Eventos de Formulario

onfocus

El evento “**focus**” sucede cuando el usuario ingresa con el cursor dentro de un campo input.

{}

```
let miInput = document.querySelector('#miInput');
miInput.onfocus = function(){
    alert('Ingresaste el cursor al campo :)');
}
```

onblur

Por el contrarios el evento “**blur**” sucede cuando el cursor abandona el campo en donde se encuentra.

{}

```
let miInput = document.querySelector('#miInput');
miInput.onblur = function(){
    alert('Saliste del campo :)');
}
```

onchange

El evento “**change**” permite identificar que el valor de un campo cambió. La ventaja que presenta el evento “**change**” es que podemos aplicarlo sobre cualquiera de los campos del formulario, inclusive sobre el formulario completo.

{}

```
let miInput = document.querySelector('#miInput');
miInput.onchange = function(){
    alert('Hubo un cambio por aquí...');
}
```

onsubmit

El evento “**submit**” identifica el momento en que se clickea sobre un botón o un campo input, ambos de tipo “**submit**”.

{}

```
let miForm = document.querySelector('#miForm');
miForm.onsubmit = function(){
    alert('El formulario se está por enviar...');
}
```

onsubmit y preventDefault()

Para evitar el envío de un formulario, necesitamos incluir a la función **preventDefault** en la primer línea dentro de la función.

```
let miForm = document.querySelector('#miForm');
miForm.onsubmit = function(event){
    if( algunCampoEnBlanco ) {
        event.preventDefault();
        alert('El formulario no se envió...');
    }else{
        alert('El formulario se está por enviar...');
    }
}
```

{}

setInterval

Se utiliza cuando queremos que nuestro código se ejecute una y otra vez, pasado un tiempo determinado.

```
const delay = 3000;
setInterval(miFuncion,delay);
function miFuncion(){
    alert("¡Este es mi anuncio repetido :) !")
}
```

Nota

miFuncion, es aquella función la cual voy a ejecutar por callback cuando se cumpla el tiempo determinado por **delay** (en milisegundos, en el ejemplo serían: 3 segundos).

clearTimeout y clearInterval

Ahora bien… ¿Puedo **detener** la ejecución de un timer cuando desee?

¡Obviamente! Tanto setTimeout como setInterval, pueden detenerse. Veamos el siguiente ejemplo con **clearInterval**:

{}

```
const delay = 3000;
const miIntervalo = setInterval(miFuncion,delay);
function miFuncion(){
    alert("¡Este es mi anuncio repetido :) !")
}
clearInterval(miIntervalo);
```