

# Validaciones

# Capturar el formulario

El primer objetivo será obtener el formulario. Para esto tenemos dos opciones:

```
JS let formulario = document.querySelector("form.reservation")
```

```
JS let formulario = document.forms["reservation"]
```

# Eventos del formulario

El evento submit es aquel que se ejecuta cuando enviamos los datos.

```
JS formulario.addEventListener("submit", function(event){});
```

```
JS formulario.onSubmit = (event) => {}
```

# Validando los campos

Podemos obtener nuestro **input** con **querySelector** para que finalmente preguntemos si el valor campo está vacío.

```
JS event.preventDefault();  
let campoNombre = document.querySelector("input.nombre");  
if(campoNombre.value == ""){  
    alert("El campo nombre no debe estar vacío");  
}
```



Para detener el envío de formulario usamos:  
**event.preventDefault()**

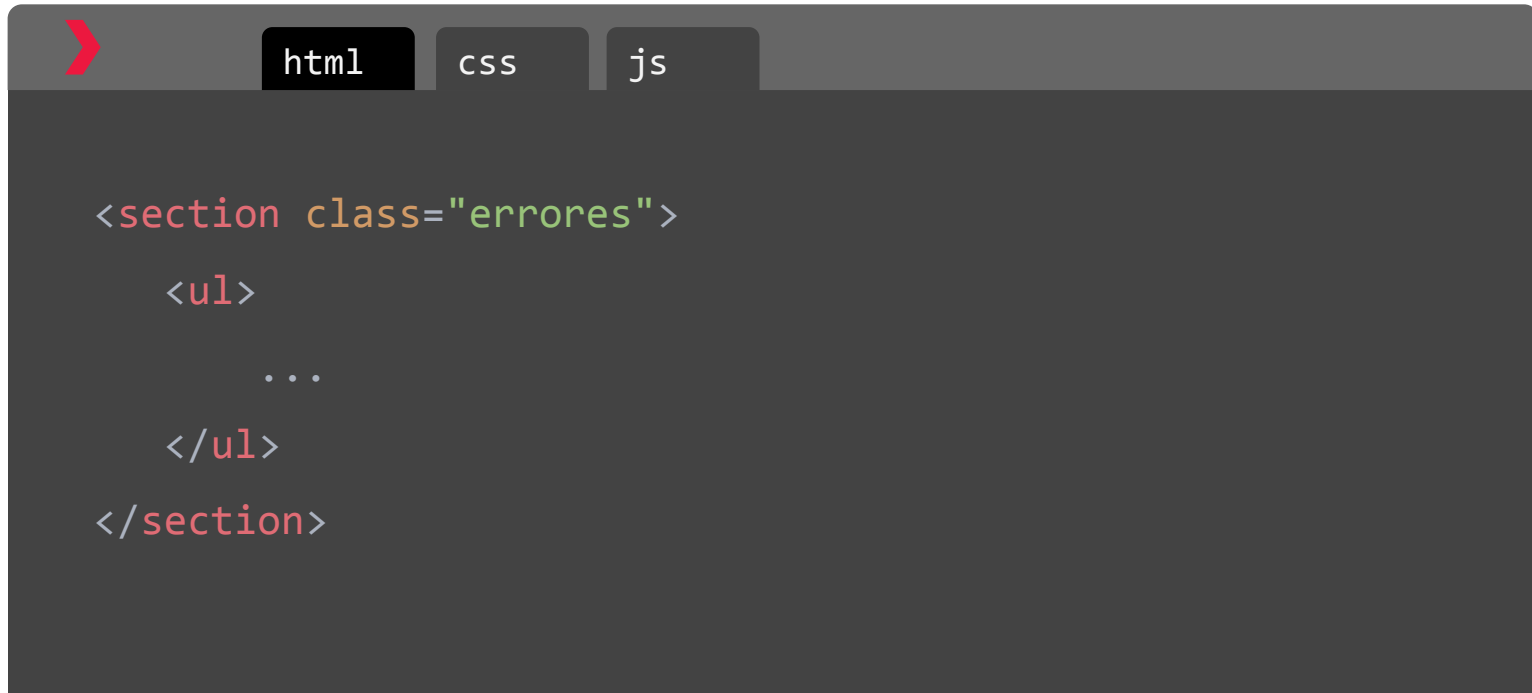
# Almacenar los errores

Creamos un **array** para acumular estos errores y cambiar nuestra lógica. Es decir, si el array no está vacío, entonces prevenimos el envío del formulario, caso contrario, el formulario se enviará.

JS

```
let errores = [];  
let campoNombre = document.querySelector("input.nombre");  
if(campoNombre.value == ""){  
    errores.push("El campo nombre está vacío");  
}  
if(errores.length > 0){  
    event.preventDefault();  
}
```

# Mostrando errores



A code editor interface with a dark gray background. At the top, there is a tab bar with three tabs: 'html' (selected), 'css', and 'js'. To the left of the tabs is a red arrow icon pointing right. The main area of the editor displays the following HTML code:

```
<section class="errores">  
  <ul>  
    ...  
  </ul>  
</section>
```

# En el javascript



html

css

js

```
if(errores.length > 0){  
    event.preventDefault();  
    let ulErrores = document.querySelector(".errores ul");  
    errores.forEach(error => {  
        ulErrores.innerHTML += `<li>${error}</li>`  
    });  
}
```

DigitalHouse>  
Coding School