FrontEndII
Profesor: Martín Bonino

Módulo 3. Web Reactiva

Sesión 8. Eventos.
Marzo 17 de 2022

Evento
Es una acción que transcurre en el navegador o que es ejecutada por el usuario.

onload
permite que todo el script se ejecute cuando se haya cargado por completo el objeto document dentro del objeto window.

```
window.onload = function(){
	console.log('el documento está listo');
}
// Se suele escribir el código dentro de esta función para prevenir errores que pueden ocurrir si el documento n o está totalmente cargado al momento de la ejecución del script.
```

onclick
Permite ejecutar una acción cuando se haga clic sobre un elemento.

```
btn.onclick = function(){
	console.log('hiciste clic!');
}
```

preventDefault()
Permite evitar que se ejecute el evento predeterminado —o nativo— del elemento.

```
let hipervinculo = document.querySelector('a');
hipervinculo.addEventListener('click', function(event){
	console.log('hiciste click');
	event.preventDefault();
});
```

Eventos más usados:
onclick
ondblclick

onmouseover
onmousemove
onscroll
onkeydown
onload
onsubmit

Eventos del mouse

onmouseover
El evento se desencadena cuando se pasa el mouse por encima del elemento:

```
let texto = document.querySelector('.text');
texto.onmouseover = function(){
	console.log('pasaste el mouse');
}
```

onmouseout
Sucede cuando se retira el mouse después de haberlo posado sobre un elemento.

```
texto.addEventListener('mouseout', function(){
	console.log('quitaste el mouse');
});
```

Para aplicar un evento a varios elementos (en este caso de una misma clase):

```
window.addEventListener("load", function() {
	let botones = document.querySelectorAll(".w3-button")

	for (let I = 0; I < botones.length; i++) {
		botones[i].addEventListener("click", function() {
			this.style.color = "red"
		})
	}
})
```

Eventos del teclado

onkeydown
Cuando se presiona una tecla. A diferencia del evento keypress, keydown es lanzado para las teclas que producen un carácter y también para las que no lo producen.

onkeyup
Cuando se libera una tecla que estaba presionada

onkeypress
Cuando se ejecuta la acción de presionar y soltar una tecla.

Ejemplo

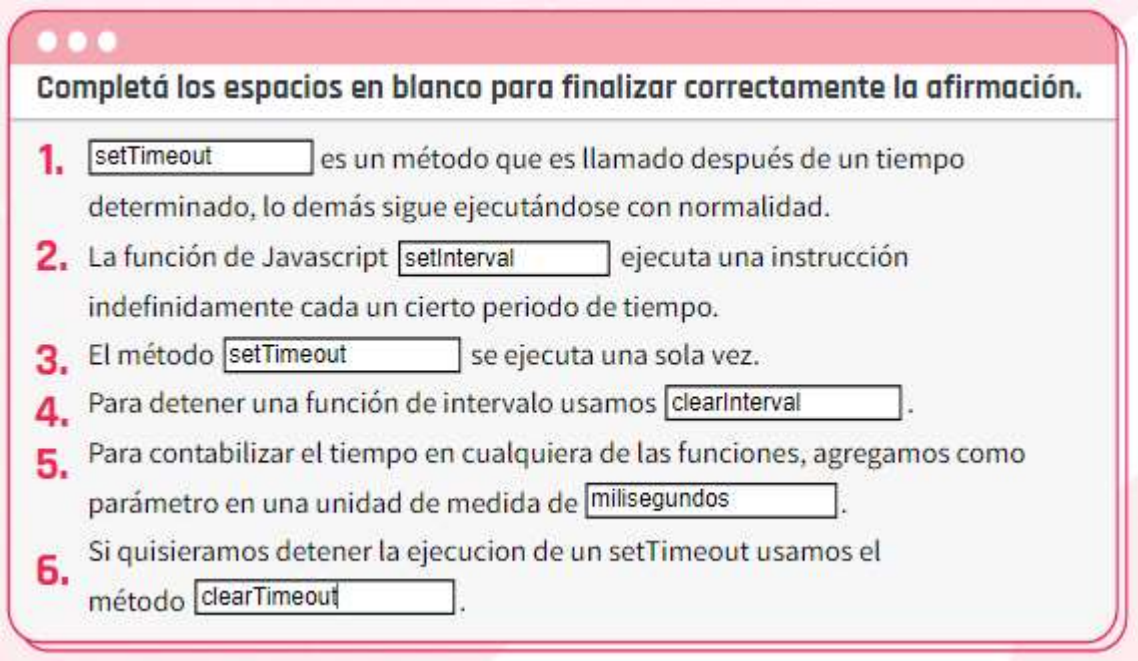
```
window.addEventListener("load", function() {
	window.addEventListener("keypress", function(e) {
		console.log("Se presionó la tecla " + e.key);
	})
})
```

Funciones como parámetros

```
function ejecutor(func) {
	// código de la función
	func(1, 2);
	// código de la función
}

function sumar(a, b) {
	return a + b;
}

ejecutor(sumar); // La función ejecutor  realiza todo su algoritmo y, cuando lo necesita, ejecuta la función func pasada como parámetro, que corresponde a la función suma. Se dice que la función ejecutador es la responsable de ejecutar la función sumar.
```



.

