UNIVERSIDAD AUTÓNOMA DE ENCARNACIÓN

FACULTAD DE CIENCIAS, ARTE Y TECNOLOGÍA

TRABAJO DE INVESTIGACION JQUERY VS JAVASCRIPT VANILLA

Diseño Y Programación Web II

Análisis De Sistemas.

Prof. Osvaldo Enrique Micniuk

Franco Vergara

Kevin Nieto Vera

Alejandro Dutil

Katsuo Kumagai

David Rodriguez

CURSO: 2° año

Encarnación – Paraguay

Abril – 2023

Introduccion

En el desarrollo web, la elección entre utilizar una biblioteca como jQuery o trabajar directamente con JavaScript Vanilla es una decisión crucial que afecta la eficiencia, el rendimiento y la facilidad de mantenimiento de un proyecto. Ambas opciones tienen sus ventajas y desventajas, y es importante comprenderlas para tomar decisiones informadas al comenzar un nuevo proyecto o al realizar mejoras en uno existente.
A continuación, haremos una comparativa para el análisis para comprender las fortalezas y limitaciones de cada enfoque.

1. Obtener Elementos del DOM: ID, clase, etiqueta

- jQuery: Utiliza selectores como $('#id'), $('.clase'), $('etiqueta').
- JavaScript Vanilla: Utiliza métodos como document.getElementById('id'), document.getElementsByClassName('clase'), document.getElementsByTagName('etiqueta').

```
// jQuery
$('#id').text('Elemento obtenido por ID en jQuery');

// JavaScript Vanilla
document.getElementById('id').textContent = 'Elemento obtenido por ID en JavaScript Vanilla';
```

2. Modificar Contenido: Texto, atributos, estilos

- jQuery: Emplea métodos como .text(), .attr(), .css().
- JavaScript Vanilla: Utiliza propiedades y métodos directamente.

```
// jQuery
$('.clase').attr('href', 'https://www.ejemplo.com').text('Enlace modificado por jQuery');

// JavaScript Vanilla
const element = document.querySelector('.clase');
element.setAttribute('href', 'https://www.ejemplo.com');
element.textContent = 'Enlace modificado por JavaScript Vanilla';
```

3. Agregar/Eliminar Elementos del DOM

- jQuery: Usa métodos como .append(), .prepend(), .remove().
- JavaScript Vanilla: Emplea métodos como appendChild(), insertBefore(), removeChild().

```
// jQuery
$('.contenedor').append('<div>Nuevo elemento agregado por jQuery</div>');

// JavaScript Vanilla
const nuevoElemento = document.createElement('div');
nuevoElemento.textContent = 'Nuevo elemento agregado por JavaScript Vanilla';
```

```
document.querySelector('.contenedor').appendChild(nuevoElemento);
```

4. Manejar Eventos del Usuario: Clicks, desplazamientos

- jQuery: Usa métodos como .click(), .scroll().
- JavaScript Vanilla: Emplea .addEventListener().

```
// jQuery
$('#boton').click(function() {
 alert('Haz clic en jQuery');
});

// JavaScript Vanilla
document.getElementById('boton').addEventListener('click', function() {
 alert('Haz clic en JavaScript Vanilla');
});
```

5. Anidar Elementos

- jQuery: Utiliza métodos como .append(), .prepend().
- JavaScript Vanilla: Emplea métodos como appendChild(), insertBefore().

```
// jQuery
$('.padre').append('<div>Hijo agregado por jQuery</div>');

// JavaScript Vanilla
const nuevoHijo = document.createElement('div');
nuevoHijo.textContent = 'Hijo agregado por JavaScript Vanilla';
document.querySelector('.padre').appendChild(nuevoHijo);
```

6. Animar Elementos

- jQuery: Usa métodos como .animate().
- JavaScript Vanilla: Requiere el uso de CSS o librerías externas para animaciones complejas.

```
// jQuery
$('.caja').animate({ left: '100px' }, 'slow');

// JavaScript Vanilla con CSS
document.querySelector('.caja').style.transition = 'left 0.5s ease-in-out';
document.querySelector('.caja').style.left = '100px';
```

7. Estilos de Elementos: Obtener y establecer

- jQuery: Utiliza métodos como .css().
- JavaScript Vanilla: Emplea la propiedad .style del elemento.

```
// jQuery
$('.elemento').css('color', 'red');

// JavaScript Vanilla
document.querySelector('.elemento').style.color = 'red';
```

8. Posición y Tamaño de Elementos

- jQuery: Emplea métodos como .position(), .width(), .height().
- JavaScript Vanilla: Utiliza propiedades como .offsetTop, .offsetWidth, .offsetHeight.

```
// jQuery
const posicion = $('.elemento').position();
const ancho = $('.elemento').width();
const alto = $('.elemento').height();

// JavaScript Vanilla
const elemento = document.querySelector('.elemento');
const posicionTop = elemento.offsetTop;
const anchoElemento = elemento.offsetWidth;
const altoElemento = elemento.offsetHeight;
```

