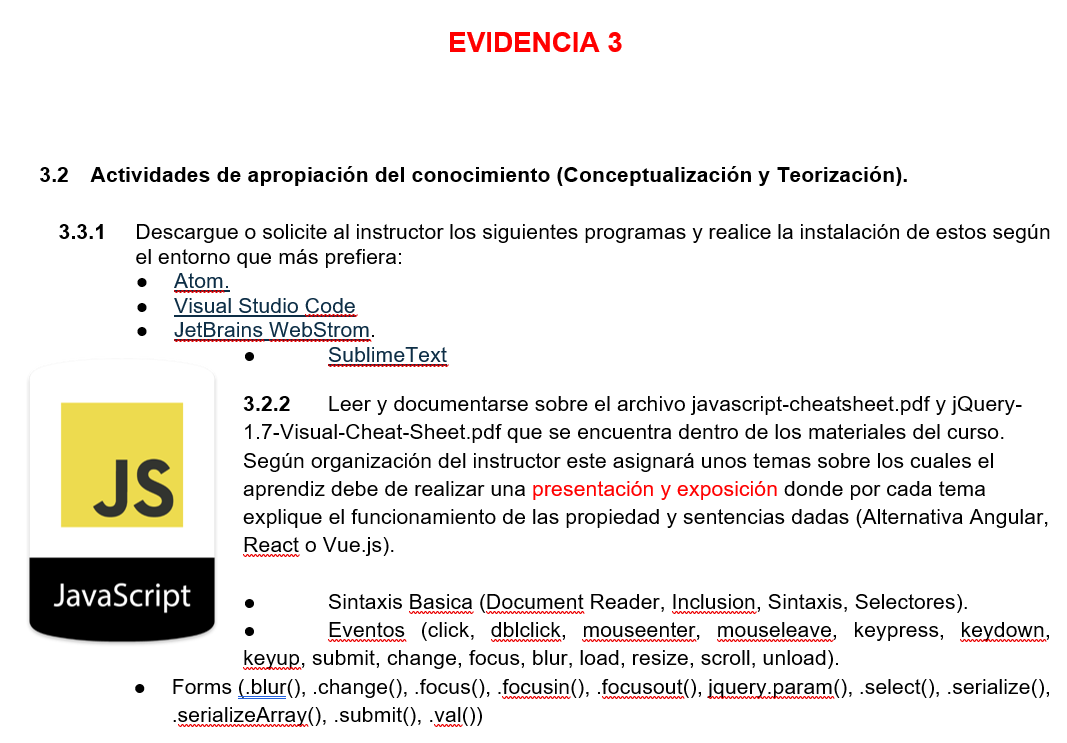
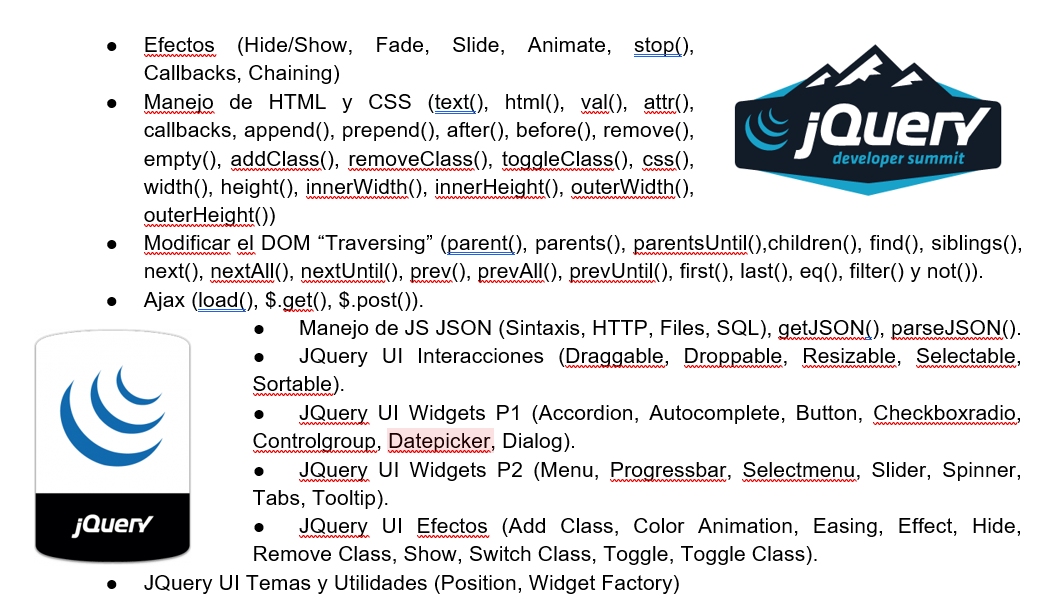
**SERVICIO NACIONAL DE APRENDIZAJE**

|  |  |  |  |
| --- | --- | --- | --- |
| **APRENDIZ:** | Marco Antonio Mesa Cáceres | | |
| **No. DOCUMENTO:** | 74362247 | **FICHA No.** | 2670142 |
| **OBJETIVO:** | **Desarrollar Actividad 03 de JavaScript** | | |
|  |

****

****

**DESARROLLO:**

**Sintaxis Básica**

**Document Reader**

El Document Object Model (DOM) es una representación en memoria del documento HTML. Con JavaScript, puedes leer y manipular este documento.

