JS

JavaScript 2.0

▼ querySelector()

1. Sintaxis de `querySelectorAll`

```
document.querySelectorAll("selector_css")
```

- **Selector CSS**: Se usa cualquier selector válido de CSS.
- **Devuelve**: Una **NodeList** con todos los elementos que coinciden con el selector.

---

2. Diferencia entre `querySelector` y `querySelectorAll`

Método	Descripción	Devuelve
`.querySelector()`	Selecciona el **primer** elemento que coincida con el selector.	Un **elemento único**
`.querySelectorAll()`	Selecciona **todos** los elementos que coincidan con el selector.	Una **NodeList**

✔ **Ejemplo: Seleccionar el primer y todos los elementos con clase `"item"`:**

```
document.querySelector(".item"); // Solo el primer elemento
document.querySelectorAll(".item"); // Todos los elementos con clase "it
em"
```

---

3. Recorrer una NodeList

3.1. Usando `forEach()`

```
document.querySelectorAll("p").forEach(function(parrafo) {
    console.log(parrafo.textContent);
```

```
});
```

3.2. Usando `for...of`

```
for (let parrafo of document.querySelectorAll("p")) {
    console.log(parrafo.textContent);
}
```

3.3. Convertir a un array y usar `map()`

```
let elementos = Array.from(document.querySelectorAll("p"));
elementos.map(el => console.log(el.textContent));
```

---

4. Selectores CSS Compatibles con `querySelectorAll`

4.1. Seleccionar por etiqueta

```
document.querySelectorAll("p"); // Todos los párrafos
```

4.2. Seleccionar por clase

```
document.querySelectorAll(".miClase"); // Todos los elementos con la cl
ase "miClase"
```

4.3. Seleccionar por id

```
document.querySelectorAll("#mild"); // Elemento con id "mild" (solo hab
rá uno)
```

4.4. Seleccionar elementos anidados

```
document.querySelectorAll("div p"); // Todos los <p> dentro de <div>
```

4.5. Seleccionar hijos directos

```
document.querySelectorAll("div > p"); // Solo los <p> hijos directos de <div>
```

4.6. Seleccionar elementos adyacentes

```
document.querySelectorAll("h1 + p"); // Primer <p> después de un <h1>
document.querySelectorAll("h1 ~ p"); // Todos los <p> después de un <h1>
```

---

5. Selectores Avanzados

5.1. Seleccionar elementos con atributos específicos

```
document.querySelectorAll("input[type='checkbox']"); // Todos los checkboxes
document.querySelectorAll("a[target='_blank']"); // Todos los enlaces que abren en nueva ventana
```

5.2. Seleccionar elementos por parte del atributo

```
document.querySelectorAll("[id^='btn']"); // Elementos cuyo id empieza con "btn"
document.querySelectorAll("[id$='end']"); // Elementos cuyo id termina con "end"
document.querySelectorAll("[id*='parte']"); // Elementos cuyo id contiene "parte"
```

5.3. Seleccionar elementos por estado

```
document.querySelectorAll(":checked"); // Todos los inputs marcados (checkbox/radio)
document.querySelectorAll(":disabled"); // Todos los inputs deshabilitad
```

```
os
document.querySelectorAll(":required"); // Todos los inputs obligatorios
```

6. Selección por Pseudo-Clases

Selector	Descripción
`:first-child`	Primer hijo de un elemento
`:last-child`	Último hijo de un elemento
`:nth-child(n)`	Elemento número `n` hijo
`:nth-of-type(n)`	Elemento número `n` de su tipo
`:not(selector)`	Elementos que **no** coincidan con el selector

Ejemplo: Seleccionar el primer y último párrafo

```
document.querySelectorAll("p:first-child");
document.querySelectorAll("p:last-child");
```

Ejemplo: Seleccionar los elementos impares

```
document.querySelectorAll("li:nth-child(odd)");
```

7. Modificar Elementos con `querySelectorAll`

7.1. Cambiar el color de todos los párrafos

```
document.querySelectorAll("p").forEach(p ⇒ p.style.color = "blue");
```

7.2. Cambiar el texto de los elementos con clase `"rojo"`

```
document.querySelectorAll(".rojo").forEach(el ⇒ el.textContent = el.text
Content.toUpperCase());
```

7.3. Agregar una clase a todos los elementos con clase `"verde"`

```
document.querySelectorAll(".verde").forEach(el => el.classList.add("fondogris"));
```

