Tabla de contenidos

1. Introducción a jQuery

1. ¿Qué es jQuery?

Definición

jQuery es una biblioteca de JavaScript rápida, pequeña y rica en funcionalidades que simplifica la manipulación de documentos HTML, el manejo de eventos, las animaciones y las peticiones AJAX.

Lema de jQuery: *"Write less, do more"* (Escribe menos, haz más)

Analogía Práctica

```
JavaScript Vanilla = Cocinar desde cero con ingredientes básicos
jQuery = Usar electrodomésticos que facilitan la cocina

🍳 Tarea: Hacer puré de papas
──────────────────────────────────────────────

JavaScript Vanilla:
1. Hervir agua manualmente
2. Pelar papas una por una
3. Hervir papas vigilando el tiempo
4. Escurrir manualmente
5. Aplastar con tenedor (mucho trabajo)

jQuery:
1. Usar olla express (automática)
2. Usar pelador eléctrico
3. Usar procesador de alimentos (rápido y fácil)

Resultado: El mismo puré, pero con menos esfuerzo ⚡
```

¿Por qué usar jQuery?

- **Simplicidad**: Reduce la complejidad del código.
- **Compatibilidad**: Funciona en todos los navegadores.
- **Productividad**: Desarrollo más rápido.
- **Comunidad**: Gran ecosistema de plugins.
- **Documentación**: Excelente documentación y ejemplos.

2. Ventajas sobre JavaScript Vanilla

Tabla Comparativa de Ventajas

Aspecto	JavaScript Vanilla	jQuery
Líneas de código	Más verboso	Más conciso
Compatibilidad	Manual (polyfills)	Automática
Curva aprendizaje	Más empinada	Más suave
Animaciones	Código complejo	Métodos simples

Cuándo Usar jQuery vs JavaScript Vanilla

```
// ✅ USAR JQUERY CUANDO:
// - Prototipado rápido
// - Proyectos con muchas manipulaciones DOM
// - Necesitas compatibilidad con navegadores antiguos
// - Animaciones y efectos frecuentes

// ✅ USAR JAVASCRIPT VANILLA CUANDO:
// - Performance crítica
// - Aplicaciones muy grandes (frameworks modernos)
// - Solo navegadores modernos
// - Quieres entender JavaScript profundamente
```

3. Instalación e Incorporación

Método 1: CDN (Content Delivery Network)

Ventajas del CDN:

- ✅ No necesitas descargar archivos.
- ✅ Mejor rendimiento (cacheo global).
- ✅ Perfecto para aprender y prototipos.

```
<!-- jQuery desde CDN - Siempre antes de tu código -->
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>

<!-- Tu código JavaScript -->
<script>
    console.log("jQuery cargado desde CDN");
</script>
```

Método 2: Descarga Local

Ventajas de descarga local:

- ✅ Control total sobre la versión.
- ✅ Funciona sin internet.
- ✅ Mejor para proyectos de producción.

```
<!-- jQuery desde archivo local -->
<script src="js/jquery-3.7.1.min.js"></script>

<!-- Tu código -->
<script src="js/mi-script.js"></script>
```

4. Resumen del Capítulo

Conceptos Vistos

- ✅ **Definición de jQuery**: Biblioteca que simplifica JavaScript.

- ✅ **Ventajas principales**: Menos código, mejor compatibilidad, más productividad.
- ✅ **Métodos de instalación**: CDN y descarga local.
- ✅ **Verificación**: Cómo confirmar que jQuery está funcionando.

2. Selectores jQuery

1. Sintaxis Básica $()

La Función $ (Dólar)

El símbolo `$` es el núcleo de jQuery. Es una función que puede realizar tres tareas principales:

1. Seleccionar elementos del DOM.
2. Crear nuevos elementos.
3. Ejecutar código cuando el DOM esté listo.

```
// 1. SELECCIONAR ELEMENTOS
$("p")              // Selecciona todos los párrafos
$("#miId")          // Selecciona elemento con ID "miId"
$(".miClase")       // Selecciona elementos con clase "miClase"

// 2. CREAR ELEMENTOS
$("<p>Nuevo párrafo</p>")   // Crea un párrafo

// 3. DOCUMENT READY
$(document).ready(function() {
    // Código que se ejecuta cuando el DOM está listo
});

// Forma abreviada (más común)
$(function() {
    // DOM listo
});
```

2. Selectores CSS en jQuery

jQuery utiliza **la misma sintaxis que CSS** para seleccionar elementos:

