Facultad de
ESTADÍSTICA E INFORMÁTICA

Licenciatura en Ingeniería en Sistemas
y Tecnologías de la Información

Universidad Veracruzana

Actividad | Refactorización

LUNES 12 DE MAYO DE 2025

SALDAÑA MARLENE

Ejercicio de Refactorización

Introducción

En este trabajo se presenta la comparación entre versiones no modulares y modulares de diversas páginas web construidas con HTML y JavaScript. El objetivo principal es destacar las ventajas de separar el código JavaScript en archivos externos (.js), lo cual permite una estructura más limpia, profesional y fácil de mantener

Desarrollo:

Version no modular: **Página01**

pagina01.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pagina 01</title>
</head>

<body>
    <button onclick="loadXMLDoc()">Ejecutar función</button>
    <table id="demo"></table>

<script>
function loadXMLDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    const xmlDoc = xhttp.responseXML;
    const cd = xmlDoc.getElementsByTagName("CD");
    myFunction(cd);
  }
  xhttp.open("GET", "cd_catalog.xml");
  xhttp.send();
}

function myFunction(cd) {
  let table="<tr><th>Artist</th><th>Title</th></tr>";
  for (let i = 0; i < cd.length; i++) {
    table += "<tr><td>" +
    cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
    "</td><td>" +
    cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
    "</td></tr>";
  }
  document.getElementById("demo").innerHTML = table;
}
</script>

</body>
</html>

</body>
</html>
```

Version modular:

pagina01.html > html > body > script

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Página 01 (Modular)</title>
</head>
<body>
  <button id="btnLoad">Ejecutar función</button>
  <table id="demo"></table>

  <script type="module" src="pagina01.js"></script>
</body>
</html>

```

pagina01.js > ...

```javascript
function loadXMLDoc() {
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function () {
      const xmlDoc = xhttp.responseXML;
      const cd = xmlDoc.getElementsByTagName("CD");
      myFunction(cd);
    };
    xhttp.open("GET", "cd_catalog.xml");
    xhttp.send();
  }

  function myFunction(cd) {
    let table = "<tr><th>Artist</th><th>Title</th></tr>";
    for (let i = 0; i < cd.length; i++) {
      table += "<tr><td>" +
        cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
        "</td><td>" +
        cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
        "</td></tr>";
    }
    document.getElementById("demo").innerHTML = table;
  }

  document.getElementById("btnLoad").addEventListener("click", loadXMLDoc);

```

Versión no modular

- Todo el código JavaScript está dentro del mismo archivo HTML, dentro de una etiqueta <script>.
- El botón usa onclick="loadXMLDoc()" directamente en el HTML, lo cual no es lo más recomendable porque mezcla el diseño con el código.
- No hay separación entre lo visual (HTML) y la lógica (JS), lo que puede hacer que el código sea más difícil de entender o modificar si el proyecto crece.
- No se usa addEventListener, por lo tanto el manejo de eventos no está bien estructurado.
- Cuando se carga la página, el navegador lee y ejecuta el JavaScript al momento en que lo encuentra, todo junto.
- En la pestaña del navegador, solo se ve que se cargó el archivo HTML, y dentro de él va todo incluido.

Versión modular

- El JavaScript está en un archivo aparte llamado pagina01.js, lo cual ayuda a mantener el código más limpio.
- En el HTML, se usa <script type="module" src="pagina01.js"></script> para conectarlo, indicando que es un módulo.
- El botón ya no usa onclick; ahora se conecta con addEventListener, que es una forma más moderna y ordenada.
- El HTML solo tiene lo visual, y el JS tiene toda la lógica. Esto lo hace más fácil de leer, entender y modificar.

Version no modular: **Página02**

pagina02.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pagina 02</title>
</head>

<body>

    <button onclick="loadXMLDoc()">Ejecutar función</button>
    <table id="demo"></table>