7.4. Cambiar el color del primer `<h1>` a verde

```
document.querySelector("h1").style.color = "green";
```

▼ Cookies

1. Creación y Uso de Cookies

1.1. Formato de una Cookie

```
nombre=valor; expires=fecha; max-age=segundos; path=ruta; domain=dominio; secure; httponly;
```

1.2. Ejemplo de cabecera HTTP de respuesta que crea cookies

```
Set-Cookie: user=JohnDoe; Expires=Wed, 09 Jun 2025 10:18:14 GMT; Path=/;
```

1.3. Ejemplo de cabecera HTTP de solicitud con cookies

```
Cookie: user=JohnDoe; theme=dark;
```

2. Parámetros de una Cookie

Parámetro	Descripción
`nombre=valor`	Identifica la cookie y su contenido
`expires=fecha`	Fecha de expiración (UTC)
`max-age=segundos`	Tiempo de vida en segundos (prioridad sobre `expires`)
`path=/ruta`	Define qué rutas pueden acceder a la cookie

`domain=dominio.com`	Define qué subdominios pueden acceder a la cookie
`secure`	Solo se envía en conexiones HTTPS
`HttpOnly`	No accesible desde JavaScript (protección contra XSS)

Ejemplo de cookie con parámetros

```
document.cookie = "theme=dark; max-age=3600; path=/";
```

---

3. Trabajar con Cookies en JavaScript

3.1. Crear una Cookie

```
document.cookie = "usuario=Juan; max-age=86400; path=/";
```

- **Se almacena con una duración de 1 día (86400 segundos)**
- **Es accesible desde cualquier página del dominio (`path=/`)**

3.2. Leer Cookies

JavaScript **no** proporciona un método directo para leer cookies individuales, pero podemos extraerlas manualmente.

```
console.log(document.cookie);
```

- **Salida:** `"usuario=Juan; theme=dark"`

---

4. Modificar Cookies

Si una cookie ya existe, simplemente **reasignamos su valor**.

```
document.cookie = "usuario=Pedro";
```

- **Se reemplaza el valor de `usuario`**

---

5. Eliminar Cookies

Para eliminar una cookie, se debe establecer una **fecha de expiración pasada** o `max-age=0`:

```
document.cookie = "usuario=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";
```

O usando `max-age=0`:

```
document.cookie = "usuario=; max-age=0; path=/";
```

---

6. Leer el Valor de una Cookie en JavaScript

Dado que `document.cookie` devuelve un string con todas las cookies, debemos extraer el valor manualmente.

7.1. Método con `split()`

```
function getCookie(nombre) {
  let cookies = document.cookie.split("; ");
  for (let cookie of cookies) {
    let [key, value] = cookie.split("=");
    if (key === nombre) return decodeURIComponent(value);
  }
  return null;
}
```

- **Ejemplo de uso:**

```
console.log(getCookie("usuario")); // "Pedro"
```

7.2. Método con Expresiones Regulares

```
function getCookie(name) {
  return decodeURIComponent(document.cookie.replace(
    new RegExp("(?:(?:^|.*;)\\s*" + name.replace(/[\-\.\+\*]/g, "\\$&") +
    "\\s*\\=\\s*([^;]*).*$)|^.*$"), "$1")) || null;
}
```

▼ Formularios

Apuntes de JavaScript: Formularios

1. Introducción

- JavaScript nació para validar formularios sin recargar la página.
- Aunque ahora AJAX es más usado, el manejo de formularios sigue siendo esencial.

2. Acceso a formularios y elementos

Métodos de acceso

- **Por índice (desaconsejado)**: `document.forms[0]`
- **Por nombre (recomendado)**: `document.formulario`
- **Por ID (recomendado)**:

```
var formulario = document.getElementById("formulario");
var input = document.getElementById("elemento");
```

3. Propiedades de los elementos de formulario

Propiedad	Descripción
`type`	Tipo de input (`text` , `checkbox` , `select-one` , etc.)
`form`	Referencia al formulario padre
`name`	Nombre del elemento (solo lectura)
`value`	Valor actual del campo (lectura/escritura)

4. Eventos comunes

Evento	Descripción
`onclick`	Se activa al hacer clic
`onchange`	Se activa al cambiar el valor

`onfocus`	Se activa al entrar en un campo
`onblur`	Se activa al salir de un campo