Ejemplo de acceso a elementos del DOM:

javascript

document.getElementById("miElemento");

document.getElementsByClassName("miClase");

document.getElementsByTagName("div");

document.querySelector("#miElemento");

document.querySelectorAll(".miClase");

**Inclusión**

Puedes incluir JavaScript en tu HTML de varias maneras:

1. **Interno**:

html

<script>

console.log("Hola Mundo");

</script>

1. **Externo**:

html

<script src="miScript.js"></script>

**Sintaxis**

JavaScript tiene una sintaxis basada en C:

* **Variables**:

javascript

var nombre = "Juan";

let edad = 30;

const PI = 3.1416;

* **Funciones**:

javascript

function saludar() {

console.log("Hola!");

}

saludar();

* **Condicionales**:

javascript

if (edad > 18) {

console.log("Eres mayor de edad.");

} else {

console.log("Eres menor de edad.");

}

* **Bucles**:

javascript

for (let i = 0; i < 10; i++) {

console.log(i);

}

**Selectores**

Los selectores se utilizan para seleccionar elementos del DOM:

javascript

document.getElementById("miElemento");

document.getElementsByClassName("miClase");

document.getElementsByTagName("div");

document.querySelector("#miElemento");

document.querySelectorAll(".miClase");

**Eventos**

**Eventos de Mouse**

* **click**:

javascript

document.getElementById("miBoton").addEventListener("click", function() {

alert("Botón clickeado");

});

* **dblclick**:

javascript

document.getElementById("miBoton").addEventListener("dblclick", function() {

alert("Botón doble clickeado");

});

* **mouseenter**:

javascript

document.getElementById("miElemento").addEventListener("mouseenter", function() {

console.log("Mouse entró");

});

* **mouseleave**:

javascript

document.getElementById("miElemento").addEventListener("mouseleave", function() {

console.log("Mouse salió");

});

**Eventos de Teclado**

* **keypress**:

javascript

document.addEventListener("keypress", function(event) {

console.log("Tecla presionada: " + event.key);

});

* **keydown**:

javascript

document.addEventListener("keydown", function(event) {

console.log("Tecla abajo: " + event.key);

});

* **keyup**:

javascript

document.addEventListener("keyup", function(event) {

console.log("Tecla arriba: " + event.key);

});

**Eventos de Formulario**

* **submit**:

javascript

document.getElementById("miFormulario").addEventListener("submit", function(event) {

event.preventDefault();

console.log("Formulario enviado");

});

* **change**:

javascript

document.getElementById("miInput").addEventListener("change", function() {

console.log("Valor cambiado");

});

* **focus**:

javascript

document.getElementById("miInput").addEventListener("focus", function() {

console.log("Input enfocado");

});

* **blur**:

javascript

document.getElementById("miInput").addEventListener("blur", function() {

console.log("Input desenfocado");

});

**Otros Eventos**

* **load**:

javascript

window.addEventListener("load", function() {

console.log("Página cargada");

});

* **resize**:

javascript

window.addEventListener("resize", function() {

console.log("Ventana redimensionada");

});

* **scroll**:

javascript

window.addEventListener("scroll", function() {

console.log("Ventana desplazada");

});

* **unload**:

javascript

window.addEventListener("unload", function() {

console.log("Página descargada");

});

**Métodos de Formularios en JavaScript**

**.blur()**

Este evento se dispara cuando un elemento pierde el foco.

javascript

document.getElementById("miInput").addEventListener("blur", function() {

console.log("Input perdió el foco");

});

**.change()**

Este evento se dispara cuando el valor de un elemento cambia.

javascript

document.getElementById("miInput").addEventListener("change", function() {

console.log("Valor cambiado");

});

**.focus()**

Este evento se dispara cuando un elemento gana el foco.

javascript

document.getElementById("miInput").addEventListener("focus", function() {

console.log("Input enfocado");

});

**.focusin()**

Este evento se dispara cuando un elemento o uno de sus hijos gana el foco. A diferencia de focus, se propaga.

javascript

document.getElementById("miInput").addEventListener("focusin", function() {

console.log("Input enfocado (focusin)");

});

**.focusout()**

Este evento se dispara cuando un elemento o uno de sus hijos pierde el foco. A diferencia de blur, se propaga.

javascript

document.getElementById("miInput").addEventListener("focusout", function() {

console.log("Input desenfocado (focusout)");

});

**.select()**

Este evento se dispara cuando el texto dentro de un <input> o <textarea> es seleccionado.

javascript

document.getElementById("miInput").addEventListener("select", function() {

console.log("Texto seleccionado");

});

**.submit()**

Este evento se dispara cuando se envía un formulario.

javascript

document.getElementById("miFormulario").addEventListener("submit", function(event) {

event.preventDefault();

console.log("Formulario enviado");

});

**.val()**

Método para obtener o establecer el valor de un elemento <input>.

javascript

let valor = document.getElementById("miInput").value; // Obtener el valor

document.getElementById("miInput").value = "Nuevo valor"; // Establecer el valor

**Métodos de Formularios en jQuery**

**.blur()**

javascript

$("#miInput").blur(function() {

console.log("Input perdió el foco");

});

**.change()**

javascript

$("#miInput").change(function() {

console.log("Valor cambiado");

});

**.focus()**

javascript

$("#miInput").focus(function() {

console.log("Input enfocado");

});

**.focusin()**

javascript

$("#miInput").focusin(function() {

console.log("Input enfocado (focusin)");

});

**.focusout()**

javascript

$("#miInput").focusout(function() {

console.log("Input desenfocado (focusout)");

});

**.select()**

javascript

$("#miInput").select(function() {

console.log("Texto seleccionado");

});

**.submit()**

javascript

$("#miFormulario").submit(function(event) {

event.preventDefault();

console.log("Formulario enviado");

});

**.val()**

Método para obtener o establecer el valor de un elemento <input> en jQuery.

javascript

let valor = $("#miInput").val(); // Obtener el valor

$("#miInput").val("Nuevo valor"); // Establecer el valor

**jquery.param()**

Este método crea una cadena de consulta serializada a partir de un objeto o una matriz.

javascript

let params = { name: "John", age: 30 };

let queryString = $.param(params);

console.log(queryString); // "name=John&age=30"

**.serialize()**

Serializa los datos del formulario en una cadena de consulta URL.

javascript

let datosSerializados = $("#miFormulario").serialize();

console.log(datosSerializados); // "name=John&age=30"

**.serializeArray()**

Serializa los datos del formulario en una matriz de objetos.