Selectores Básicos

```
// Por etiqueta
$("p")              // Todos los <p>

// Por ID
$("#titulo")        // <element id="titulo">

// Por clase
$(".destacado")     // <element class="destacado">

// Por atributo
$("[type='text']")  // <input type="text">
```

Selectores Combinados

```
// Descendentes
$("div p")              // <p> dentro de cualquier <div>

// Hijos directos
$("ul > li")            // <li> que son hijos directos de <ul>

// Hermanos
$("h1 + p")             // <p> inmediatamente después de <h1>

// Múltiples
$("h1, h2, h3")         // Todos los títulos
```

3. Filtros y Pseudo-selectores

Filtros de Posición

```
$("li:first")           // Primer <li>
$("li:last")            // Último <li>
$("p:even")             // Párrafos en posiciones pares (0, 2, 4...)
$("p:odd")              // Párrafos en posiciones impares (1, 3, 5...)
$("li:eq(2)")           // <li> en la posición 2 (tercero)
```

Filtros de Contenido

```
$(":contains('texto')")     // Elementos que contienen "texto"
$(":empty")                 // Elementos vacíos
$(":has(p)")                // Elementos que contienen un <p>
$(":not(.oculto)")          // Elementos que NO tienen la clase .oculto
```

Filtros de Formulario

```
$(":input")             // Todos los elementos de formulario
$(":text")              // <input type="text">
$(":checkbox")          // <input type="checkbox">
$(":checked")           // Checkboxes/radios seleccionados
$(":disabled")          // Elementos deshabilitados
```

4. Selectores Avanzados y Optimización

Encadenamiento de Filtros

```
$("li:visible:even")        // <li> visibles en posiciones pares
$("input:enabled:text")     // Inputs de texto habilitados
```

Optimización de Selectores

Para un mejor rendimiento, es bueno seguir algunas prácticas:

- ✅ **Usar IDs:** `$("#mild")` es el selector más rápido.
- ✅ **Ser específico pero no demasiado:** `$("#menu a")` es mejor que `$(".nav .container .menu-principal a")`.
- ✅ **Cachear selectores:** Guarda selecciones frecuentes en variables.

```
// Cachear un selector para reutilizarlo
let $menuLinks = $("#menu a");

$menuLinks.addClass("enlace");
$menuLinks.css("color", "blue");
$menuLinks.hover(function() { /* ... */ });
```

5. Resumen del Capítulo

Conceptos Vistos

- ✅ **Sintaxis básica $()**: Seleccionar, crear y Document Ready.
- ✅ **Selectores CSS**: Uso de etiquetas, IDs, clases y combinaciones.
- ✅ **Filtros y pseudo-selectores**: Para selecciones precisas.
- ✅ **Optimización**: Mejores prácticas para un código eficiente.

3. Manipulación del DOM con jQuery

1. Modificación de Contenido

text() - Contenido de Texto

El método `text()` maneja el contenido de texto plano de los elementos, sin interpretar HTML.

```
// OBTENER TEXTO
let texto = $("#titulo").text();

// ESTABLECER TEXTO
$("#titulo").text("Nuevo título");

// TEXTO CON MÚLTIPLES ELEMENTOS
$("p").text("Mismo texto para todos");
```

html() - Contenido HTML

El método `html()` maneja contenido HTML completo, interpretando etiquetas.

```
// OBTENER HTML
let contenidoHTML = $("#contenedor").html();

// ESTABLECER HTML
$("#contenedor").html("<p>Nuevo <strong>párrafo</strong></p>");
```

val() - Valores de Formulario

El método `val()` maneja valores de elementos de formulario (input, select, textarea).

```
// OBTENER VALORES
let nombre = $("#nombre").val();

// ESTABLECER VALORES
$("#nombre").val("Juan Pérez");

// LIMPIAR VALORES
$("#formulario input").val("");
```

2. Manipulación de Atributos

attr() - Atributos HTML

El método `attr()` maneja atributos HTML estándar y personalizados.

```
// OBTENER ATRIBUTOS
let src = $("#imagen").attr("src");

// ESTABLECER ATRIBUTOS
$("#imagen").attr("src", "nueva-imagen.jpg");

// MÚLTIPLES ATRIBUTOS
$("#imagen").attr({
    "src": "imagen.jpg",
    "alt": "Descripción de la imagen"
});

// ELIMINAR ATRIBUTOS
$("#elemento").removeAttr("data-id");
```

prop() - Propiedades DOM

El método `prop()` maneja propiedades DOM (estado interno de elementos).

```
// OBTENER PROPIEDADES
let checked = $("#checkbox").prop("checked"); // true/false

// ESTABLECER PROPIEDADES
$("#checkbox").prop("checked", true);
```

Diferencias: attr() vs prop()

Usa `attr()` para atributos HTML (src, href, data-*) y `prop()` para propiedades de estado (checked, disabled, selected).