5. Métodos esenciales

Obtener valor de un input

```
var valor = document.getElementById("texto").value;
```

Obtener opción seleccionada en un `<select>`

```
var lista = document.getElementById("opciones");
var valorSeleccionado = lista.options[lista.selectedIndex].value;
```

Obtener radiobutton seleccionado

```
var radios = document.getElementsByName("pregunta");
for (var i = 0; i < radios.length; i++) {
  if (radios[i].checked) {
    console.log("Seleccionado:", radios[i].value);
  }
}
```

Comprobar si un checkbox está marcado

```
var checkbox = document.getElementById("condiciones");
console.log("Seleccionado:", checkbox.checked);
```

6. Control de formularios

Establecer el foco automáticamente

```
document.getElementById("primero").focus();
```

Evitar envío duplicado

```
<input type="button" value="Enviar" onclick="this.disabled=true; this.value='Enviando...'; this.form.submit();" />
```

Limitar caracteres en un `<textarea>`

```
function limita(maximoCaracteres) {
   var elemento = document.getElementById("texto");
   if(elemento.value.length >= maximoCaracteres ) {
      return false;
   }
   else {
      return true;
   }
   }
<textarea id="texto" onkeypress="return limita(100);"></textarea>
```

```
<textarea id="texto" onkeypress="return limita(100);"></textarea>
```

Restringir caracteres en un campo

```
function permite(evento, permitidos) {
  var numeros = "0123456789";
  var letras = " abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
  var todo = numeros + letras;
  var teclasEspeciales = [8, 37, 39, 46];

  var permitido = permitidos == 'num' ? numeros :
                  permitidos == 'car' ? letras : todo;

  var evento = evento || window.event;
  var codigo = evento.charCode || evento.keyCode;
  var caracter = String.fromCharCode(codigo);

  return permitido.indexOf(caracter) != -1 || teclasEspeciales.includes(co
```

```
digo);
}
```

```
<input type="text" onkeypress="return permite(event, 'num')"> <!-- Sol
o números -->
<input type="text" onkeypress="return permite(event, 'car')"> <!-- Solo
letras -->
<input type="text" onkeypress="return permite(event, 'num_car')"> <!--
Números y letras -->
```

---

7. Validaciones

Validar campo obligatorio

```
valor = document.getElementById("campo").value;
if( valor == null || valor.length == 0 || /^\s+$/.test(valor) ) {
   return false;
}
```

Validar número

```
valor = document.getElementById("campo").value;
if( isNaN(valor) ) {
   return false;
}
```

Validar selección en `<select>`

```
indice = document.getElementById("opciones").selectedIndex;
if( indice == null || indice == 0 ) {
   return false;
}
```

Validar email

```
valor = document.getElementById("campo").value;
if( !(/\w{1,}[@][\w\-]{1,}([.]([\w\-]{1,})){1,3}$/.test(valor)) ) {
   return false;
}
```

Validar fecha

```
var ano = document.getElementById("ano").value;
var mes = document.getElementById("mes").value;
var dia = document.getElementById("dia").value;

valor = new Date(ano, mes, dia);
if( !isNaN(valor) ) {
   return false;
}
```

Validar DNI

```
valor = document.getElementById("campo").value;
var letras = ['T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B', 'N', 'J',
'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E', 'T'];
if( !(/^\d{8}[A-Z]$/.test(valor)) ) {
   return false;
}
if(valor.charAt(8) != letras[(valor.substring(0, 8))%23]) {
   return false;
}
```

Validar teléfono (9 dígitos)

```
valor = document.getElementById("campo").value;
if( !(/^\d{9}$/.test(valor)) ) {
   return false;
}
```

Validar checkbox seleccionado

```
// comprovar solo un checkbox
elemento = document.getElementById("campo");
if( !elemento.checked ) {
    return false;
}

// comprovar si hay varios checbox
formulario = document.getElementById("formulario");
for(var i=0; i<formulario.elements.length; i++) {
    var elemento = formulario.elements[i];
    if(elemento.type == "checkbox") {
        if(!elemento.checked) {
            return false;
        }
    }
}
```

Validar grupo de radiobutton

```
opciones = document.getElementsByName("opciones");
var seleccionado = false;
for(var i=0; i<opciones.length; i++) {
    if(opciones[i].checked) {
        seleccionado = true;
        break;
    }
}
if(!seleccionado) {
    return false;
}
```

