This in JS

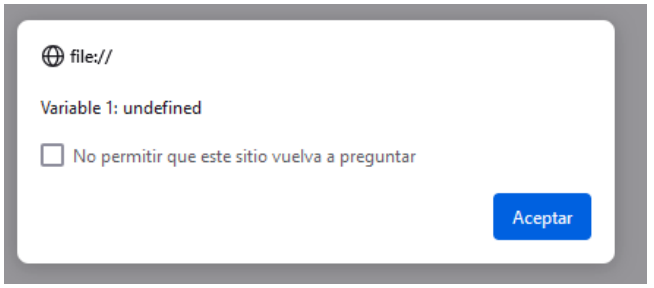
THIS

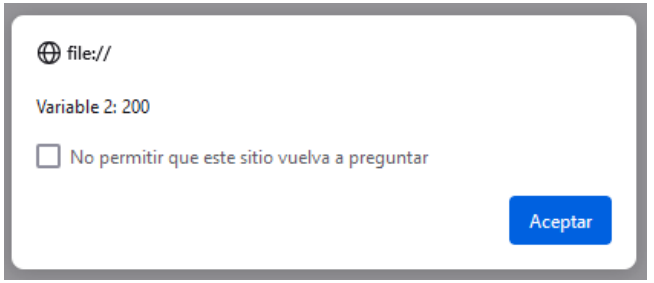
- En JavaScript, la palabra clave this se refiere al contexto en que se ejecuta el código, y su valor puede variar dependiendo de cómo y dónde se utilice

```
<body>
    <script>
        'use strict';

        alert(this);

    </script>
</body>
```

01_this_body.html

```html
<script>
    //'use strict'; //--Desactivado

    //variable 1
    //No se cuelga del objeto global (window)
    //No es accesible desde this.variable1


    //variable 2
    //Si activas "use strict" no funciona
    //Se cuelga automáticamente en window (global)

    let variable1 = 50; // como puedo leer esto con this (con LET)
    var variable2 = 200; // como puedo leer esto con this (con Var )pero en -stricto NO-

    function saludar() {
        alert(this);//

        alert("Variable 1: " + this.variable1); //undefined con LET
        alert("Variable 2: " + this.variable2); //lee bien variable2 con "var"
    }


    saludar();
</script>
```

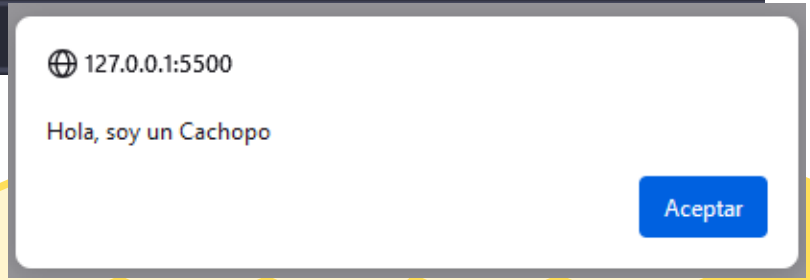




02_this_body_Funcion.html

```
<script>
    'use strict';
    const persona = {
        nombre: "Cachopo",

        saludar: function () {
            alert("Hola, soy un " + this.nombre);
        }
    };
    persona.saludar(); // Hola soy Cachopo
</script>
```

03_this_Clase.html

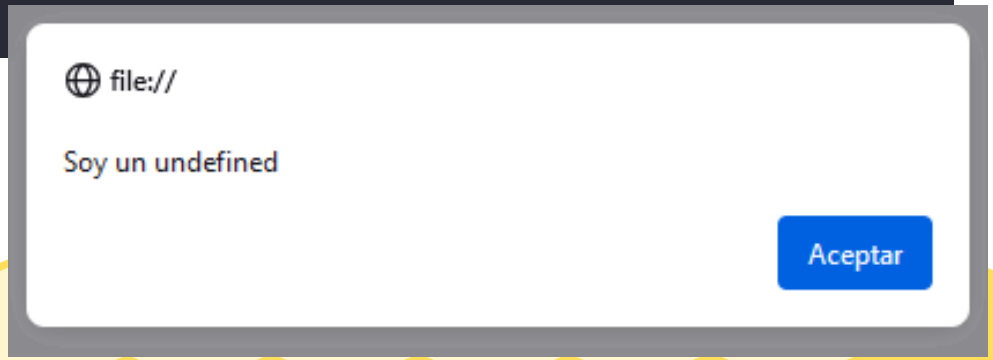


¿Caracteristicas de funciones flecha con this?

- Evitan confusiones con **this**
- No crean su propio **this**
- Heredan el **this** del contexto donde se definieron
- Pero son perfectas para callbacks (eventos) y funciones anidadas (funciones dentro de otras funciones)

05_diferencias_Clase_Y_Objeto_Literal.html

```
<script>
  'use strict';
  const persona = {
    nombre: "Cachopo",

    saludar: () => {

      alert("Soy un " + this.nombre);// "Soy un  UNDEFINED"
      alert(this);// hace referencia a [object Window] (afuera de la funcion)
    }
  };
  persona.saludar(); // Hola soy undefined
</script>
```



Comparativa: Class vs Constructor Function vs Object Literal

Característica	Class	Constructor Function	Object Literal
Definición	Sintaxis moderna para definir clases (desde ES6)	Función que actúa como molde para objetos (antes de ES6)	Objeto único definido con propiedades y métodos
Instanciación	Usa `new ClassName()`	Usa `new FunctionName()`	No se instancia, es un único objeto literal
Herencia	Sí, con `extends`	Sí, con prototipos y `call` / `apply`	No soporta herencia directa (solo prototipos manuales)
`this`	Apunta a la instancia creada	Apunta a la instancia creada	Depende del contexto donde se use
Métodos	Definidos en el prototipo automáticamente	Definidos dentro de la función constructora o prototipo	Definidos directamente como propiedades del objeto
Uso típico	Programación orientada a objetos moderna	POO clásica y retrocompatibilidad	Estructuras simples y configuraciones
Sintaxis métodos	Métodos abreviados (`saludar() {}`)	Funciones declaradas dentro del constructor o prototipo	Funciones asignadas como propiedades (normales o flechas)