<script>
    // Función principal usando async/await y fetch
    const loadXMLDoc = async () => {
      try {
        const response = await fetch('cd_catalog.xml');
        const text = await response.text();
        const parser = new DOMParser();
        const xmlDoc = parser.parseFromString(text, "application/xml");
        const cds = xmlDoc.getElementsByTagName("CD");
        myFunction(cds);
      } catch (error) {
        console.error('Error loading XML:', error);
      }
    };

    const myFunction = (cds) => {
      let table = "<tr><th>Artist</th><th>Title</th></tr>";
      Array.from(cds).forEach(cd => {
        const artist = cd.getElementsByTagName("ARTIST")[0]?.textContent || "N/A";
        const title = cd.getElementsByTagName("TITLE")[0]?.textContent || "N/A";
        table += `<tr><td>${artist}</td><td>${title}</td></tr>`;
      });
      document.getElementById("demo").innerHTML = table;
    };
    </script>

</body>
</html>
</body>
</html>
```

Version modular:

pagina02.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Pagina 02</title>
</head>

<body>
  <button id="btn">Ejecutar función</button>
  <table id="demo"></table>

  <script type="module" src="pagina02.js"></script>
</body>
</html>
```

pagina02.js > ...

```javascript
const loadXMLDoc = async () => {
    try {
      const response = await fetch('cd_catalog.xml');
      const text = await response.text();
      const parser = new DOMParser();
      const xmlDoc = parser.parseFromString(text, "application/xml");
      const cds = xmlDoc.getElementsByTagName("CD");
      myFunction(cds);
    } catch (error) {
      console.error('Error loading XML:', error);
    }
  };

  const myFunction = (cds) => {
    let table = "<tr><th>Artist</th><th>Title</th></tr>";
    Array.from(cds).forEach(cd => {
      const artist = cd.getElementsByTagName("ARTIST")[0]?.textContent || "N/A";
      const title = cd.getElementsByTagName("TITLE")[0]?.textContent || "N/A";
      table += `<tr><td>${artist}</td><td>${title}</td></tr>`;
    });
    document.getElementById("demo").innerHTML = table;
  };

  document.getElementById("btn").addEventListener("click", loadXMLDoc);
```

Versión no modular

- El código JavaScript está todo dentro del archivo HTML, lo que lo hace menos ordenado.
- El botón ejecuta la función directamente desde onclick, algo que ya no es tan recomendable.

- Aunque usa async/await y fetch, que son modernos, sigue estando todo pegado en un solo archivo.
- Si se quiere modificar algo del código, puede ser más complicado encontrarlo entre las etiquetas HTML.
- En la pestaña "Network" solo se ve que se carga el archivo .html, porque todo viene junto.
- El tiempo de carga fue de 9 ms en total, pero no se puede saber cuánto fue HTML y cuánto fue JS porque va todo junto.
- Funciona bien, pero no es tan limpio ni modular.

Versión modular

- El JavaScript está separado en su propio archivo (pagina02.js), lo que hace más fácil organizar el proyecto.
- El HTML está limpio, sin funciones directas ni onclick, y todo se maneja desde JS con addEventListener.
- Es más fácil de entender y mantener: si quieres editar algo, solo entras al JS sin tocar el HTML.
- En la pestaña "Network", se ven dos archivos separados (el .html y el .js) y puedes ver claramente cuánto tarda cada uno.
- El tiempo de carga fue de 6 ms para el HTML y 3 ms para el JS, repartido pero más controlado.
- Es más profesional y listo para que el proyecto crezca sin complicaciones.

Version no modular: **Página05**

Pagina05.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        img {
            box-shadow: 10px 10px 10px rgb(144, 196, 234);
        }
        figure {
            text-align: center;
            width: 300px;
            display: inline-block;
        }
    </style>
</head>
<body>
    <h1> Random Picture</h1>
    <!-- <figure> -->
        <!-- <img src="https://picsum.photos/id/1/200/200" alt="">
        <figcaption>
            Jhon
        </figcaption>
    </figure>
    <figure>
        <img src="https://picsum.photos/id/2/200/200" alt="">
        <figcaption>
            Jane
        </figcaption> -->
    <!-- </figure> -->

    <div id="demo"></div>

    <script>
        let ejemplo = document.getElementById("demo")
        console.log(ejemplo)

       function imagen(id) {

        const componente= "<figure>"+
            '<img src="https://picsum.photos/id/' +id+ '/200/200" alt="x">'+
            "<figcaption> John </figcaption>"+
            "</figure>"
        return componente


       }

       const Picture = (id) => {
            return `
                <figure>
                <img src="https://picsum.photos/id/${id}/200/200" alt="">
                <figcaption> Jhon</figcaption>
                `


       }

        // ejemplo.innerHTML = "hola" + imagen(1) + imagen(2)
        ejemplo.innerHTML = "hola" + Picture (1)

    </script>
</body>
</html>
```

Version modular:

Pagina05.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        img {
            box-shadow: 10px 10px 10px rgb(144, 196, 234);
        }
        figure {
            text-align: center;
            width: 300px;
            display: inline-block;
        }
    </style>
</head>
<body>
    <h1>Random Picture</h1>
    <div id="demo"></div>

    <script src="pagina05.js"></script>
</body>
</html>
```

pagina05.js

```javascript
function imagen(id) {
    return `
        <figure>
            <img src="https://picsum.photos/id/${id}/200/200" alt="">
            <figcaption> Jhon </figcaption>
        </figure>`;
}

const Picture = (id) => {
    return `
        <figure>
            <img src="https://picsum.photos/id/${id}/200/200" alt="">
            <figcaption> Jhon </figcaption>
        </figure>`;
}

let ejemplo = document.getElementById("demo");
console.log(ejemplo);

ejemplo.innerHTML = "Hola" + Picture(1);
```

Versión no modular

- El código JavaScript está directamente dentro del archivo HTML, usando la etiqueta <script>.
- El HTML contiene una lista (<ul>) vacía y un botón con onclick="cargarJSON()", lo cual mezcla lógica con estructura.
- Toda la lógica para cargar el archivo JSON (empleados.json) y llenar la lista está embebida en el HTML.
- Si el proyecto crece, esta forma dificulta organizar mejor los componentes.
- En la pestaña "Network" solo aparece pagina05.html, porque el código JS no está separado.
- No hay separación de responsabilidades, y modificar el código requiere revisar todo el HTML.

Versión modular

- Todo el código JavaScript fue extraído y colocado en un archivo separado: pagina05.js.
- El HTML ahora está más limpio: solo tiene el botón y la lista, sin funciones embebidas.
- En lugar de onclick, se usa addEventListener para vincular la función al botón desde JS, que es una práctica más moderna y recomendada.
- En "Network", ahora se ven dos archivos cargados: pagina05.html y pagina05.js, lo que permite medir tiempos y depurar de manera más eficiente.
- Mejora la organización del código, facilitando el mantenimiento y la escalabilidad.
- Listo para trabajar en equipo o en proyectos más grandes de forma profesional.

Version no modular: **Página06**

pagina06.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <style>
        img{
            box-shadow: 10px 10px 10px rgb(54, 238, 82);
            border-radius: 100%;
        }
        img.redonda {
            border-radius: 10% 40% 30% 70%;
        }
    </style>
</head>
<body>
    <div id="app"></div>


    <script>
        const Picture =(id,name) => {
            const url= `https://picsum.photos/id/${id}/200/200`
            return `
                <figure>
                    <img src="${url}" alt="" class="redonda" onClick="alert('${name}')">
                    <figcaption>${name}</figcaption>
                </figure>
            `
        };

        let x = document.getElementById('app').innerHTML = Picture(1, "Jhon") + Picture(2, "Jane")

        const lista = app.querySelectorAll("img")
        console.log(lista)


        for (let i= 0; i < lista.length; i++) {
            lista[i].addEventListener("click", () =>{
                lista[i].classList.toggle("redonda")
            })

        }

        function saludar(){
            alert("x")
        }

    </script>
</body>
</html>
```

Version modular:

pagina06.html > html > body > script

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <style>
        img {
            box-shadow: 10px 10px 10px rgb(54, 238, 82);
            border-radius: 100%;
        }
        img.redonda {
            border-radius: 10% 40% 30% 70%;
        }
    </style>
</head>
<body>
    <div id="app"></div>

    <script type="module" src="pagina06.js"></script>
</body>
</html>
```

pagina06.js > ...

```javascript
const Picture = (id, name) => {
    const url = `https://picsum.photos/id/${id}/200/200`;
    return `
        <figure>
            <img src="${url}" alt="" class="redonda" onClick="alert('${name}')">
            <figcaption>${name}</figcaption>
        </figure>
    `;
};

const loadImages = () => {
    document.getElementById('app').innerHTML = Picture(1, "Jhon") + Picture(2, "Jane");

    const lista = document.querySelectorAll("#app img");

    lista.forEach(img => {
        img.addEventListener("click", () => {
            img.classList.toggle("redonda");
        });
    });
};

document.addEventListener('DOMContentLoaded', loadImages);
```

Versión no modular

- Todo el código JavaScript está embebido directamente en el archivo HTML, lo que lo hace menos limpio y más difícil de mantener.
- Se usa onclick directamente en el HTML dentro del img, algo que ya no se recomienda porque mezcla estructura con comportamiento.
- La función Picture() está dentro del HTML, junto con el renderizado del contenido en el div #app, lo cual puede dificultar su reutilización.
- La manipulación del DOM (querySelectorAll, addEventListener, etc.) también está escrita en el mismo archivo.
- En la pestaña Network, solo se ve que se carga el .html, porque todo está unido en un solo archivo.
- Aunque funciona bien, no es una forma limpia ni profesional de estructurar el código.

Versión modular

- El archivo HTML está mucho más limpio, sin código JS embebido. Todo el comportamiento está en el archivo externo pagina06.js.
- Se elimina el onclick en el HTML. Ahora, los eventos se asignan desde JavaScript usando addEventListener, que es la práctica moderna.
- La función Picture() y la lógica de generación de contenido dinámico están bien encapsuladas y separadas del HTML.
- En la pestaña Network, puedes ver la carga del archivo .html y del .js por separado, lo que permite analizar tiempos de carga individualmente.
- Es más profesional, escalable y facilita la colaboración, ya que el HTML y JS están separados por responsabilidad.

Version no modular: **Página07**

pagina07.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario</title>
    <link rel="stylesheet" href="Formulario.css">

</head>
<!-- <body>
    <form action="https://google.com">
        <label for="">Usuario</label>
        <input type="text" id="user" name="user">
        <label for="">Contraseña</label>
        <input type="password" id="password" name="password">
        <input type="submit" value="Enviar" >
    </form> -->

<body>
    <form action="">
        <label for="">Usuario</label>
        <input type="text" id="user" name="x1">
        <label for="">Contraseña</label>
        <input type="password" id="password" name="x2">
        <!-- <input type="button" value="enviar" id ="enviar" onclick="mostrar()"> -->
        <input type="button" value="enviar" id ="enviar">
    </form>

    <script src="Formulario.js"></script>
</body>
</html>
```

Version no modular:

pagina07.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario</title>
    <link rel="stylesheet" href="Formulario.css">
</head>
<body>
    <form action="">
        <label for="">Usuario</label>
        <input type="text" id="user" name="x1">
        <label for="">Contraseña</label>
        <input type="password" id="password" name="x2">
        <input type="button" value="enviar" id="enviar">
    </form>

    <script type="module" src="pagina07.js"></script>
</body>
</html>
```

pagina07.js

```javascript
const mostrar = () => {
    const usuario = document.getElementById("user").value;
    const contraseña = document.getElementById("password").value;

    console.log(`Usuario: ${usuario}`);
    console.log(`Contraseña: ${contraseña}`);
};

const agregarEventoEnvio = () => {
    const botonEnviar = document.getElementById("enviar");

    botonEnviar.addEventListener("click", mostrar);
};

document.addEventListener('DOMContentLoaded', agregarEventoEnvio);
```

Versión no modular

- Todo el JavaScript está escrito dentro del HTML, lo cual hace que el archivo sea más largo y menos organizado.
- El botón de "enviar" tiene un onclick="mostrar()" (o estaba comentado), lo cual mezcla HTML y JS directamente.

- Las funciones como mostrar() están definidas dentro de <script>, dificultando la reutilización del código en otros archivos.
- En la pestaña Network, solo se ve que se carga el .html, y no hay separación clara del comportamiento.
- No es tan fácil modificar o escalar este formulario si luego se quiere validar más campos o usar una API.

Versión modular

- El archivo .html está limpio, sin ningún onclick ni funciones dentro del documento. Todo el JS está movido al archivo pagina07.js.
- Se usa addEventListener para detectar el clic del botón, lo que permite mantener el HTML más semántico y profesional.
- La lógica para obtener valores del formulario y mostrar mensajes está organizada en un solo lugar (el archivo JS).
- En Network, se puede ver por separado la carga del HTML y del JS, lo que facilita la depuración y control de carga.
- Es más limpio, organizado y profesional, listo para validaciones, almacenamiento en local o conexión con servidores.