▼ Expresiones regulares

1. Creación de Expresiones Regulares

Dos formas principales:

```
// Usando constructor RegExp (útil para patrones dinámicos)
var regex1 = new RegExp("pattern", "flags");

// Usando sintaxis literal (más eficiente y recomendada)
var regex2 = /pattern/flags;
```

Flags (modificadores):

Flag	Descripción
`g`	Búsqueda global (no se detiene en la primera coincidencia)
`i`	Ignora mayúsculas y minúsculas
`m`	Búsqueda en varias líneas (afecta `^` y `$`)

2. Metacaracteres más usados

Metacaracter	Descripción
`^`	Inicio de la cadena
`$`	Fin de la cadena
`.`	Cualquier carácter excepto salto de línea
`*`	0 o más repeticiones del patrón previo
`+`	1 o más repeticiones del patrón previo
`?`	0 o 1 repetición del patrón previo
`{x,y}`	Entre `x` y `y` repeticiones
`|`	Sirve para alternar expresiones.
`\`	Escapa caracteres especiales (`\.` busca un punto literal)

Ejemplo

```
/^Hola/.test("Hola mundo");  // true
/mundo$/.test("Hola mundo"); // true
/\d+/.test("Tengo 100 manzanas"); // true (busca números)
```

3. Conjuntos y Rangos

Expresión	Descripción
`[abc]`	Coincide con `a`, `b` o `c`
`[a-z]`	Cualquier letra minúscula
`[A-Z]`	Cualquier letra mayúscula
`[0-9]`	Cualquier dígito
`[^abc]`	No contiene `a`, `b` o `c`

Ejemplo

```
/[aeiou]/.test("casa");   // true (contiene vocales)
/[^0-9]/.test("123abc");  // true (contiene algo que no es un número)
```

4. Métodos del Objeto `RegExp`

Método	Descripción
`test()`	Devuelve `true` o `false` si hay coincidencia
`exec()`	Devuelve la primera coincidencia o `null`

Ejemplo

```
var regex = /\d+/;
console.log(regex.test("Hay 10 gatos"));  // true
console.log(regex.exec("Hay 10 gatos"));  // ["10"]
```

5. Métodos del Objeto `String` con Expresiones Regulares

Método	Descripción
`search()`	Devuelve la posición de la primera coincidencia
`match()`	Devuelve un array con todas las coincidencias
`replace()`	Reemplaza coincidencias en la cadena
`split()`	Divide la cadena en un array

Ejemplo

```
var texto = "Hola 123 mundo 456";

// Buscar posición del primer número
console.log(texto.search(/\d+/));  // 5

// Obtener todos los números
console.log(texto.match(/\d+/g));  // ["123", "456"]

// Reemplazar números por "X"
console.log(texto.replace(/\d+/g, "X"));  // "Hola X mundo X"

// Dividir cadena en palabras
console.log(texto.split(/\s+/));  // ["Hola", "123", "mundo", "456"]
```

---

6. Ejemplos de Validaciones con Expresiones Regulares

6.1. Validar Matrícula (formato: 4 números + 3 letras)

```
function validaMatricula(matricula) {
  var regex = /^[0-9]{4} [A-Z]{3}$/;
  return regex.test(matricula);
}
console.log(validaMatricula("1234 ABC")); // true
console.log(validaMatricula("12 ABCD"));  // false
```

6.2. Validar Fecha (Formato dd/mm/aa o dd/mm/aaaa)

```
function validaFecha(fecha) {
  var regex = /^(0?[1-9]|[12][0-9]|3[01])\/(0?[1-9]|1[012])\/[0-9]{2,4}$/;
  return regex.test(fecha);
}
console.log(validaFecha("05/07/2023")); // true
console.log(validaFecha("32/13/2023")); // false
```

6.3. Validar Email

```
function validaEmail(email) {
  var regex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
  return regex.test(email);
}
console.log(validaEmail("correo@dominio.com")); // true
console.log(validaEmail("correo@dominio"));     // false
```

6.4. Validar Número de Teléfono (Formato: 9 dígitos)

```
function validaTelefono(telefono) {
  return /^[0-9]{9}$/.test(telefono);
}
console.log(validaTelefono("612345678")); // true
console.log(validaTelefono("612-345-678")); // false
```