javascript

let datosArray = $("#miFormulario").serializeArray();

console.log(datosArray);

// [{ name: "name", value: "John" }, { name: "age", value: "30" }]

**.submit()**

Este evento se dispara cuando se envía un formulario. Puedes interceptarlo para ejecutar código antes de que el formulario se envíe o para evitar que se envíe.

javascript

document.getElementById("miFormulario").addEventListener("submit", function(event) {

event.preventDefault(); // Evita el envío del formulario

console.log("Formulario enviado");

});

**.val()**

Método para obtener o establecer el valor de un elemento <input>, <textarea>, o <select>.

javascript

// Obtener el valor de un input

let valor = document.getElementById("miInput").value;

console.log(valor);

// Establecer un nuevo valor en el input

document.getElementById("miInput").value = "Nuevo valor";

**Métodos de Formularios en jQuery**

**.submit()**

Este método se utiliza para capturar el evento de envío del formulario y puede prevenir el comportamiento predeterminado de enviar el formulario.

javascript

$("#miFormulario").submit(function(event) {

event.preventDefault(); // Evita el envío del formulario

console.log("Formulario enviado");

});

**.val()**

Método para obtener o establecer el valor de un elemento <input>, <textarea>, o <select> en jQuery.

javascript

// Obtener el valor de un input

let valor = $("#miInput").val();

console.log(valor);

// Establecer un nuevo valor en el input

$("#miInput").val("Nuevo valor");

**Efectos en jQuery**

**.hide() y .show()**

Estos métodos se utilizan para ocultar y mostrar elementos respectivamente.

javascript

// Ocultar un elemento

$("#miElemento").hide();

// Mostrar un elemento

$("#miElemento").show();

Puedes especificar una duración para que el cambio sea gradual.

javascript

// Ocultar un elemento gradualmente en 500 milisegundos

$("#miElemento").hide(500);

// Mostrar un elemento gradualmente en 500 milisegundos

$("#miElemento").show(500);

**.fadeIn() y .fadeOut()**

Estos métodos se utilizan para desvanecer elementos dentro o fuera de la vista.

javascript

// Desvanecer un elemento en 400 milisegundos

$("#miElemento").fadeIn(400);

// Desvanecer un elemento fuera de la vista en 400 milisegundos

$("#miElemento").fadeOut(400);

También puedes especificar una función de callback que se ejecutará después de que la animación se complete.

javascript

$("#miElemento").fadeOut(400, function() {

console.log("Animación completada");

});

**.slideDown() y .slideUp()**

Estos métodos se utilizan para deslizar elementos hacia abajo o hacia arriba.

javascript

// Deslizar un elemento hacia abajo en 400 milisegundos

$("#miElemento").slideDown(400);

// Deslizar un elemento hacia arriba en 400 milisegundos

$("#miElemento").slideUp(400);

**.animate()**

Este método se utiliza para crear animaciones personalizadas.

javascript

$("#miElemento").animate({

left: '250px',

opacity: '0.5',

height: '+=50px',

width: '+=50px'

}, 400);

**.stop()**

Este método detiene la animación actual de los elementos seleccionados.

javascript

$("#miElemento").stop();

**Callbacks**

Las funciones de callback se utilizan para ejecutar código después de que una animación se complete.

javascript

$("#miElemento").hide(400, function() {

console.log("El elemento se ha ocultado");

});

**Chaining**

El encadenamiento (chaining) permite ejecutar múltiples métodos en un solo selector jQuery.

javascript

$("#miElemento").slideUp(200).slideDown(200).fadeOut(200).fadeIn(200);

**Ejemplos Combinados**

**Ejemplo de Encadenamiento con Callbacks**

javascript

$("#miElemento")

.slideUp(200)

.slideDown(200)

.fadeOut(200, function() {

console.log("El elemento se ha desvanecido");

})

.fadeIn(200);

**Ejemplo de Uso de .stop()**

javascript

$("#start").click(function() {

$("#miElemento").animate({ left: '250px' }, 2000);

});

$("#stop").click(function() {

$("#miElemento").stop();

});

**Métodos de jQuery para Manejo de HTML y CSS**

**.text()**

Obtiene o establece el contenido de texto de los elementos seleccionados.

javascript

// Obtener el contenido de texto

let texto = $("#miElemento").text();

console.log(texto);

// Establecer el contenido de texto

$("#miElemento").text("Nuevo contenido de texto");

**.html()**

Obtiene o establece el contenido HTML de los elementos seleccionados.

javascript

// Obtener el contenido HTML

let contenidoHTML = $("#miElemento").html();

console.log(contenidoHTML);

// Establecer el contenido HTML

$("#miElemento").html("<p>Nuevo contenido HTML</p>");

**.val()**

Obtiene o establece el valor de los elementos de formulario.

javascript

// Obtener el valor

let valor = $("#miInput").val();

console.log(valor);

// Establecer el valor

$("#miInput").val("Nuevo valor");

**.attr()**

Obtiene o establece el valor de un atributo para los elementos seleccionados.

javascript

// Obtener el valor de un atributo

let src = $("#miImagen").attr("src");

console.log(src);

// Establecer el valor de un atributo

$("#miImagen").attr("src", "nuevaImagen.jpg");

**Callbacks**

Las funciones de callback se ejecutan después de que una operación se complete.

javascript

$("#miBoton").click(function() {

$("#miElemento").hide(400, function() {

alert("El elemento se ha ocultado");

});

});

**.append()**

Inserta contenido al final de los elementos seleccionados.

javascript

$("#miLista").append("<li>Nuevo elemento al final</li>");

**.prepend()**

Inserta contenido al principio de los elementos seleccionados.

javascript

$("#miLista").prepend("<li>Nuevo elemento al principio</li>");

**.after()**

Inserta contenido después de los elementos seleccionados.

javascript

$("#miElemento").after("<p>Contenido después del elemento</p>");

**.before()**

Inserta contenido antes de los elementos seleccionados.