9. Eventos: Manejo, asignación, manipulación

- jQuery: Utiliza métodos como .on(), .off().
- JavaScript Vanilla: Emplea .addEventListener(), .removeEventListener().

```
// jQuery
$('#elemento').on('click', function() {
 console.log('Haz clic en jQuery');
});
$('#elemento').off('click'); // Desactivar evento

// JavaScript Vanilla
document.getElementById('elemento').addEventListener('click', function() {
```

```
 console.log('Haz clic en JavaScript Vanilla');
});
document.getElementById('elemento').removeEventListener('click', function() {
 console.log('Evento removido en JavaScript Vanilla');
});
```

10. Valores de Atributos: Establecer y obtener.

JavaScript Vanilla:

- Para obtener un atributo: getAttribute('nombreAtributo')
- Para establecer un atributo: setAttribute('nombreAtributo', 'valor')
- Para obtener o establecer el valor de un elemento: elemento.value

jQuery:

- Para obtener o establecer un atributo: attr('nombreAtributo', 'valor')
- Para obtener o establecer el valor de un elemento: val('nuevoValor')

```
// Ejemplo de JavaScript vanilla const elemento =
document.getElementById('miElemento'); const atributo =
elemento.getAttribute('nombreAtributo');
elemento.setAttribute('nombreAtributo', 'nuevoValor'); const valor =
elemento.value; elemento.value = 'nuevoValor';
```

```
// Ejemplo de jQuery
 const elemento = $('#miElemento'); const atributo =
elemento.attr('nombreAtributo'); elemento.attr('nombreAtributo', 'nuevoValor');
const valor = elemento.val(); elemento.val('nuevoValor');
```

11. Manejar Eventos del Usuario: Clicks, desplazamientos.

JavaScript Vanilla

- Para añadir un event listener para un clic (click event): elemento.addEventListener('click', callback)
- Para añadir un event listener para un desplazamiento (scroll event): window.addEventListener('scroll', callback)

jQuery:

- Para añadir un event listener para un clic (click event): $(selector).click(callback)
- Para añadir un event listener para un desplazamiento (scroll event): $(window).scroll(callback)

```
// Ejemplo de JavaScript vanilla
.const boton = document.getElementById('miBoton');
// Manejar un clic en el botón
```

```
boton.addEventListener('click', function() { console.log('¡Haz hecho clic en el botón!'); });
// Manejar un desplazamiento de la ventana window.addEventListener('scroll', function() { console.log('¡La ventana se ha desplazado!'); });

// Ejemplo de jQuery para manejar eventos
const boton = $('#miBoton');

// Manejar un clic en el botón
boton.click(function() {
    console.log('¡Haz hecho clic en el botón!');
});

// Manejar un desplazamiento de la ventana
$(window).scroll(function() {
    console.log('¡La ventana se ha desplazado!');
});
```

Comparación de Rendimiento y Facilidad de Uso

- Sintaxis: jQuery ofrece una sintaxis más compacta y fácil de recordar para realizar tareas comunes.
- Facilidad de Uso: jQuery es más amigable para los principiantes y reduce la cantidad de código necesario en comparación con JavaScript Vanilla.
- Rendimiento: JavaScript Vanilla tiende a ser más rápido en operaciones simples y en proyectos grandes debido a su menor sobrecarga y dependencia de librerías externas.

En última instancia, la elección entre jQuery y JavaScript Vanilla depende de una serie de factores, incluido el tamaño y la complejidad del proyecto, las habilidades y preferencias del equipo de desarrollo, y los objetivos específicos en términos de rendimiento y tiempo de desarrollo. Si bien jQuery sigue siendo una opción viable y popular, especialmente para proyectos más pequeños y para desarrolladores que buscan una curva de aprendizaje más suave, JavaScript Vanilla ofrece un mayor control y optimización para proyectos más grandes y exigentes.
Al comprender las fortalezas y limitaciones de cada enfoque, los desarrolladores pueden tomar decisiones informadas que les permitan crear aplicaciones web eficientes, potentes y sostenibles en el tiempo.

Bibliografía

- jQuery. (s.f.). Documentación oficial de jQuery. Recuperado de https://api.jquery.com/

- MDN Web Docs. (s.f.). JavaScript. Recuperado de https://developer.mozilla.org/en-US/docs/Web/JavaScript

- Stack Overflow. (s.f.). Preguntas etiquetadas ‘jquery’. Recuperado de https://stackoverflow.com/questions/tagged/jquery