6.5. Validar DNI Español

```
function validaDNI(dni) {
  var letras = "TRWAGMYFPDXBNJZSQVHLCKE";
  var regex = /^\d{8}[A-Z]$/;

  if (!regex.test(dni)){
     return false;
  } else{
     return dni[8] === letras[dni.substring(0, 8) % 23];
  }

}

console.log(validaDNI("12345678Z")); // true
console.log(validaDNI("12345678A")); // false
```

▼ jQuery: Introducción y Selectores

1. Incluir jQuery en una Página Web

Usando un CDN

```
<script src="https://code.jquery.com/jquery-3.1.0.min.js"></script>
```

---

2. Conceptos Básicos

2.1. Alias `$`

- jQuery usa `$` como alias para acceder a sus métodos:

```
$(document).ready(function() {
  console.log("DOM cargado y listo");
});
```

- Si hay conflicto con otras librerías, usar:

```
$.noConflict();
jQuery(document).ready(function() { ... });
```

2.2. `$(document).ready()`

- Asegura que el código jQuery se ejecute solo cuando el **DOM esté cargado**:

```
$(document).ready(function() {
  alert("La página está lista");
});
```

- Versión simplificada:

```
$(function() {
  alert("La página está lista");
});
```

---

3. Selectores en jQuery

Los **selectores** permiten identificar elementos del DOM para manipularlos.

3.1. Tipos de Selectores

Selector	Descripción	Ejemplo
`*`	Selecciona **todos** los elementos	`$(" * ")`
`tag`	Selecciona todas las etiquetas	`$("p")`
`#id`	Selecciona por `id`	`$("#miElemento")`
`.clase`	Selecciona por `class`	`$(".miClase")`
`tag.clase`	Selecciona un tag con una clase específica	`$("p.destacado")`

Ejemplos

```
$("div").css("background-color", "yellow"); // Todos los <div>
$("#titulo").hide(); // Elemento con id "titulo"
$(".importante").css("color", "red"); // Elementos con clase "importante"
```

4. Selectores Avanzados

4.1. Selectores de Relaciones

Selector	Descripción	Ejemplo
`padre > hijo`	Elementos **hijos directos**	`$("div > p")`
`ancestro hijo`	Todos los **descendientes**	`$("ul li")`
`hermano + siguiente`	Hermano **inmediato**	`$("h1 + p")`
`hermano ~ todos`	Todos los **hermanos siguientes**	`$("h1 ~ p")`

Ejemplo

```
$("div > p").css("color", "blue");  // Párrafos dentro de un div
$("h1 + p").hide();  // Oculta el <p> inmediatamente después de un <h1>
```

5. Filtros en jQuery

Los filtros refinan la selección de elementos.

Filtro	Descripción	Ejemplo
`:first`	Primer elemento	`$("p:first")`
`:last`	Último elemento	`$("p:last")`
`:eq(n)`	Elemento en la posición `n`	`$("li:eq(2)")`
`:odd`	Elementos impares	`$("tr:odd")`
`:even`	Elementos pares	`$("tr:even")`
`:hidden`	Elementos ocultos	`$("div:hidden")`
`:visible`	Elementos visibles	`$("span:visible")`

Ejemplo: Filas tipo "cebra" en una tabla

```
$("tr:even").css("background", "#f2f2f2");
$("tr:odd").css("background", "#ffffff");
```

6. Selectores de Atributos

Permiten seleccionar elementos en función de atributos.

Selector	Descripción	Ejemplo
`[atributo]`	Tiene el atributo	`$("[disabled]")`
`[atributo=valor]`	Atributo con valor exacto	`$("[type='text']")`
`[atributo^=valor]`	Atributo comienza con `valor`	`$("[href^='http']")`
`[atributo$=valor]`	Atributo termina con `valor`	`$("[src$='.jpg']")`
`[atributo*=valor]`	Atributo contiene `valor`	`$("[title*='info']")`

Ejemplo: Quitar subrayado a enlaces de "Google"

```
$("a[href*='google']").css("text-decoration", "none");
```

7. Ejercicios Prácticos

Ejercicio 1: Cargar jQuery desde un CDN

```
<script src="https://code.jquery.com/jquery-3.1.0.min.js"></script>
```

Ejercicio 2: Usar `document.ready`

```
$(document).ready(function() {
    console.log("El DOM está listo");
});
```