Tipos de Funciones en JavaScript

Funciones declaradas

Se ejecuta de forma independiente al resto del código planteado. Es decir, puede ser llamada antes de ser definida.

```
function saludar() {
	return “Hola”;
}
saludar();
```

Funciones expresadas

Tiene asignada una variable. Se crea cuando la ejecución del código llega a la línea donde está definida.

```
let saludar = function() {
	return “Hola”;
}
```

Callback

Es una función que se pasa como parámetro a otra función.

```
function saludar(nombre) {
	return “Hola, “ + nombre;
}

function procesarEntradaUsuario(callback) {
```

```
        var nombre = prompt("Por favor, ingresá tu nombre:";
        callback(nombre);
}
```

Closure
Es una función anidada a otra, con la que se puede acceder a las variables de la función externa y tener la posibilidad de sobreescribirlas.

```
function creaFunc() {
        var nombre = "Mozilla";
        function muestraNombre() {
                alert(nombre);
        }
        return muestraNombre;
}
creaFunc();
```

Palabras reservadas para declaración de variables

var
El scope es su contexto de ejecución (la variable vive dentro de la función en la cual es declarada). El scope de una variable declarada fuera de una función es global.

let
Las variables asignadas con let solo son accesibles dentro de su bloque (el bloque es aquella porción de código que se encuentra definida entre llaves). Las variables declaradas fuera de un bloque de código tienen un scope global.

const
Su comportamiento es igual que el de let, salvo por la diferencia de que no se puede reasignar su valor.

Nota sobre const, let y var
Lo ideal al momento de declarar una variable es utilizar siempre la palabra reservada const, salvo que en algún momento necesitemos reasignar su valor. En esos casos, se puede implementar la palabra reservada let.
Prácticamente no se debería utilizar la palabra var, sin embargo, si en algún momento resulta necesario utilizarla, entonces posiblemente se requiere refactorizar el código.

Sincrónico

Programación de un cronómetro:

```
// Para iniciar el conteo
let cont = setInterval( function() {
        number.innerHTML = n++;
}, 1000)

// Para detener el conteo
let btn = document.getElemenById(“btn”);
btn.addEventListener(“click”, ()=> { clearInterval(cont)});
```

Sesión 11. Formularios
Marzo 22 de 2022

Elementos de Formularios

```
// input de texto
<input type="text">

// input que solo admite números
<input type="number">

// input para campos de email
<input type="email">

// input de fecha
<input type="date">

// grupo de opciones de selección única
<input type="radio" name="miOpcion" value="1">
<input type="radio" name="miOpcion" value="2">
<input type="radio" name="miOpcion" value="3">

// grupo de opciones de selección múltiple
<input type="checkbox" name="miOpcion" value="1">
<input type="checkbox" name="miOpcion" value="2">
<input type="checkbox" name="miOpcion" value="3">
```

Para los casos de radio y checkbox son importantes los campos de name y de value, ya que definen el grupo al que pertenecen y el valor que se entrega en caso de ser seleccionado, respectivamente.

Select

```
<select>
	<option value="opcion0"> nombre de la opción 0</option>
	<option value="opcion1"> nombre de la opción 1</option>
	<option value="opcion2"> nombre de la opción 2</option>
</select>
```

Textarea

```
<textarea><textarea>
```

Evitar envío del formulario

```
let formulario = document.querySelector("form");
formulario.addEventListener("submit", function(event) {
	event.preventDefault()
})
```

Limitar el tamaño de caracteres de un textarea

```
function limita(maximoCaracteres) {
	var elemento = document.getElementById("texto");
	if(elemento.value.length >= maximoCaracteres ) {
		return false;
	}
	else {
		return true;
	}
}
// El valor por defecto de los eventos en JavaScript es true. Si se cambia  por false, se evita que el evento se produzca, por lo tanto, al hacerlo con el evento onkeypress, la tecla presionada no se transforma en ningún carácter dentro del textarea.
```

Obtener datos de un formulario

Atributo value:

```
<input type="radio" name="medio" value="Efectivo">
<input type="radio" name="medio" value="Débito" checked>
```

El atributo value almacena la información que será enviada si el usuario seleccionada un radio-button específico.

```
var elementos = document.getElementsByName("medio");
elementos.forEach(function(elemento) {
	console.log(`Elementos: ${elemento.value}`)
	console.log(`Seleccionado: ${elemento.checked}`)
})

// Elemento: Efectivo Seleccionado: false
// Elemento: Débito Seleccionado: true
```

Almacenar datos de input text y number. Revisar.

```
<input type="text" id="nombre" value="OpcionA">
<input type="number" id="numero" value="OpcionB">
```

```
var nombre = document.getElementById("nombre").value;
console.log(nombre) // OpcionA
var numero = document.getElementById("numero").value;
console.log(numero) // OpcionB
```

Almacenar el valor de un checkbox

```
<input type="checkbox" id="privacidad" value="privacidad">
He leído la política de privacidad

var privacidad = document.getElementById("privacidad");
console.log(`Elementos: ${privacidad.value}`)
console.log(`Seleccionado: ${privacidad.checked}`)
```

Normalización de Datos. Métodos de String

.toUpperCase()
.toLowerCase()
.concat()
. trim(): Elimina los espacios en blanco al inicio y al final de un string
.replace()
.replaceAll()

Ejemplos

```
nombre.concat(" ", apellido);

numero.replaceAll(".", ""); // Elimina los puntos almacenados en un número.
```

Validar:
Establecer una serie de reglas que debe cumplir un dato para ser correcto.

Normalizar:
Organizar los datos de manera tal que respeten el formato deseado para ser enviados o posiblemente almacenados en una base de datos.

Sincrónico

Normalización
Garantizar que los datos que ingresa el usuario sea vean igual toUpperCase(), toLowerCase().

Validación
Garantizar que los datos que el ingreso cumpla con los requisitos establecidos.

getElementsByName

Expresiones regulares:
regex.com

Entregable 1

.

Sesión 12.

Sincrónico

append vs appendChild?

Asignar el id dinámicamente:
div.id = `producto${prod.id}`

Vincular la API de MercadoPago