3. Manipulación de Clases

addClass() - Agregar Clases

```
$("#elemento").addClass("activo");
$("#elemento").addClass("activo destacado grande");
```

removeClass() - Eliminar Clases

```
$("#elemento").removeClass("activo");
$("#elemento").removeClass(); // Elimina TODAS las clases
```

toggleClass() - Alternar Clases

```
$("#elemento").toggleClass("activo");
```

hasClass() - Verificar Clases

```
if ($("#elemento").hasClass("activo")) {
    console.log("El elemento tiene la clase activo");
}
```

4. Resumen del Capítulo

Conceptos Vistos

- ✅ **Modificación de contenido**: text(), html(), val()
- ✅ **Manipulación de atributos**: attr(), prop(), removeAttr()
- ✅ **Gestión de clases**: addClass(), removeClass(), toggleClass(), hasClass()
- ✅ **Diferencias importantes**: attr() vs prop(), cuándo usar cada uno

4. Efectos y Animaciones jQuery

1. Efectos Básicos

show(), hide(), toggle()

```
// MOSTRAR Y OCULTAR ELEMENTOS

// Mostrar elemento
$("#elemento").show();              // Instantáneo
$("#elemento").show("slow");        // Lento (600ms)
$("#elemento").show("fast");        // Rápido (200ms)
$("#elemento").show(1000);          // Tiempo personalizado en ms

// Ocultar elemento
$("#elemento").hide();              // Instantáneo
$("#elemento").hide("slow");        // Lento
$("#elemento").hide(500);           // 500ms

// Alternar visibilidad
$("#elemento").toggle();            // Muestra si está oculto, oculta si está visible
$("#elemento").toggle("fast");      // Con velocidad
```

2. Efectos Fade

fadeIn(), fadeOut(), fadeToggle()

```
// EFECTOS DE DESVANECIMIENTO

// Aparecer gradualmente
$("#elemento").fadeIn();            // Velocidad normal
$("#elemento").fadeIn("slow");      // Lento
$("#elemento").fadeIn(800);         // 800ms

// Desaparecer gradualmente
$("#elemento").fadeOut();           // Velocidad normal
$("#elemento").fadeOut("fast");     // Rápido
$("#elemento").fadeOut(500);        // 500ms

// Alternar fade
$("#elemento").fadeToggle();        // Alterna fadeIn/fadeOut
$("#elemento").fadeToggle("slow");  // Con velocidad

// Fade a opacidad específica
$("#elemento").fadeTo("slow", 0.5); // Fade a 50% opacidad
$("#elemento").fadeTo(1000, 0.2);   // Fade a 20% en 1 segundo
```

3. Efectos Slide

slideUp(), slideDown(), slideToggle()

```
// EFECTOS DE DESLIZAMIENTO

// Deslizar hacia abajo (mostrar)
$("#elemento").slideDown();         // Velocidad normal
$("#elemento").slideDown("slow");   // Lento
$("#elemento").slideDown(600);      // 600ms

// Deslizar hacia arriba (ocultar)
$("#elemento").slideUp();           // Velocidad normal
$("#elemento").slideUp("fast");     // Rápido
$("#elemento").slideUp(400);        // 400ms

// Alternar slide
$("#elemento").slideToggle();       // Alterna slideUp/slideDown
$("#elemento").slideToggle("slow"); // Con velocidad
```

4. Animaciones Personalizadas

animate()

```
// ANIMACIÓN BÁSICA
$("#elemento").animate({
    width: "300px",
    height: "200px",
    opacity: 0.8
}, 1000);

// ANIMACIÓN CON CALLBACK
$("#elemento").animate({
    left: "250px",
    top: "100px"
}, "slow", function() {
    console.log("Animación completada");
});

// ENCADENAR ANIMACIONES
$("#elemento")
    .animate({ width: "200px" }, 500)
    .animate({ height: "200px" }, 500)
    .animate({ opacity: 0.5 }, 300);
```