javascript

$("#miElemento").before("<p>Contenido antes del elemento</p>");

**.remove()**

Elimina los elementos seleccionados del DOM.

javascript

$("#miElemento").remove();

**.empty()**

Elimina el contenido interno de los elementos seleccionados.

javascript

$("#miElemento").empty();

**.addClass()**

Agrega una o más clases a los elementos seleccionados.

javascript

$("#miElemento").addClass("miClase nuevaClase");

**.removeClass()**

Elimina una o más clases de los elementos seleccionados.

javascript

$("#miElemento").removeClass("miClase");

**.toggleClass()**

Agrega o elimina una clase de los elementos seleccionados, dependiendo de si están presentes o no.

javascript

$("#miElemento").toggleClass("miClase");

**.css()**

Obtiene o establece propiedades CSS de los elementos seleccionados.

javascript

// Obtener el valor de una propiedad CSS

let color = $("#miElemento").css("color");

console.log(color);

// Establecer el valor de una propiedad CSS

$("#miElemento").css("color", "blue");

// Establecer múltiples propiedades CSS

$("#miElemento").css({

"color": "blue",

"font-size": "18px"

});

**.width() y .height()**

Obtiene o establece el ancho y alto de los elementos seleccionados.

javascript

// Obtener el ancho y alto

let ancho = $("#miElemento").width();

let alto = $("#miElemento").height();

console.log("Ancho:", ancho, "Alto:", alto);

// Establecer el ancho y alto

$("#miElemento").width(200);

$("#miElemento").height(100);

**.innerWidth() y .innerHeight()**

Obtiene el ancho y alto internos de los elementos seleccionados, incluyendo el relleno (padding) pero no el borde.

javascript

let anchoInterno = $("#miElemento").innerWidth();

let altoInterno = $("#miElemento").innerHeight();

console.log("Ancho Interno:", anchoInterno, "Alto Interno:", altoInterno);

**.outerWidth() y .outerHeight()**

Obtiene el ancho y alto externos de los elementos seleccionados, incluyendo el relleno (padding) y el borde.

javascript

let anchoExterno = $("#miElemento").outerWidth();

let altoExterno = $("#miElemento").outerHeight();

console.log("Ancho Externo:", anchoExterno, "Alto Externo:", altoExterno);

**Ejemplo Combinado**

**javascript**

$(document).ready(function() {

// Cambiar el contenido de texto y HTML

$("#miTexto").text("Nuevo texto");

$("#miHTML").html("<strong>Nuevo contenido HTML</strong>");

// Manejar valores de input

$("#miInput").val("Nuevo valor");

// Manipular atributos

$("#miImagen").attr("alt", "Nueva descripción");

// Añadir y quitar clases

$("#miElemento").addClass("activo");

$("#miElemento").removeClass("activo");

$("#miElemento").toggleClass("activo");

// Añadir contenido

$("#miLista").append("<li>Elemento al final</li>");

$("#miLista").prepend("<li>Elemento al principio</li>");

$("#miElemento").after("<p>Después del elemento</p>");

$("#miElemento").before("<p>Antes del elemento</p>");

// Eliminar contenido o elementos

$("#miElemento").empty();

$("#miElemento").remove();

// Cambiar estilos CSS

$("#miElemento").css("color", "blue");

$("#miElemento").css({

"font-size": "20px",

"background-color": "yellow"

});

// Obtener y establecer dimensiones

let width = $("#miElemento").width();

let height = $("#miElemento").height();

$("#miElemento").width(300).height(200);

let innerWidth = $("#miElemento").innerWidth();

let innerHeight = $("#miElemento").innerHeight();

let outerWidth = $("#miElemento").outerWidth();

let outerHeight = $("#miElemento").outerHeight();

});

**Modificar el DOM "Traversing" en jQuery**

**.parent()**

Selecciona el padre directo de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miElemento").parent().css("border", "1px solid red");

**.parents()**

Selecciona todos los ancestros (padres, abuelos, etc.) de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miElemento").parents().css("border", "1px solid blue");

**.parentsUntil()**

Selecciona todos los ancestros hasta que coincide con el selector especificado.

javascript

$("#miElemento").parentsUntil("#miContenedor").css("border", "1px solid green");

**.children()**

Selecciona todos los hijos directos de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miContenedor").children().css("border", "1px solid yellow");

**.find()**

Selecciona todos los elementos descendientes del elemento en el conjunto de elementos coincidentes.

javascript

$("#miContenedor").find(".miClase").css("border", "1px solid purple");

**.siblings()**

Selecciona todos los hermanos de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miElemento").siblings().css("border", "1px solid orange");

**.next()**

Selecciona el hermano inmediato siguiente de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miElemento").next().css("border", "1px solid pink");

**.nextAll()**

Selecciona todos los hermanos siguientes de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miElemento").nextAll().css("border", "1px solid brown");

**.nextUntil()**

Selecciona todos los hermanos siguientes hasta que coincide con el selector especificado.

javascript

$("#miElemento").nextUntil(".miClaseFinal").css("border", "1px solid cyan");

**.prev()**

Selecciona el hermano inmediato anterior de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miElemento").prev().css("border", "1px solid lime");

**.prevAll()**

Selecciona todos los hermanos anteriores de cada elemento en el conjunto de elementos coincidentes.

javascript

$("#miElemento").prevAll().css("border", "1px solid magenta");

**.prevUntil()**

Selecciona todos los hermanos anteriores hasta que coincide con el selector especificado.

javascript

$("#miElemento").prevUntil(".miClaseInicio").css("border", "1px solid navy");

**.first()**

Selecciona el primer elemento en el conjunto de elementos coincidentes.

javascript

$("div").first().css("border", "1px solid gold");

**.last()**

Selecciona el último elemento en el conjunto de elementos coincidentes.

javascript

$("div").last().css("border", "1px solid silver");

**.eq()**

Selecciona el elemento en el índice especificado dentro del conjunto de elementos coincidentes.

javascript

$("li").eq(2).css("border", "1px solid teal");

**.filter()**

Reduce el conjunto de elementos coincidentes a aquellos que coinciden con el selector o la función.

javascript

$("div").filter(".miClase").css("border", "1px solid coral");

**.not()**

Reduce el conjunto de elementos coincidentes eliminando aquellos que coinciden con el selector o la función.

