**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.

onclick=” document.getElementById(‘titulo’).innerHTML = ‘adios’;”

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

document.getElementById(‘titulo’).addEventListener(‘click’, ()=>{

document.getElementById(‘titulo’).innerHTML = ‘adios’;

});

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

document.getElementById(‘titulo’).removeEventListener(‘click’);

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

¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?¿?

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

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

Así: ()

<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.

Onclick=’prompt(“”)? : e.prevenDefault’

¿¿?¿?¿?¿?¿?¿?¿?¿

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

Enlace.addEventListener(‘click’, función);

Function función (e){

If(!prompt(“”))

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 | El mismo???¿?¿?¿?¿?¿? |
| 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?

The DOMContentLoaded event will fire as soon as the DOM hierarchy has been fully constructed, the load event will do it when all the images and sub-frames have finished loading.

DOMContentLoaded==window.onDomReady()

Load==window.onLoad()

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

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

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();

});

#### 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) |  |
| Si el elemento está chequeado o no (chekbox) |  |

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 |  |
| Contiene el mensaje que mostrará el navegador si el elemento no cumple la validación |  |
| Modifica el mensaje que mostrará el navegador si el elemento no cumple la validación |  |

1. Indicar el resultado de evaluar las siguientes expresiones:

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

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".match(/(H)([ay]+)/)[2] |  |
| "Hay 23 niños de 2 años".match(/[a-z]+/)[0] |  |
| "Hay 23 niños de 2 años".match(/\w+/ig)[1] |  |
| "Hay 23 niños de 2 años".match(/[0-9]+/)[0] |  |

1. Indica el resultado de evaluar las siguientes expresiones:

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

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*.

Docment.cookie = “nombre=Pepa; max-age=3600”;

Document.cookie = “nombre=Pepe”;

Document.cookie=”nombre=; 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?
2. Investiga las diferencias existentes entre las *cookies* y *Web Storage*: