Lenguajes de script de cliente.

Caso práctico

Ministerio de Educación (CC BY-NC-SA)

Una vez estudiado los lenguajes de marcas de los documentos web. **Ana** se pregunta si será posible hacer algo más interactiva la página web. Y que esa interacción sea gestionada en el propio sistema informático del cliente de manera que la respuesta sea lo más rápida posible y sin requerir la intervención de un servidor de internet. Hablando con **Juan** de su duda, este le comenta que ha escuchado que existen lenguajes que pueden ser ejecutados por el navegador del cliente sin intervención de ningún servidor. Estos lenguajes son capaces de transformar el contenido y aspecto de la página navegando por cada uno de sus elementos, organizando estos en estructura de árbol y pudiendo modificar su estructura y contenido.

Ministerio de Educación (CC BY-NC-SA)

La tendencia actual de las aplicaciones web es poner todas las tareas que sean posible en el lado del cliente. Con eso se ahorra en infraestructura y energía, y además la respuesta de la aplicación será más rápida y fluida. Siempre y cuando no se comprometa las seguridad y privacidad. Hay que pensar que tampoco pueden ser grandes bloques de código por que el lenguaje usado en el lado del cliente será interpretado con lo que tampoco será muy eficiente.

En esta unidad aprenderás un lenguaje de script del lado del cliente que es JavaScript (que es el más usado hoy en día) y a comprender como se puede navegar por la estructura DOM de una página web y como se puede modificar dicha estructura y sus valores para cambiar la representación de la página web en el navegador del usuario.

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y DEPORTES

Ministerio de Educación, Formación Profesional y Deportes (Dominio público)

1.- Introducción.

Caso práctico

Ministerio de Educación (CC BY-NC-SA)

Ana ha investigado sobre los lenguajes de script. Y ha descubierto que están divididos en dos grandes categorías los que se ejecutarán en el lado del servidor (Por ejemplo: PHP, ASP, JSP), es decir el código esta en la máquina del servidor donde se ejecutará y el resultado se hará llegar de alguna forma al cliente. Y los que se ejecutarán en el lado del cliente, siendo necesario que el cliente primero se descargue el código para después ejecutarlo y mostrar los resultados al usuario.

Cuando un usuario abre su **navegador** de internet y consulta una **página web**. El navegador lo que hace es ir a internet para buscar al **servidor** que tendrá esa página web y enviarle la consulta. El servidor recibe la consulta y tratará de **resolverla**. Cuando el servidor web encuentra la página a la que se refiere la consulta puede ocurrir que esa página sea de dos tipos:

- **Pasiva**: El contenido de la página se encuentra almacenado directamente en un archivo. El contenido de la página no varía con ningún parámetro, ni con el tiempo si no es porque se modifique directamente el archivo que almacena la página. Se dice que es un contenido estático.
- **Activa**: El contenido de la página es el resultado de ejecutar un script (programa escrito en un lenguaje de script que será interpretado por el servidor). El script en ningún momento debe enviarse como respuesta a la consulta directamente. El resultado de la interpretación de ese script por parte del servidor, que será un documento web, será lo que se envíe como respuesta a la consulta.

Cuando el servidor **responde** a la consulta envía al cliente un **documento web**. Este documento web puede contener **scripts** que serán ejecutados esta vez en el sistema informático del cliente. Y el **cliente** tratará de representar el documento y de **interpretar** los scripts que contenga, así como de recibir los posibles **eventos**, provocados por el usuario o por un cambio en el propio sistema informático, que activen algún script del propio documento web.

Los lenguajes de script del lado del cliente que podemos encontrar son:

- **JavaScript**: Es el más usado, y en el que se basan muchos framework como jQuery, React (Facebook), Vue, Angular (Google), etc.
- **VbScript**: Fue creado en 1996 por Microsoft para Internet Explorer pero cada vez se usa menos. Microsoft también saco **JScript** que es su implementación de **JavaScript** para sus navegadores.

El lenguaje de script del lado del cliente que hoy conocemos como JavaScript nació como Mocha desarrollado para Netscape. Después se cambio el nombre a LilveScript y finalmente

JavaScript (Nombre registrado por Oracle). Luego el lenguaje fue adoptado por ECMA (European Computer Manufacturers Association de Ginebra) y llamado **ECMAScript**.