Ejercicio 3: Aplicar Estilos con Selectores

```
$(".item").css("background-color", "#cecece");  // Fondo gris a los artíc
ulos
$("#cart_items").css("border", "4px solid black");  // Borde negro al carr
ito
$("img").css("border", "1px solid blue");  // Borde azul a todas las imáge
nes
$(".item > label").css("text-decoration", "underline"); // Subraya etiquet
as dentro de items
$("button", "#cart_container").css("color", "red"); // Texto rojo en boton
es del carrito
```

▼ jQuery: Modificación del DOM y Estilos

1. Encadenamiento (Chaining)

- jQuery permite ejecutar múltiples operaciones en una sola línea, mejorando la legibilidad y eficiencia.
- Ejemplo:

```
$("td:contains(Henry)") // Encuentra celdas con "Henry"
  .parent() // Selecciona el padre
  .addClass("highlight"); // Agrega clase "highlight"
```

2. Métodos de Recorrido del DOM (DOM Traversing)

Método	Descripción
`.eq(n)`	Selecciona el elemento en la posición `n`
`.first()` / `.last()`	Selecciona el primer/último elemento
`.parent()`	Selecciona el padre directo
`.parents()`	Selecciona todos los ancestros
`.children()`	Selecciona los hijos directos
`.siblings()`	Selecciona los hermanos
`.find('selector')`	Selecciona los descendientes que coincidan con el selector
`.next()` / `.prev()`	Selecciona el siguiente/anterior hermano

Ejemplo: Resaltar `span` dentro de `p`

```
$("p").find("span").css("color", "red");
```

3. Modificación del DOM

3.1. Manipulación de Elementos

Método	Descripción
`.html()`	Obtiene o cambia el contenido HTML de un elemento
`.text()`	Obtiene o cambia el texto (sin etiquetas)
`.append()`	Agrega contenido al final de un elemento
`.prepend()`	Agrega contenido al inicio de un elemento
`.before()`	Agrega contenido antes de un elemento
`.after()`	Agrega contenido después de un elemento
`.replaceWith()`	Reemplaza un elemento por otro
`.wrap()`	Envuelve el contenido dentro de otro elemento
`.unwrap()`	Elimina el envoltorio del elemento
`.clone()`	Duplica un elemento

`.remove()`	Elimina un elemento del DOM
`.empty()`	Elimina el contenido de un elemento sin eliminar el elemento en sí

Ejemplo: Cambiar contenido HTML

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

Ejemplo: Añadir elementos

```
$("#contenido").append("<p>Elemento añadido al final</p>");
$("#contenido").prepend("<p>Elemento añadido al inicio</p>");
```

---

4. Manipulación de Atributos

Método	Descripción
`.attr("atributo")`	Obtiene el valor de un atributo
`.attr("atributo", "valor")`	Cambia el valor de un atributo
`.removeAttr("atributo")`	Elimina un atributo
`.prop("propiedad")`	Obtiene una propiedad del elemento
`.prop("propiedad", "valor")`	Modifica una propiedad
`.val()`	Obtiene o cambia el valor de inputs, selects o textareas

Ejemplo: Cambiar el `alt` de una imagen

```
$("img").attr("alt", "Nueva descripción");
```

Ejemplo: Obtener el valor de un input

```
var valor = $("#campo").val();
console.log("Valor del input:", valor);
```

---

5. Manipulación de Estilos

Método	Descripción
`.css("propiedad")`	Obtiene el valor de una propiedad CSS
`.css("propiedad", "valor")`	Cambia el valor de una propiedad CSS
`.addClass("clase")`	Agrega una clase CSS
`.removeClass("clase")`	Elimina una clase CSS
`.toggleClass("clase")`	Alterna la clase (la agrega si no está, la elimina si está)
`.hasClass("clase")`	Verifica si un elemento tiene una clase aplicada

Ejemplo: Cambiar el color de fondo

```
$(".contenedor").css("background-color", "yellow");
```

Ejemplo: Alternar clase al hacer clic

```
$("#boton").click(function() {
    $(this).toggleClass("activo");
});
```

6. Introducción a Eventos en jQuery

Evento	Descripción
`.click()`	Se ejecuta al hacer clic en un elemento
`.dblclick()`	Se ejecuta al hacer doble clic
`.hover()`	Se ejecuta cuando el ratón entra o sale
`.focus()`	Se activa cuando un input obtiene el foco
`.blur()`	Se activa cuando un input pierde el foco
`.submit()`	Se ejecuta al enviar un formulario
`.change()`	Se ejecuta cuando cambia el valor de un input o select
`.keydown()`	Se ejecuta cuando se presiona una tecla
`.keyup()`	Se ejecuta cuando se suelta una tecla

Ejemplo: Contador con Click

```
$("a").click(function(event) {
    event.preventDefault(); // Evita la acción predeterminada del enlace
    var contador = parseInt($(this).text());
    contador++;
    $(this).text(contador);
});
```

▼ jQuery: Eventos y Manipulación del DOM

1. Recorrer Elementos con `.each()`

- `.each()` permite recorrer elementos seleccionados y aplicar cambios individualmente.

