# Guía Rápida de jQuery y AJAX para Exámenes

## 1. Introducción a jQuery

jQuery es una biblioteca de JavaScript que simplifica la manipulación del DOM, el manejo de eventos, la animación y las interacciones AJAX. Es ampliamente utilizada debido a su simplicidad y potencia.

## 2. $(document).ready()

El método $(document).ready() asegura que el código se ejecute solo después de que el DOM esté completamente cargado. Esto es crucial para evitar errores al manipular elementos del DOM que aún no existen.

Ejemplo:

```
$(document).ready(function() {
    // Código a ejecutar cuando el DOM está listo
    $("p").text("El DOM está completamente cargado y listo.");
});
```

## 3. Realizar Peticiones AJAX con jQuery

jQuery facilita las peticiones AJAX con métodos como $.ajax(), $.get(), y $.post(). Estos métodos permiten interactuar con el servidor de manera asíncrona, obteniendo o enviando datos sin recargar la página.

Ejemplo básico de $.ajax():

```
$.ajax({
    url: '/mensajes',
    type: 'GET',
    success: function(data) {
        console.log(data);
    },
    error: function(error) {
        console.error('Error:', error);
    }
});
```

4. Ejemplo Completo con AJAX y jQuery

Para construir la página parteAJAX.html que realiza una petición AJAX al servlet /mensajes y muestra los mensajes, podemos usar el siguiente código:

HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Mensajes AJAX</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

```
    <script src="script.js"></script>
</head>
<body>
    <select id="messageSelect"></select>
    <textarea id="messageContent" rows="10" cols="50"></textarea>
</body>
</html>
```

JavaScript (script.js):

```
$(document).ready(function() {
    // Realizar la petición AJAX al cargar la página
    $.ajax({
        url: '/mensajes',
        type: 'GET',
        success: function(data) {
            // Suponiendo que data es un array de objetos con 'id' y 'text'
            var select = $('#messageSelect');
            data.forEach(function(message) {
                select.append(new Option(message.id, message.text));
            });
        },
        error: function(error) {
            console.error('Error:', error);
        }
    });
```

```
    // Manejar el cambio de selección en el select
    $('#messageSelect').change(function() {
        var selectedText = $(this).val();
        $('#messageContent').val(selectedText);
    });
});
```

5. Métodos Adicionales de jQuery para AJAX

$.get()

Método abreviado para realizar peticiones GET.

```
$.get('/mensajes', function(data) {
    console.log(data);
});
```

$.post()

Método abreviado para realizar peticiones POST.

```
$.post('/enviar', { t: 'Nuevo mensaje' }, function(data) {
    console.log(data);
});
```

$.ajax()

Método más flexible y completo para realizar peticiones AJAX.

```
$.ajax({
    url: '/ruta',
    type: 'POST',
    data: { key: 'value' },
    success: function(data) {
        console.log('Éxito:', data);
    },
    error: function(xhr, status, error) {
        console.error('Error:', error);
    }
});
```

6. Manejo de Errores en AJAX

Es importante manejar errores en las peticiones AJAX para mejorar la experiencia del usuario y depurar problemas.

Ejemplo:

```
$.ajax({
    url: '/ruta',
    type: 'GET',
    success: function(data) {
        console.log('Datos recibidos:', data);
    },
```

```
    error: function(xhr, status, error) {
        console.error('Error:', error);
        alert('Ocurrió un error al procesar la solicitud.');
    }
});
```

## 7. Conclusión

jQuery simplifica enormemente el uso de AJAX en aplicaciones web, permitiendo realizar peticiones asíncronas de manera sencilla y eficiente. Dominar estos conceptos es clave para resolver problemas relacionados con la manipulación dinámica de datos en exámenes y en la práctica profesional.