**Cuestiones Unidad 4 - Interacción con el usuario: eventos y formularios.**

#### Validar formularios y almacenar información en el navegador

Tenemos el siguiente código HTML:

<!DOCTYPE html>

<html lang="es">

<head>

<title>Eventos y validación de formularios</title>

</head>

<body>

<section>

<h3 id="titulo">Hola</h3>

<a href="http://www.google.es">Google</a>

</section>

</body>

</html>

1. Al hacer clic sobre el título h3, el texto del h3 tiene que cambiar por “Adiós”.

Escribe el código necesario, utilizando el modelo de eventos en línea.

<h3 id="titulo" onclick="this.innerHTML='Adiós'">Hola</h3>

1. Haz lo mismo que en el ejercicio anterior, pero utilizando el modelo de eventos del W3C.

document.addEventListener("DOMContentLoaded", function () {

let titulo = document.getElementById("titulo");

titulo.addEventListener("click", sust);

function sust(e) {

e.target.innerHTML = "Adiós";

}

});

1. Partiendo del ejercicio anterior, ¿cómo desenlazarías el manejador de eventos, para que al hacer clic ya no ocurra nada?

titulo.removeEventListener(“click”, sust);

1. En alguno de los tres modelos de eventos que hemos visto, ¿se pueden asignar varios manejadores a un evento?

Sí, en el modelo del W3C

1. ¿Podríamos utilizar el siguiente código para cambiar el color del texto cuando el usuario hace clic? Escríbelo de forma correcta en caso negativo.

<h3 id="titulo" onclick="this.style.color='red'">Hola</h3>

Sí funciona

1. ¿Podríamos utilizar el siguiente código para llamar a la función miFuncion cuando el usuario hace clic? Escríbelo de forma correcta en caso negativo.

<h3 id="titulo" onclick="miFuncion">Hola</h3>

<h3 id="titulo" onclick="miFuncion()">Hola</h3>

1. Al hacer clic sobre el enlace, se preguntará al usuario si estás seguro de que quieres ir a Google. En caso afirmativo, se irá a esa página, y en caso negativo, no se irá.

Cómo se haría utilizando el modelo de eventos en línea.

<a href="http://www.google.es" onclick="return confirm(‘¿Deseas ir a google?’)”>Google</a>

1. Repite la cuestión anterior utilizando el modelo de eventos del W3C.

<a href="http://www.google.es" id=”enlace”>Google</a>

document.addEventListener("DOMContentLoaded", function () {

let enlace = document.getElementById(“enlace”);

enlace.addEventListener(“click”, function(e) {

if(!confirm(‘¿Deseas ir a Google?’)) {

e.preventDefault;

};

});

});

1. Si e es un objeto de tipo *Event* (evento), ¿qué propiedades tienen la siguiente información?

|  |  |
| --- | --- |
| Elemento en el que se disparó el evento | e.target; |
| Elemento que es objetivo del evento | e.currentTarget; |
| Nombre del evento que se está gestionando (ej. click) | e.type; |

1. ¿Qué diferencia hay entre el evento **DOMContentLoaded** y el evento **load**? ¿Cuál se recomienda usar de ellos dos y por qué?

Load se produce cuando se han cargado todos los elementos (imágenes, estilos…) y DOMContentLoaded cuando se carga el árbol del DOM (HTML).

Mejor usar DOMContentLoaded porque va un poco más rápido.

1. Modifica el siguiente código para que se ejecute cuando la página haya cargado completamente:

document.addEventListener(“DOMContentLoaded”, () => {

let d = document.getElementById("divPrincipal"); d.addEventListener("click", muestraMensaje, false);

});

1. Explica brevemente lo que hace el siguiente trozo de código:

document.addEventListener("click", ev => {

if (ev.target.matches("#n1")) vaciar();

else if (ev.target.matches("#b1")) cuadrado();

});

El código lo que hace es: cuando se hace clic, hace una funcion. Mira el elemento en el que se disparó el evento con el target. Si coincide con el que tiene id=”n1” (con l propiedad .match), llama a la función vaciar. Si coincide con el que tiene id=”b1”, llama a la función cuadrado.