**javascript**

$("div").not(".miClase").css("border", "1px solid olive");

**Ejemplos Combinados**

**javascript**

$(document).ready(function() {

// Seleccionar el padre del elemento

$("#miElemento").parent().css("border", "1px solid red");

// Seleccionar todos los ancestros del elemento

$("#miElemento").parents().css("border", "1px solid blue");

// Seleccionar todos los ancestros hasta un selector

$("#miElemento").parentsUntil("#miContenedor").css("border", "1px solid green");

// Seleccionar todos los hijos directos del contenedor

$("#miContenedor").children().css("border", "1px solid yellow");

// Seleccionar todos los descendientes con una clase específica

$("#miContenedor").find(".miClase").css("border", "1px solid purple");

// Seleccionar todos los hermanos del elemento

$("#miElemento").siblings().css("border", "1px solid orange");

// Seleccionar el siguiente hermano del elemento

$("#miElemento").next().css("border", "1px solid pink");

// Seleccionar todos los hermanos siguientes del elemento

$("#miElemento").nextAll().css("border", "1px solid brown");

// Seleccionar todos los hermanos siguientes hasta un selector específico

$("#miElemento").nextUntil(".miClaseFinal").css("border", "1px solid cyan");

// Seleccionar el anterior hermano del elemento

$("#miElemento").prev().css("border", "1px solid lime");

// Seleccionar todos los hermanos anteriores del elemento

$("#miElemento").prevAll().css("border", "1px solid magenta");

// Seleccionar todos los hermanos anteriores hasta un selector específico

$("#miElemento").prevUntil(".miClaseInicio").css("border", "1px solid navy");

// Seleccionar el primer div

$("div").first().css("border", "1px solid gold");

// Seleccionar el último div

$("div").last().css("border", "1px solid silver");

// Seleccionar el tercer elemento en una lista

$("li").eq(2).css("border", "1px solid teal");

// Seleccionar todos los divs con una clase específica

$("div").filter(".miClase").css("border", "1px solid coral");

// Seleccionar todos los divs que no tienen una clase específica

$("div").not(".miClase").css("border", "1px solid olive");

});

**Ajax en jQuery**

**.load()**

El método .load() se utiliza para cargar datos de un servidor y colocar el contenido devuelto dentro del elemento seleccionado.

**javascript**

// Cargar contenido en un elemento div

$("#miDiv").load("archivo.html");

// Cargar contenido y manejar la respuesta

$("#miDiv").load("archivo.html", function(response, status, xhr) {

if (status == "success") {

console.log("Contenido cargado exitosamente");

}

if (status == "error") {

console.log("Error: " + xhr.status + " " + xhr.statusText);

}

});

**$.get()**

El método $.get() realiza una solicitud GET de Ajax a una URL y puede manejar la respuesta.

**javascript**

// Realizar una solicitud GET simple

$.get("datos.json", function(data) {

console.log(data);

});

// Realizar una solicitud GET con parámetros y manejar la respuesta

$.get("datos.php", { nombre: "John", edad: 30 }, function(data) {

console.log(data);

});

**$.post()**

El método $.post() realiza una solicitud POST de Ajax a una URL y puede manejar la respuesta.

**javascript**

// Realizar una solicitud POST simple

$.post("guardar.php", { nombre: "John", edad: 30 }, function(data) {

console.log(data);

});

// Realizar una solicitud POST con manejo de errores

$.post("guardar.php", { nombre: "John", edad: 30 })

.done(function(data) {

console.log("Datos guardados: " + data);

})

.fail(function() {

console.log("Error en la solicitud");

});

**Ejemplos Combinados**

**Cargar contenido con .load()**

**javascript**

$(document).ready(function() {

// Cargar contenido de archivo.html en el elemento con id miDiv

$("#miDiv").load("archivo.html", function(response, status, xhr) {

if (status == "success") {

console.log("Contenido cargado exitosamente");

}

if (status == "error") {

console.log("Error: " + xhr.status + " " + xhr.statusText);

}

});

});

**Obtener datos con $.get()**

**javascript**

$(document).ready(function() {

// Obtener datos de datos.json

$.get("datos.json", function(data) {

console.log(data);

});

// Obtener datos de datos.php con parámetros

$.get("datos.php", { nombre: "John", edad: 30 }, function(data) {

console.log(data);

});

});

**Enviar datos con $.post()**

**javascript**

$(document).ready(function() {

// Enviar datos a guardar.php

$.post("guardar.php", { nombre: "John", edad: 30 }, function(data) {

console.log("Datos guardados: " + data);

});

// Enviar datos a guardar.php con manejo de errores

$.post("guardar.php", { nombre: "John", edad: 30 })

.done(function(data) {

console.log("Datos guardados: " + data);

})

.fail(function() {

console.log("Error en la solicitud");

});

});

**Resumen**

* **.load()**: Carga contenido HTML en un elemento desde una URL especificada.
* **$.get()**: Realiza una solicitud HTTP GET a una URL y maneja la respuesta.
* **$.post()**: Realiza una solicitud HTTP POST a una URL y maneja la respuesta.

**JSON en JavaScript**

**Sintaxis JSON**

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos. La sintaxis JSON es similar a la de los objetos de JavaScript, con algunas diferencias.

**Ejemplo de un objeto JSON:**

**json**

{

"nombre": "Juan",

"edad": 30,

"ciudad": "Madrid",

"hobbies": ["leer", "viajar", "deportes"]

}

**Manejo de JSON en HTTP**

Puedes enviar y recibir datos JSON utilizando solicitudes HTTP con fetch, XMLHttpRequest, o librerías como jQuery.

