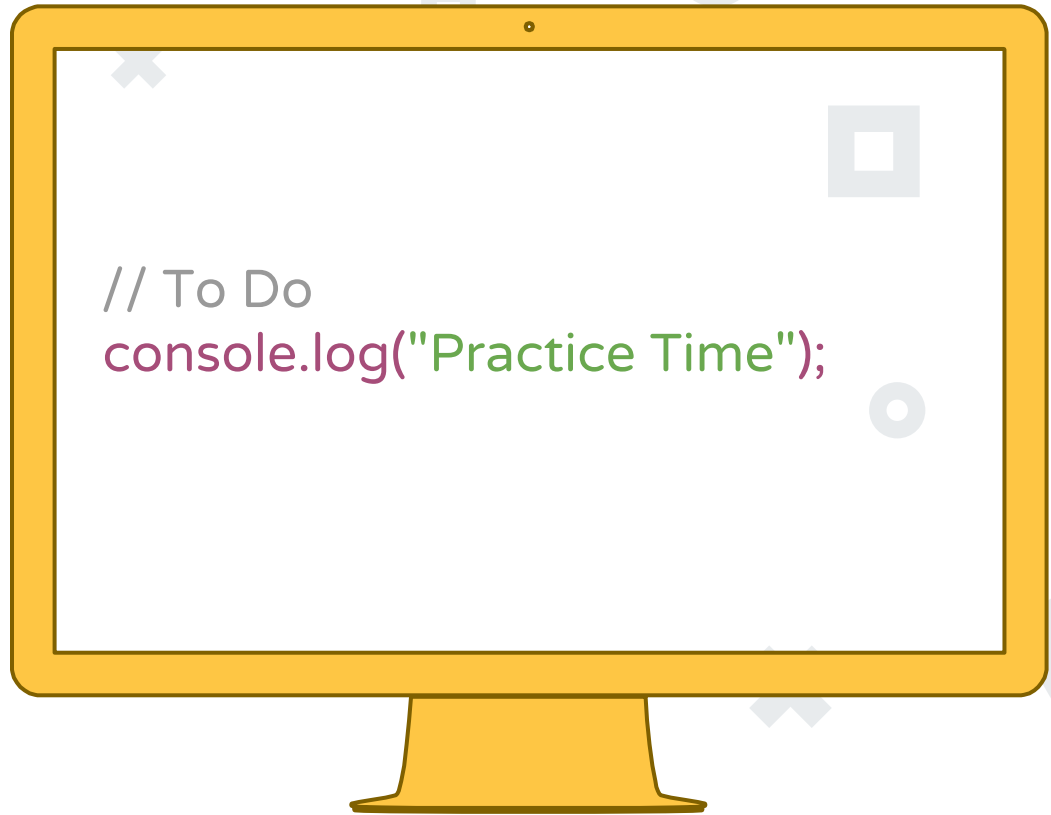
```
// "f()" ¿?
```



El **for in** es una bucle que  
nos permite **iterar** sobre cada  
una de las propiedades de un  
**objeto literal**.