🧨 CUIDADO con `this` en eventos DOM:

```js
const boton = document.querySelector("button");
boton.addEventListener("click", function () {
  console.log(this); // el botón
});
```

Pero si usas flecha:

```js
boton.addEventListener("click", () => {
  console.log(this); // window (NO el botón)
});
```

05_diferencias_Clase_Y_Objeto_Literal.html

```
function Persona1() { //clase
    this.nombre = "Cachopo";
    this.saludar = () => {
        console.log("(persona 1) Soy un ", this.nombre + "."); // lo lee bien
    };
}

const p1 = new Persona1();
p1.saludar(); // "(persona 1) Soy un Cachopo"

const Persona2 = { //objeto literal
    nombre: "Cachopo",
    saludar: () => {
        console.log("(persona 2) Soy un ", this.nombre + "."); //   un undefined
    }
};

Persona2.saludar(); // "(persona 2) Soy un UNDEFINED"
```

Bind - Call - Apply

🎯 RESUMEN CORTO

Método	¿Cuándo?	¿Qué hace?
`bind`	Quieres una función para usar después	Fija el `this` pero **no la ejecuta**
`call`	Quieres ejecutarla **ya** con `this`	Ejecuta con `this` y argumentos separados
`apply`	Igual que `call`, pero con array	Ejecuta con `this` y argumentos en array

No funcionan en funciones flechas

Bind

```
const hablar = function() {
  alert("Soy " + this.nombre);
};

const persona = { nombre: "Batman" };

const hablarComoBatman = hablar.bind(persona);

hablarComoBatman(); // Soy Batman
```

06_1_Bind_Hablar_como.html

⚡**Útil cuando:**

Necesitas usar la función más tarde (eventos, callbacks).

Quieres asegurarte de que this siempre sea el mismo.

Te da una copia de la función, pero con un **this** ya fijo.

No la ejecuta aún, solo la deja preparada.

Call

Ejecuta la función en el momento, usando el **this** que tú le digas, y le pasas los argumentos uno a uno.

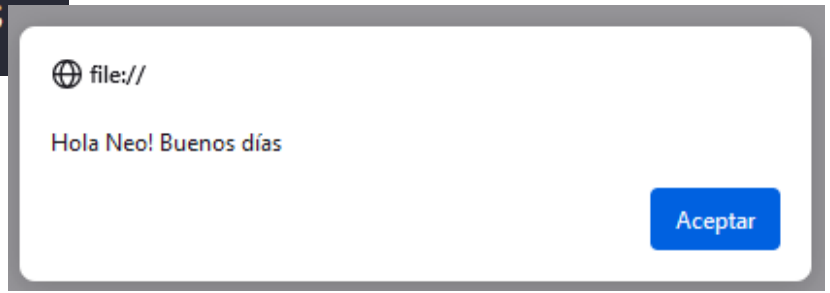
```
function saludar(nombre) {
  alert("Hola " + nombre + ", soy " + this.apodo);
}

const heroe = { apodo: "Spiderman" };

saludar.call(heroe, "Miguel"); // Hola Miguel, soy Spiderman
```

06_2_Call_Soy.html

Apply

```
function saludar(saludo, mensaje) {
    alert(`${saludo} ${this.nombre}! ${mensaje}`);
}

const persona = { nombre: "Neo" };


saludar.apply(persona, ["Hola", "Buenos días"]);
```

06_3_Apply_Array.html



Resumen

bind

Devuelve una nueva función con el contexto fijado a un objeto específico; **no ejecuta la función en ese momento**, solo la prepara para ejecutarse con ese contexto más adelante.

call

Ejecuta una función de **manera inmediata** utilizando un contexto específico, pasando los argumentos de forma individual.

apply

Ejecuta una función inmediatamente con un contexto específico, pero los **argumentos se pasan como un único array**.

Recursos en Web

Un artículo claro que explica detalladamente cómo funcionan call, apply y bind para controlar el contexto de this, cuándo se ejecutan y la diferencia entre ellos, con ejemplos:

https://www.luisllamas.es/como-funciona-call-apply-bind-en-javascript/

Otro post que desgrana qué es this en JavaScript y cómo usan esos tres métodos para modificarlo, con código y explicaciones paso a paso:

https://dev.to/khriztianmoreno/como-usar-los-metodos-javascript-call-apply-y-bind-32ek

Una explicación con ejemplos práctica en freeCodeCamp en español, que muestra como llamar funciones usando estos métodos y la diferencia entre ellos:

https://www.freecodecamp.org/espanol/news/como-usar-la-palabra-clave-this-en-javascript/

Un blog con una explicación sencilla y resumida de cada método, resaltando que bind crea una copia de la función con el contexto fijo, mientras que call y apply la ejecutan directamente:

https://www.juannicolas.eu/bind-call-y-apply-en-js/

Recursos en Youtube

Video explicativo sobre this y los métodos bind, call y apply de forma simple:

https://www.youtube.com/watch?v=bS71_W_BDFE

Video sobre para qué sirven y diferencias entre call, apply y bind en JavaScript:

https://www.youtube.com/watch?v=OZ02GSH9QkY

Curso completo con explicación específica de call, apply y bind:

https://www.youtube.com/watch?v=qy6IIjlykD0