**Ejemplo con fetch:**

**javascript**

// Realizar una solicitud GET y recibir datos JSON

fetch('https://api.ejemplo.com/datos')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

// Realizar una solicitud POST enviando datos JSON

fetch('https://api.ejemplo.com/guardar', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ nombre: "Juan", edad: 30 })

})

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

**Manejo de Archivos JSON**

Puedes cargar archivos JSON localmente o desde un servidor.

**Ejemplo con jQuery:**

**javascript**

$.getJSON('datos.json', function(data) {

console.log(data);

});

**Integración con SQL**

Para trabajar con datos JSON en bases de datos SQL, puedes convertir JSON a SQL o usar funcionalidades específicas del motor de base de datos.

**Ejemplo en PostgreSQL:**

**sql**

-- Insertar datos JSON en una tabla

INSERT INTO mi\_tabla (datos)

VALUES ('{"nombre": "Juan", "edad": 30}');

-- Seleccionar datos JSON

SELECT datos->>'nombre' AS nombre

FROM mi\_tabla

WHERE datos->>'ciudad' = 'Madrid';

**Métodos en jQuery**

**getJSON()**

Este método realiza una solicitud HTTP GET y espera una respuesta JSON, que se analiza automáticamente y se pasa a una función de callback.

**javascript**

$.getJSON('https://api.ejemplo.com/datos', function(data) {

console.log(data);

});

**parseJSON()**

Este método analiza una cadena JSON y devuelve el objeto JavaScript correspondiente.

**javascript**

// Ejemplo de cadena JSON

let jsonString = '{"nombre": "Juan", "edad": 30}';

// Analizar la cadena JSON

let obj = $.parseJSON(jsonString);

console.log(obj.nombre); // Salida: Juan

**Ejemplos Combinados**

**Usando getJSON() para obtener datos**

**javascript**

$(document).ready(function() {

$.getJSON('https://api.ejemplo.com/datos', function(data) {

console.log(data);

// Manipular los datos recibidos

$('#nombre').text(data.nombre);

$('#edad').text(data.edad);

});

});

**Analizando JSON con parseJSON()**

**javascript**

$(document).ready(function() {

let jsonString = '{"nombre": "Juan", "edad": 30}';

let obj = $.parseJSON(jsonString);

console.log(obj.nombre); // Salida: Juan

console.log(obj.edad); // Salida: 30

// Usar los datos en la página

$('#nombre').text(obj.nombre);

$('#edad').text(obj.edad);

});

**Resumen**

* **Sintaxis JSON**: JSON utiliza una sintaxis basada en texto que es similar a la de los objetos de JavaScript.
* **HTTP**: Puedes enviar y recibir datos JSON utilizando solicitudes HTTP.
* **Archivos**: Los datos JSON pueden ser cargados desde archivos locales o remotos.
* **SQL**: Puedes almacenar y manipular datos JSON en bases de datos SQL con soporte adecuado.
* **getJSON()**: Realiza una solicitud HTTP GET y recibe una respuesta JSON.
* **parseJSON()**: Analiza una cadena JSON y devuelve un objeto JavaScript.

**Interacciones en jQuery UI**

**Draggable**

Permite hacer que un elemento sea arrastrable.

**html**

<div id="arrastrable" style="width: 100px; height: 100px; background: #f00;"></div>

<script>

$(function() {

$("#arrastrable").draggable();

});

</script>

**Droppable**

Permite especificar un área donde los elementos arrastrables pueden ser soltados.

**html**

<div id="arrastrable" style="width: 100px; height: 100px; background: #f00;"></div>

<div id="soltar" style="width: 150px; height: 150px; background: #0f0; margin-top: 20px;"></div>

<script>

$(function() {

$("#arrastrable").draggable();

$("#soltar").droppable({

drop: function(event, ui) {

$(this).css("background", "#00f");

}

});

});

</script>

**Resizable**

Permite cambiar el tamaño de un elemento.

**html**

<div id="redimensionable" style="width: 150px; height: 150px; background: #0f0;"></div>

<script>

$(function() {

$("#redimensionable").resizable();

});

</script>

**Selectable**

Permite seleccionar elementos de una lista usando el ratón.

**html**

<ol id="seleccionable">

<li>Elemento 1</li>

<li>Elemento 2</li>

<li>Elemento 3</li>

<li>Elemento 4</li>

</ol>

<script>

$(function() {

$("#seleccionable").selectable();

});

</script>

**Sortable**

Permite reordenar elementos arrastrándolos.

**html**

<ul id="ordenable">

<li>Elemento 1</li>

<li>Elemento 2</li>

<li>Elemento 3</li>

<li>Elemento 4</li>

</ul>

<script>

$(function() {

$("#ordenable").sortable();

});

</script>

**Ejemplos Detallados**

**Draggable con Opciones**

Puedes personalizar el comportamiento del elemento arrastrable con opciones adicionales.

**javascript**

$(function() {

$("#arrastrable").draggable({

axis: "x", // Restringe el movimiento al eje X

containment: "#contenedor", // Restringe el movimiento dentro de un contenedor

grid: [20, 20], // Hace que el elemento se mueva en incrementos de 20px

handle: "#handle" // Especifica un área específica para arrastrar el elemento

});

});

**Droppable con Eventos**

Puedes manejar varios eventos cuando un elemento es soltado.

**javascript**

$(function() {

$("#arrastrable").draggable();

$("#soltar").droppable({

drop: function(event, ui) {

$(this).addClass("ui-state-highlight").find("p").html("Soltado!");

},

over: function(event, ui) {

$(this).css("background", "#f90");

},

out: function(event, ui) {

$(this).css("background", "#0f0");

}

});

});

**Resizable con Restricciones**

Puedes restringir el tamaño mínimo y máximo de redimensionamiento.