# ¡A practicar!

Ejercicios Objetos  
Literales del 1 al 4

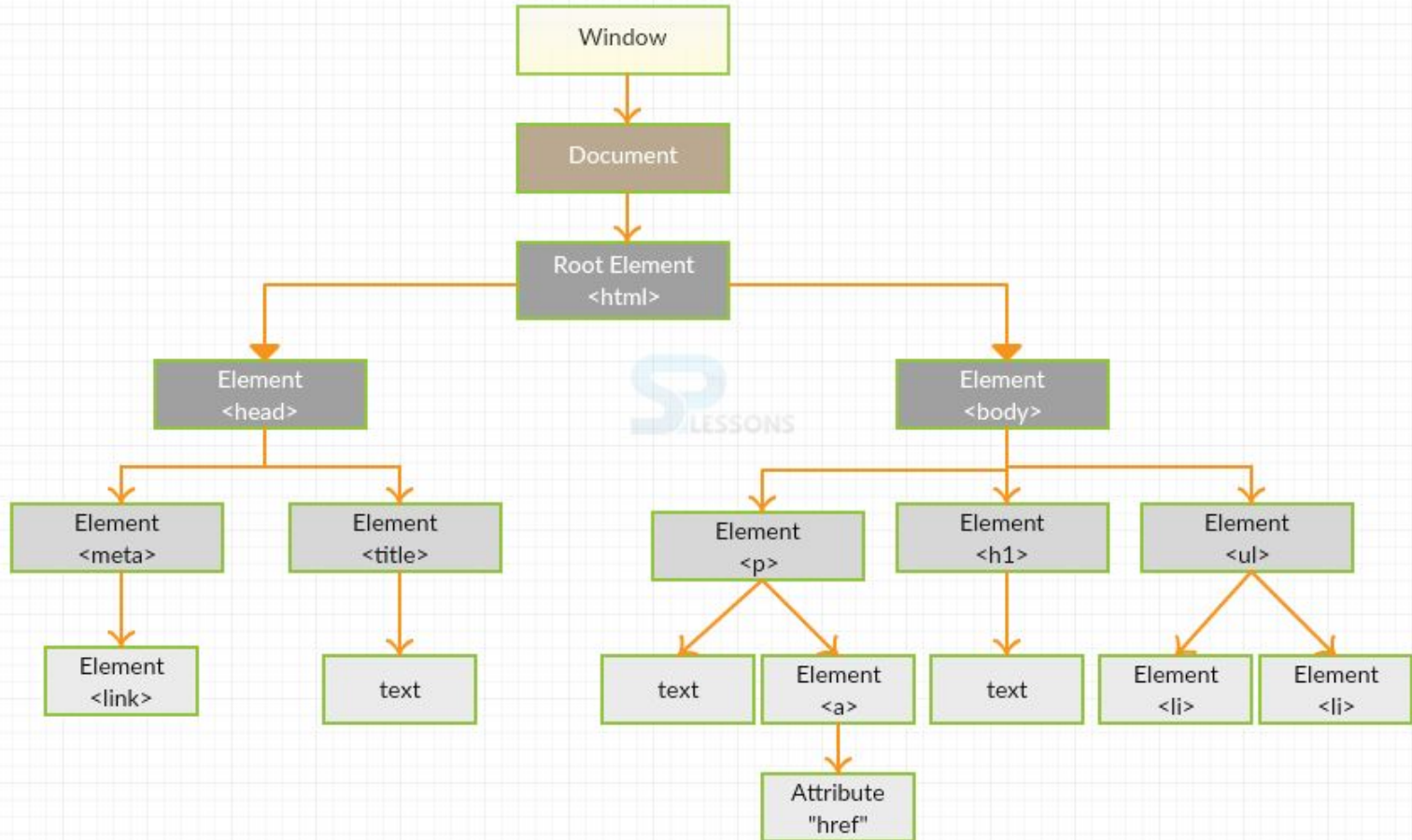




# D.O.M.

Document Object Model

## HTML DOM Tree





# Document Object Model

## ¿Qué hace JS con el D.O.M.?

- Modificar elementos, atributos y estilos de una página.
- Borrar cualquier elemento y atributo.
- Agregar nuevos elementos o atributos.
- Reaccionar a todos los eventos HTML de la página.
- Crear nuevos eventos HTML en la página.

# Document Object Model

## window

El objeto *window* es lo **primero** que se carga en el navegador. El objeto *window* tiene propiedades como *length*, *innerWidth*, etc.

## document

El objeto *document* representa al html, php u otro documento que será cargado en el navegador. El *document* es cargado dentro del objeto *window* y tiene propiedades como *title*, *url*, *cookie*, etc.



¿Cómo capturamos a los  
elementos existentes en el HTML?

# Selectores

Para acceder a los **elementos** de una página utilizamos selectores.

Cada selector puede retornar un **solo elemento** o una **lista de elementos**.

Algunos selectores básicos son (*reciben un string como parámetro*):

- `document.querySelector();`
- `document.querySelectorAll();`
- `document.getElementById();`

# Selectores

```
<body>  
  <h1 class="title">¡Hola mundo!</h1>  
</body>
```

```
// main.js  
var titulo = document.querySelector(".title");
```



¿Cuál es el objetivo de capturar a  
algún elemento de HTML?

# Propiedad style

Lo primero que podríamos hacer sería cambiar los estilos CSS de un elemento a nuestro antojo.

```
// main.js
```

```
var titulo = document.querySelector(".title");  
titulo.style.color = "pink";  
titulo.style.textAlign = "center";  
titulo.style.fontSize = "23px";  
titulo.style.backgroundColor = "#f2f2f2";
```

**¡A practicar!**

Ejercicios D.O.M.  
del 1 al 4