#### Validar formularios y almacenar información en el navegador

1. Si elemento es un objeto de tipo Element, ¿qué propiedades almacenan la siguiente información?

|  |  |
| --- | --- |
| Valor del elemento (en el caso de los input) | elemento.value |
| Si el elemento está chequeado o no (chekbox) | elemento.checked |

1. Si elemento es un objeto de tipo Element, y deseamos personalizar los mensajes de validación de HTML5, ¿qué propiedades y métodos JavaScript corresponden a las siguientes descripciones?

|  |  |
| --- | --- |
| Almacena el valor true si tiene un atributo required pero no tiene valor | elemento.validity.valueMissing |
| Contiene el mensaje que mostrará el navegador si el elemento no cumple la validación | elemento.validationMessage |
| Modifica el mensaje que mostrará el navegador si el elemento no cumple la validación | elemento.setCustomValidity() |

1. Indicar el resultado de evaluar las siguientes expresiones:

|  |  |
| --- | --- |
| "Hay 13 niños \nde 10 años".match(/^de/)[0] | null |
| "Hay 13 niños \nde 10 años".match(/[0-9]/g)[2] | [“1”, “3”, “1”, “0”] |
| "Hay 13 niños \nde 10 años".match(/[0-9]$/)[0] | null |
| "Hay 13 niños \nde 10 años".match(/hay/i)[0] | Hay |
| "Hay 13 niños \nde 10 años".match(/^de/m)[0] | de |

1. Indicar el resultado de evaluar las siguientes expresiones:

|  |  |
| --- | --- |
| "Hay 23 niños de 2 años".match(/[\wñáéíóú]+/ig)[2] | [“Hay”, “23”, “niños”, “de”, “2”, “años”] |
| "Hay 23 niños de 2 años".match(/(H)([ay]+)/)[2] | [‘Hay’, ‘H’, ‘ay’] |
| "Hay 23 niños de 2 años".match(/[a-z]+/)[0] | ay |
| "Hay 23 niños de 2 años".match(/\w+/ig)[1] | [“Hay”, “23”, “ni”, “os”, “de”, “2”, “a”, “os”] |
| "Hay 23 niños de 2 años".match(/[0-9]+/)[0] | 23 |

1. Indica el resultado de evaluar las siguientes expresiones:

|  |  |
| --- | --- |
| "niños".replace(/ños/, "") | ni |
| "niños".replace(/(ni)ñ(os)/, "$1") | ni |
| "niños".replace(/niñ(os)/, "añ" + "$1") | años |
| "niños".replace(/(ni)ñ(os)/, "$2") | os |
| "niños".replace(/ni/, "a") | años |

1. Crea una *cookie* de nombre Pepa que expire a los 3600 segundos. A continuación, modifica la *cookie* para que el nombre sea Pepe y por último elimina la *cookie*.

Document.cookie = “nombre=pepa; max-age=3600”

Document.cookie = “nombre=pepe”

Document.cookie = “nombre=pepa; max-age=0”

1. Utilizando *Web Storage*, almacena de forma persistente, incluso si cerramos el navegador, el nombre de usuario Pepa. A continuación, modifica ese nombre a Pepe y por último elimina esos datos.

localStorage.setItem(“nombre”, “Pepa”);

localStorage.setItem(“nombre”, “Pepe”);

localStorage.removeItem(“nombre”);

1. ¿Qué diferencia hay entre los contenedores localStorage y sessionStorage?

Se definen de diferente forma y se trabajan de de forma diferente se guarda en sitios diferentes localStorage es más fácil de utilizar.

1. Investiga las diferencias existentes entre las *cookies* y *Web Storage*:

Una de ellas mantiene la información después de cerrar el navegador y la otra no. Se usa más el localStorage.