**javascript**

$(function() {

$("#redimensionable").resizable({

maxHeight: 300,

maxWidth: 300,

minHeight: 150,

minWidth: 150

});

});

**Selectable con Funciones de Callback**

Puedes manejar eventos cuando se seleccionan elementos.

**javascript**

$(function() {

$("#seleccionable").selectable({

stop: function() {

var result = $("#select-result").empty();

$(".ui-selected", this).each(function() {

var index = $("#seleccionable li").index(this);

result.append(" #" + (index + 1));

});

}

});

});

**Sortable con Conexión a Listas**

Puedes conectar varias listas para permitir la transferencia de elementos entre ellas.

**javascript**

$(function() {

$("#ordenable, #otraLista").sortable({

connectWith: ".conectado"

}).disableSelection();

});

**Resumen**

* **Draggable**: Hace que los elementos sean arrastrables.
* **Droppable**: Define áreas donde los elementos arrastrables pueden ser soltados.
* **Resizable**: Permite redimensionar elementos.
* **Selectable**: Permite seleccionar elementos de una lista.
* **Sortable**: Permite reordenar elementos arrastrándolos.

**Widgets de jQuery UI - Parte 1**

**Accordion**

Crea un menú desplegable con múltiples paneles colapsables.

**html**

<div id="acordeon">

<h3>Sección 1</h3>

<div>

<p>Contenido de la sección 1.</p>

</div>

<h3>Sección 2</h3>

<div>

<p>Contenido de la sección 2.</p>

</div>

<h3>Sección 3</h3>

<div>

<p>Contenido de la sección 3.</p>

</div>

</div>

<script>

$(function() {

$("#acordeon").accordion();

});

</script>

**Autocomplete**

Proporciona sugerencias mientras el usuario escribe en un campo de entrada.

**html**

<label for="tags">Etiqueta:</label>

<input id="tags">

<script>

$(function() {

var etiquetas = ["Manzana", "Banana", "Cereza", "Damasco"];

$("#tags").autocomplete({

source: etiquetas

});

});

</script>

**Button**

Transforma un elemento HTML en un botón con opciones adicionales.

**html**

<button id="boton">Haz clic</button>

<script>

$(function() {

$("#boton").button();

});

</script>

**Checkboxradio**

Permite la personalización de los estilos de los checkboxes y radio buttons.

**html**

<fieldset>

<legend>Selecciona una opción:</legend>

<label for="radio1">Opción 1</label>

<input type="radio" id="radio1" name="radio">

<label for="radio2">Opción 2</label>

<input type="radio" id="radio2" name="radio">

</fieldset>

<script>

$(function() {

$("input[type='radio']").checkboxradio();

});

</script>

**Controlgroup**

Agrupa varios widgets como botones en un único control visual.

**html**

<div id="grupo">

<button>Hola</button>

<button>Adiós</button>

<button>Ahora</button>

</div>

<script>

$(function() {

$("#grupo").controlgroup();

});

</script>

**Datepicker**

Permite seleccionar fechas de un calendario interactivo.

**html**

<label for="fecha">Fecha:</label>

<input type="text" id="fecha">

<script>

$(function() {

$("#fecha").datepicker();

});

</script>

**Dialog**

Crea un cuadro de diálogo modal.

**html**

<div id="dialogo" title="Diálogo simple">

<p>Este es un cuadro de diálogo simple.</p>

</div>

<script>

$(function() {

$("#dialogo").dialog();

});

</script>

**Resumen**

* **Accordion**: Crea un menú desplegable con paneles colapsables.
* **Autocomplete**: Proporciona sugerencias mientras el usuario escribe en un campo de entrada.
* **Button**: Transforma elementos HTML en botones con opciones adicionales.
* **Checkboxradio**: Personaliza los estilos de los checkboxes y radio buttons.
* **Controlgroup**: Agrupa varios widgets como botones en un único control visual.
* **Datepicker**: Permite seleccionar fechas de un calendario interactivo.
* **Dialog**: Crea cuadros de diálogo modales.

**Widgets de jQuery UI - Parte 2**

**Menu**

Crea un menú desplegable con opciones.

**html**

<ul id="menu">

<li>Elemento 1

<ul>

<li>Subelemento 1</li>

<li>Subelemento 2</li>

</ul>

</li>

<li>Elemento 2</li>

<li>Elemento 3</li>

</ul>

<script>

$(function() {

$("#menu").menu();

});

</script>

**Progressbar**

Muestra el progreso de una tarea en forma visual.

**html**

<div id="barraProgreso"></div>

<script>

$(function() {

$("#barraProgreso").progressbar({

value: 50 // Valor inicial de la barra de progreso

});

});

</script>

**Selectmenu**

Transforma un elemento <select> en un menú desplegable personalizable.

**html**

<select id="selector">

<option value="opcion1">Opción 1</option>

<option value="opcion2">Opción 2</option>

<option value="opcion3">Opción 3</option>

</select>

<script>

$(function() {

$("#selector").selectmenu();

});

</script>

**Slider**

Permite seleccionar valores de un rango deslizando un control deslizante.

**html**

<div id="controlDeslizante"></div>

<script>

$(function() {

$("#controlDeslizante").slider({

min: 0, // Valor mínimo

max: 100, // Valor máximo

value: 50 // Valor inicial

});

});

</script>

**Spinner**

Crea un control de entrada numérica con botones para incrementar y decrementar el valor.

**html**

<label for="spinner">Cantidad:</label>

<input id="spinner" name="valor">

<script>

$(function() {

$("#spinner").spinner();

});

</script>

**Tabs**

Crea un conjunto de pestañas que muestran diferentes contenidos.

**html**

<div id="pestanas">

<ul>

<li><a href="#pestaña-1">Pestaña 1</a></li>

<li><a href="#pestaña-2">Pestaña 2</a></li>

<li><a href="#pestaña-3">Pestaña 3</a></li>

</ul>

<div id="pestaña-1">

Contenido de la pestaña 1.

</div>

<div id="pestaña-2">

Contenido de la pestaña 2.