Ejemplo: Cambiar color de `div` a azul

```
$("div").each(function () {
  if ($(this).css("color") !== "blue") {
    $(this).css("color", "blue");
  }
});
```

2. Métodos para Manejo de Eventos

2.1. Métodos Básicos de Eventos

Método	Descripción
`.click()`	Se activa cuando se hace clic en un elemento
`.dblclick()`	Se activa al hacer doble clic
`.mouseover()` / `.mouseout()`	Detecta entrada/salida del ratón
`.focus()` / `.blur()`	Se activa cuando un input obtiene/pierde foco
`.keydown()` / `.keyup()`	Se ejecuta al presionar/soltar una tecla

Ejemplo: Click en un `div`

```
$("#miDiv").click(function() {
  $(this).css("background", "yellow");
});
```

---

3. Asignación de Eventos con `.on()` y `.off()` (Desde jQuery 1.7)

- `.on()` reemplaza `.bind()`, `.live()` y `.delegate()`, unificando la gestión de eventos.
- `.off()` reemplaza `.unbind()`, `.die()` y `.undelegate()`.

Ejemplo: Asignar evento con `.on()`

```
$("#boton").on("click", function() {
  alert("¡Botón clicado!");
});
```

Ejemplo: Remover evento con `.off()`

```
$("#boton").off("click");
```

Ejemplo: Delegación de eventos (para elementos creados dinámicamente)

```
$(document).on("click", ".boton-dinamico", function() {
  alert("Botón creado dinámicamente clicado");
});
```

---

4. Métodos Avanzados de Eventos

Método	Descripción
`.one()`	El evento solo se ejecuta **una vez**
`.trigger()`	Dispara un evento manualmente

`.preventDefault()`	Cancela la acción predeterminada de un evento
`.stopPropagation()`	Evita que el evento se propague a elementos padres

Ejemplo: Evento que solo se ejecuta una vez

```
$("#boton").one("click", function() {
  alert("Este mensaje solo aparecerá una vez");
});
```

Ejemplo: Prevenir navegación en un enlace

```
$("a").click(function(event) {
  event.preventDefault();
  alert("Este enlace no te llevará a ningún lado");
});
```

Ejemplo: Detener propagación de un evento

```
$("#hijo").click(function(event) {
  event.stopPropagation();
  alert("Evento capturado en el hijo, no en el padre");
});
```

---

5. Obtener Información del Evento

El objeto `event` contiene información sobre el evento activado.

Propiedad	Descripción
`event.target`	Elemento que originó el evento
`event.pageX` / `event.pageY`	Coordenadas del cursor
`event.which`	Tecla o botón del ratón presionado

Ejemplo: Capturar la posición del ratón

```
$(document).mousemove(function(event) {
  console.log("Posición: X=" + event.pageX + " Y=" + event.pageY);
```

```
});
```

6. Posición y Tamaño de Elementos

Método	Descripción
`.offset()`	Devuelve la posición **absoluta** de un elemento
`.position()`	Devuelve la posición **relativa** dentro de su contenedor
`.height()` / `.width()`	Obtiene o cambia el tamaño de un elemento
`.scrollTop()` / `.scrollLeft()`	Obtiene o cambia la posición del scroll

Ejemplo: Obtener posición absoluta

```
var posicion = $("#miDiv").offset();
console.log("Posición: Left=" + posicion.left + " Top=" + posicion.top);
```

Ejemplo: Mover un elemento 50px a la derecha

```
var pos = $("#miDiv").offset();
pos.left += 50;
$("#miDiv").offset(pos);
```