En cuanto a la versiones que han existido de JavaScript (ECMAScript) podemos resumirlas en las siguientes:

- 1 en **1997**, primera versión de JavaScript. Introdujo conceptos como variables, funciones y objetos.
- 2 en 1998, correcciones menores y mejoras en la especificación.
- 3 en 1999, con muchas mejoras importantes, incluyendo soporte para expresiones regulares, manejo de excepciones mejorado, etc.
- 5 en 2009, con características como métodos de vectores adicionales, propiedades getters/setters, y más mejoras en el manejo de excepciones.
- 6 en 2015, fue un gran salto, nuevas clases, operadores, funciones asíncronas, etc. Se continua desarrollando en la actualidad.

Al rededor de JavaScript tenemos muchas tecnologías como los ya conocidos HTML y CSS, pero además podemos encontrar:

- DOM (Modelo de Objetos del Documento): que básicamente es un interfaz para navegar por un documento web.
- AJAX (Asynchronous JavaScript And XML): tecnología para consultar al servidor desde JavaScript de manera asincróna alguna pequeña información y poder incorporarla en la representación actual de la página.

2.- Conceptos básicos.

Caso práctico

Ministerio de Educación (CC BY-NC-SA)

Ana esta muy interesada en este lenguaje que permite de una manera sencilla programar páginas para hacer pequeñas tareas y dotar a la página web de alguna respuesta rápida a la interacción del usuario. Sin tener que volver a consultar con el servidor descargando a este último de la ejecución de ese código. Se da cuenta de que esta es una manera muy interesante de el servidor tenga menos carga de trabajo delegando algunas tareas al sistema informático del cliente. Lo que será un ahorra para la empresa y hará que la interacción con el usuario sera más rápida y flulida.

Una página web básicamente estará compuesta por un contenido (**HTML** <https://www.w3schools.com/html/>), un aspecto (**CSS** <https://www.w3schools.com/CSS/>) y un componente dínamico (**JavaScript** <https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript>).

JavaScript es un lenguaje interpretado de alto nivel dialecto del estándar **ECMAScript**. Es un lenguaje fuertemente orientado a objetos, basado en prototipos (y no en clases). Es multiparadigma y admite estilos de programación imperativo, funcional y basado en eventos. Tiene interfaces de programación de aplicaciones (API) para trabajar con texto, fechas, expresiones regulares, estructuras de datos estándar y el modelo de objetos de documento (**DOM**). Permite modificar de forma dinámica el contenido y aspecto de una página web.

Para abrir una consola de JavaScript en el navegdor pulsar Ctrl+Shift+k (Cmd+Opt+K)

A continuación se exponen varios trozos de código a modo de ejemplo de la sintaxis básica del lenguaje JavaScript:

- Declaración de variables y vectores:

```
...
var nombre = "Luzia"; // declara una variable y opcionalmente la inicia con un valor.
let edad = 25;         // declara una variable de ámbito local a un bloque (no existirá fuera).
let coche = true;     // declara una variable booleana.
const PI = 3.1416;     // declara una variable constante que no se puede modificar y ámbito local.
const idiomas = ["Español", "Inglés", "Francés"];  // declara un vector con 3 valores.
// Es aconsejable declarar una variable antes de usarla.
```

```
console.log("edad="+edad);    //  imprime  en  consola  la  concatenación  de
"edad=" y el valor de la variable "edad".
...
```

- Condicionales:

```
...
if (edad >= 18) {  // Los operadores ( == sin mirar tipo, === mira tipo, !=
... lógicos &&,||,! )
        console.log("Eres mayor de edad");
} else {
        console.log("Eres menor de edad");
}
switch (variable) {
        case valor1: console.log("Es valor1"); break;
        case valor2: console.log("Es valor2"); break;
        default: console.log("Cualquier otro"); break;
}
...
```

- Bucles:

```
...
for (let i = 0; i < 5; i++) {    // inicialmente i vale 0 e incrementará de
uno en uno hasta menor que 5
        console.log("El valor de i es: " + i);
}
while (edad < 30) {
        edad++;
}
...
```

- Funciones:

```
...
function saludar(nombre) {
        return "¡Hola, " + nombre + "!";
}
console.log(saludar("Juan"));
...
```

3.- DOM.