5. Velocidades y Callbacks

Velocidades Predefinidas

```
// VELOCIDADES DISPONIBLES
$("#elemento").show("slow");    // 600ms
$("#elemento").show("fast");    // 200ms
$("#elemento").show();          // 400ms (por defecto)
$("#elemento").show(1500);      // Tiempo personalizado
```

Callbacks (Funciones al completar)

```
// EJECUTAR CÓDIGO AL TERMINAR EFECTO
$("#elemento").hide("slow", function() {
    console.log("Elemento ocultado");
    $("#otro-elemento").show();
});

$("#elemento").fadeOut(800, function() {
    $(this).remove();  // Eliminar del DOM después del fade
});
```

6. Resumen del Capítulo

Conceptos Vistos

- ✅ **show/hide/toggle**: Mostrar y ocultar elementos
- ✅ **fadeIn/fadeOut**: Efectos de desvanecimiento
- ✅ **slideUp/slideDown**: Efectos de deslizamiento
- ✅ **animate()**: Animaciones personalizadas básicas
- ✅ **Velocidades**: slow, fast, tiempos personalizados

5. Introducción a Plugins

1. ¿Qué son los Plugins de jQuery?

Definición y Concepto

Los plugins de jQuery son extensiones que añaden funcionalidades específicas a la biblioteca base de jQuery.

```
// jQuery base proporciona:
$("#elemento").hide();              // Funcionalidad básica
$("#elemento").addClass("clase");   // Manipulación básica

// Los plugins añaden:
$("#slider").slick();               // Plugin de slider
$("#galeria").lightbox();           // Plugin de lightbox
$("#formulario").validate();        // Plugin de validación
```

¿Por qué usar Plugins?

- ✅ Ahorro de tiempo de desarrollo.
- ✅ Código probado y optimizado.
- ✅ Comunidad activa y soporte.
- ✅ Configuración flexible.
- ✅ Compatibilidad con diferentes navegadores.

2. Instalación y Configuración

Métodos de Instalación

La forma más común es a través de un CDN (Content Delivery Network), que permite enlazar los archivos del plugin directamente en tu HTML.

```
<!-- 1. jQuery (requerido) -->
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>

<!-- 2. Archivos del Plugin (CSS y JS) -->
<link rel="stylesheet" href="URL_DEL_PLUGIN.css">
<script src="URL_DEL_PLUGIN.js"></script>
```

Orden de Carga Importante

El orden en que cargas los archivos es crucial:

1. **CSS de plugins:** Generalmente en el `<head>`.
2. **jQuery base:** Siempre primero antes de los plugins.
3. **Plugins de jQuery:** Después de jQuery.
4. **Tu código personalizado:** Al final, para que pueda usar jQuery y los plugins.

3. Plugins Populares y Ejemplos

Plugin 1: Slick Carousel (Slider de imágenes)

```
// HTML
<div class="slider">
  <div><h3>1</h3></div>
  <div><h3>2</h3></div>
  <div><h3>3</h3></div>
</div>

// JavaScript
$(document).ready(function(){
  $('.slider').slick({
    dots: true,
    infinite: true,
    speed: 300
  });
});
```

Plugin 2: jQuery Validation (Validación de formularios)

```
// HTML
<form id="mi-formulario">
  <input type="text" name="nombre" required minlength="2">
  <input type="email" name="email" required>
  <button type="submit">Enviar</button>
</form>

// JavaScript
$(document).ready(function(){
  $("#mi-formulario").validate();
});
```

4. Creando un Plugin Básico

Crear un plugin simple es una excelente manera de entender cómo funcionan.

```
(function($) {
    $.fn.resaltar = function(opciones) {
        // Configuración por defecto
        var config = $.extend({
            color: 'yellow'
        }, opciones);

        // Aplicar a cada elemento
        return this.each(function() {
            $(this).css('background-color', config.color);
        });
    };
})(jQuery);

// Uso del plugin
$(".mi-elemento").resaltar({ color: 'lightblue' });
```

5. Resumen del Capítulo

Conceptos Vistos

- ✅ **Qué son los plugins**: Extensiones de funcionalidad para jQuery.
- ✅ **Instalación**: CDN, descarga local, y orden de carga.
- ✅ **Plugins populares**: Slick, Validation.
- ✅ **Creación de plugins**: Estructura básica y personalización.
- ✅ **Mejores prácticas**: Performance, debugging, gestión.