▼ jQuery: Animaciones y Efectos

1. JSON en jQuery

- **JSON (JavaScript Object Notation)** es un formato para intercambiar datos estructurados.
- Se usa frecuentemente en jQuery para manipular datos y realizar peticiones AJAX.
- **Ejemplo de JSON:**

```
var usuario = {
  nombre: "Carlos",
  edad: 25,
```

```
  ciudad: "Madrid"
};
console.log(usuario.nombre); // "Carlos"
```

- **Ejemplo de array en JSON:**

```
var colores = ["Rojo", "Verde", "Azul"];
```

- **Ejemplo de objeto con arrays en JSON:**

```
var empresa = {
  nombre: "TechCorp",
  empleados: ["Ana", "Luis", "Pedro"]
};
```

---

2. Métodos para Mostrar y Ocultar Elementos

Método	Descripción
`.show()`	Muestra elementos ocultos
`.hide()`	Oculta elementos visibles
`.toggle()`	Alterna entre mostrar y ocultar

Ejemplo: Mostrar/Ocultar con Botones

```
$("#mostrar").click(function() {
  $("p").show();
});

$("#ocultar").click(function() {
  $("p").hide();
});

$("#alternar").click(function() {
  $("p").toggle();
});
```

---

3. Animaciones con Duración

- Se pueden añadir animaciones especificando **duración (milisegundos)** y una **función de completado**.

Ejemplo: Mostrar con Animación

```
$("p").show(600, function() {
  alert("Párrafo mostrado");
});
```

Ejemplo: Ocultar con Animación

```
$("p").hide(600, function() {
  alert("Párrafo oculto");
});
```

Ejemplo: Alternar con Animación

```
$("p").toggle(600, function() {
  alert("Cambiado");
});
```

---

4. Animaciones Predefinidas

Método	Descripción
`.fadeIn()`	Muestra elementos con transición de opacidad
`.fadeOut()`	Oculta elementos con transición de opacidad
`.fadeToggle()`	Alterna entre `.fadeIn()` y `.fadeOut()`
`.fadeTo(duración, opacidad)`	Ajusta la opacidad de un elemento
`.slideDown()`	Muestra elementos deslizándolos
`.slideUp()`	Oculta elementos deslizándolos
`.slideToggle()`	Alterna entre `.slideDown()` y `.slideUp()`

Ejemplo: Desvanecer Elemento

```
$("#boton").click(function() {
  $("p").fadeOut(600);
});
```

Ejemplo: Desplegar Elemento

```
$("#boton").click(function() {
  $("p").slideDown(400);
});
```

---

5. Animaciones Personalizadas con `.animate()`

- Se pueden animar **propiedades CSS numéricas**.
- **Ejemplo de sintaxis:**

```
$(selector).animate({ propiedad: valor }, duración, easing, compl
etado);
```

Ejemplo: Mover un `div` a la derecha

```
$("#miDiv").animate({ left: "+=100px" }, 500);
```

Ejemplo: Animar varias propiedades

```
$("#miDiv").animate({
  opacity: 0.5,
  height: "toggle",
  left: "+=50px"
}, 1000);
```

---

6. Encolar Animaciones

- Por defecto, jQuery **ejecuta animaciones en secuencia**.
- Para ejecutar **varias animaciones al mismo tiempo**, usa `queue: false`.

Ejemplo: Animación en cola

```
$("#caja").animate({ width: "90%" }, { queue: false, duration: 2000 })
          .animate({ fontSize: "24px" }, 1000)
          .animate({ borderLeftWidth: "10px" }, 1000);
```

7. Detener Animaciones

Método	Descripción
`.stop()`	Detiene una animación en curso
`.finish()`	Completa inmediatamente todas las animaciones en cola

Ejemplo: Detener animación al salir del ratón

```
$("#caja").mouseover(function() {
  $(this).animate({ left: "+=200px" }, 1000);
}).mouseout(function() {
  $(this).stop();
});
```

8. Problemas con Animaciones y Soluciones

- **Problema:** Acceder a una propiedad CSS mientras se anima puede dar valores inesperados.
- **Solución 1:** Usar una **función de completado**.

  ```
  $("#caja").animate({ left: "+=100px" }, 500, function() {
    console.log("Posición final: " + $(this).css("left"));
  });
  ```

- **Solución 2:** Usar una **variable centinela**.

  ```
  var enAnimacion = false;

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

```
  if (!enAnimacion) {
    enAnimacion = true;
    $("#caja").animate({ left: "+=100px" }, 500, function() {
      enAnimacion = false;
    });
  }
});
```