Caso práctico

Ministerio de Educación (CC BY-NC-SA)

Ana quiere aprender a modificar el aspecto y el contenido de una página web desde el lenguaje JavaScript navegando por la estructura de un documento HTML (DOM). Ver como se puede hacer referencia a un elemento en concreto de la página web y modificar su aspecto o su contenido. O incluso como añadir un nuevo elemento o borrarlo de la página web. Se da cuenta de que lo primero que hay que comprender es que la página se almacena con una estructura de árbol. Donde habrá un raíz (del documento) y de este colgaría un nodo que representaría la etiqueta "html". Y de este último colgarían 2 nodos uno para el "head", y otro para el "body". Y del "body" colgaría un parrrafo ("p")... y así hasta completar el documento HTML.

El **DOM** (Modelo de Objetos del Documento) es una **representación** en forma de **árbol**, en la **memoria** del sistema informático del cliente, del documento **HTML** que compone la página web que muestra el navegador de internet del usuario. El navegador de internet del usuario al mostrar la página web va creando una estructura en árbol con todos los elementos de la página web.

Por ejemplo:

- Documento
  - html
    - head
      - title
    - body
      - h1
      - p
      - p
        - a (un enlace dentro del párrafo)
        - img

Esta estructura en árbol que también contendrá la información de aspecto del **CSS**, es la que permitirá **navegar** por cada elemento a un lenguaje de script del lado del cliente como **JavaScript**. Con JavaScript podremos seleccionar un elemento de la página y modificarlo. Y esta **modificación** se verá reflejada en la **representación** que muestra el navegador de internet del usuario. El código JavaScript podrá ser ejecutado al cargar la página web o cuando se produzca algún otro evento producido por el usuario, como pulsar un botón, o por el propio sistema, como terminar de cargar la página.

Seleccionar elemento:

Para seleccionar un elemento del DOM desde JavaScript podemos usar:

- **`document.getElementById("idPuestoEnHTML")`** : Selecciona un elemento del documento HTML identificado con el atributo "id" igual a "idPuestoEnHTML".
- **`document.getElementByClassName(".clase")`** : Selecciona todos los elementos con el atributo "class" igual a "clase" del docuemnto HTML.
- **`document.getElementByTagName("p")`** : Selecciona todos los párrafos del docuemnto HTML.
- **`document.querySelector("p")`** : Selecciona el primer elemento párrafo del docuemnto HTML.
- **`document.querySelectorAll(".clase")`** : Selecciona todos los elementos con el atributo "class" igual a "clase" del docuemnto HTML.

Eventos:

- Pulsar un botón:

```
<script>
    function funcion() {
        document.getElementById("parrafo").innerHTML = "Modifica contenido";
    }
</script>
...
<button type="button" onclick="funcion()">Pulsa aquí</button>
```

- Terminar de cargar la página: "**`window.onload = function()`**"

Trabajar con los elementos:

- **`parrafo.style.color = "blue"`** : Modificar el aspecto, el color de la letra para este caso, al color azul del elemento "parrafo".
- **`parrafo.style.textContent = "Hola mungo"`** : Modificar el texto contenido dentro del elemento "parrafo".
- **`parrafo.style.innerHtml = "<B>Hola</B> mundo"`** : Modificar el HTML contenido dentro del elemento "parrafo".
- **`imagen.setAttribute("src", "otra.png")`** : Cambia el atributo "src", es decir, la imagen que se mostraba en el elemento "imagen".
- Crear un nuevo elemento:

```
const nuevo = document.createElement("p");
nuevo.textContent = "Nuevo párrafo.";
document.body.appendChild(nuevo);
```

- **`parrafo.remove`** : borrar el elemento "parrafo".

Normalmente las funciones para **fijar un valor** (de contenido o de aspecto) también sirven para obtenerlo. Es decir si están a la **izquierda** del igual en una sentencia de asignación, reciben un valor, pues lo que haremos es fijar el valor. Si

están a la **derecha** del igual en una sentencia de asignación servirán para **obtener el valor** actual del contenido o aspecto.

Ejercicio Resuelto

Crear un script en JavaScript que se ejecute una vez haya finalizado de cargar la página web y modifique el contenido y el color de las letras de un párrafo.

Mostrar retroalimentación

```html
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8" />
        <title>Equipos</title>
        <script>
            window.onload = function() {
                const parrafo = document.getElementById("saluda");
                parrafo.textContent = "¡Hola, mundo!";
                parrafo.style.color = "red";
            }
        </script>
    </head>
    <body>
        <p id="saluda">aaaadios</p>
    </body>
</html>
```