</div>

<div id="pestaña-3">

Contenido de la pestaña 3.

</div>

</div>

<script>

$(function() {

$("#pestanas").tabs();

});

</script>

**Tooltip**

Crea un mensaje emergente informativo al pasar el cursor sobre un elemento.

**html**

<div title="Mensaje emergente">Pasa el cursor aquí</div>

<script>

$(function() {

$(document).tooltip();

});

</script>

**Resumen**

* **Menu**: Crea menús desplegables con opciones.
* **Progressbar**: Muestra el progreso de una tarea en forma visual.
* **Selectmenu**: Transforma elementos <select> en menús desplegables personalizables.
* **Slider**: Permite seleccionar valores de un rango deslizando un control deslizante.
* **Spinner**: Crea controles de entrada numérica con botones de incremento/decremento.
* **Tabs**: Crea un conjunto de pestañas que muestran diferentes contenidos.
* **Tooltip**: Crea mensajes emergentes informativos al pasar el cursor sobre un elemento.

**Efectos en jQuery UI**

**addClass**

Agrega una o más clases a los elementos seleccionados con una animación opcional.

**javascript**

$("#miElemento").addClass("claseNueva", 1000);

**Color Animation**

Permite animar cambios de color en los elementos seleccionados.

**javascript**

$("#miElemento").animate({ backgroundColor: "blue" }, 1000);

**Easing**

Especifica el tipo de interpolación que se utilizará para las animaciones.

**javascript**

$("#miElemento").animate({ left: "200px" }, 1000, "easeOutBounce");

**Effect**

Ejecuta un efecto de animación predefinido en los elementos seleccionados.

**javascript**

$("#miElemento").effect("bounce", { times: 3 }, 1000);

**Hide**

Oculta los elementos seleccionados con una animación opcional.

**javascript**

$("#miElemento").hide("slow");

**removeClass**

Elimina una o más clases de los elementos seleccionados con una animación opcional.

**javascript**

$("#miElemento").removeClass("claseExistente", 1000);

**Show**

Muestra los elementos seleccionados con una animación opcional.

**javascript**

$("#miElemento").show("fast");

**Switch Class**

Intercambia una o más clases de los elementos seleccionados con una animación opcional.

javascript

Copy code

$("#miElemento").switchClass("claseAnterior", "claseNueva", 1000);

**Toggle**

Alterna entre mostrar u ocultar los elementos seleccionados con una animación opcional.

**javascript**

$("#miElemento").toggle("slow");

**toggleClass**

Agrega o quita una clase de los elementos seleccionados con una animación opcional.

javascript

Copy code

$("#miElemento").toggleClass("claseToggle", 1000);

**Ejemplos Combinados**

**Aplicar addClass y luego removeClass**

**javascript**

$("#miElemento").addClass("destacado", 1000).delay(2000).removeClass("destacado", 1000);

**Alternar entre mostrar u ocultar con toggle**

**javascript**

$("#miElemento").click(function() {

$(this).toggle("slow");

});

**Aplicar switchClass en respuesta a un evento**

**javascript**

$("#boton").click(function() {

$("#miElemento").switchClass("claseAnterior", "claseNueva", 1000);

});

**Resumen**

* **addClass**: Agrega una clase a los elementos seleccionados con animación opcional.
* **Color Animation**: Permite animar cambios de color en los elementos seleccionados.
* **Easing**: Especifica el tipo de interpolación para las animaciones.
* **Effect**: Ejecuta un efecto de animación predefinido en los elementos seleccionados.
* **Hide**: Oculta los elementos seleccionados con animación opcional.
* **removeClass**: Elimina una clase de los elementos seleccionados con animación opcional.
* **Show**: Muestra los elementos seleccionados con animación opcional.
* **Switch Class**: Intercambia una clase de los elementos seleccionados con animación opcional.
* **Toggle**: Alterna entre mostrar u ocultar los elementos seleccionados con animación opcional.
* **toggleClass**: Agrega o quita una clase de los elementos seleccionados con animación opcional.

**Temas y Utilidades en jQuery UI**

**Position**

El método position() permite posicionar elementos relativos a otros elementos o al documento.

**javascript**

$("#miElemento").position({

my: "left top",

at: "right bottom",

of: "#otroElemento"

});

**Widget Factory**

La fábrica de widgets (Widget Factory) es una infraestructura que permite la creación de widgets complejos en jQuery UI. Permite definir nuevos widgets y extender los existentes.

**javascript**

$.widget("ui.miWidget", {

options: {

propiedad: valor

},

\_create: function() {

// Lógica de creación del widget

},

\_init: function() {

// Lógica de inicialización del widget

},

\_destroy: function() {

// Lógica de destrucción del widget

},

métodoPersonalizado: function() {

// Método personalizado del widget

}

});

**Ejemplo de Uso**

**Crear un nuevo widget**

Supongamos que queremos crear un widget de barra de progreso personalizado utilizando la Widget Factory.

**javascript**

$.widget("ui.miBarraProgreso", {

options: {

valor: 0,

color: "green"

},

\_create: function() {

this.element.addClass("mi-barra-progreso");

this.actualizar();

},

actualizar: function() {

var porcentaje = this.options.valor + "%";

this.element.css("width", porcentaje).css("background-color", this.options.color);

},

\_setOption: function(key, value) {

this.\_super(key, value);

if (key === "valor") {

this.actualizar();

}

}

});

// Uso del widget

$("#miBarra").miBarraProgreso({

valor: 50,

color: "blue"

});

En este ejemplo, creamos un nuevo widget llamado miBarraProgreso. Este widget puede inicializarse con un valor y un color, y luego se actualizará automáticamente para reflejar cualquier cambio en el valor.

**Resumen**

* **Position**: Permite posicionar elementos relativos a otros elementos o al documento.
* **Widget Factory**: Es una infraestructura que permite la creación de widgets complejos en jQuery UI, facilitando la definición, inicialización, destrucción y personalización de los mismos.