

Critical rendering path

Se refiere al proceso que un navegador web sigue para renderizar una página web en la pantalla del usuario

- **Recuperación de recursos** : Solicita recursos, html, css, javascript, img, fuentes, etc..
 - **Análisis del HTML**: Analiza el html para construir el DOM
 - **Construcción del árbol de estilos (CSSOM)**
 - **Combinación del DOM y el CSSOM**
 - **Layout (Diseño)**: El navegador calcula la geometría exacta de cada elemento determinando su posición y tamaño en relación con la ventana gráfica del navegador
 - **Pintura (Painting)**: El navegador pinta los píxeles en la pantalla.
 - **Composición (Composition)**: El navegador combina las capas de pintura en una sola imagen que se muestra en la pantalla.
- *** **JavaScript** se ejecuta después de que se haya construido el DOM y el CSSOM, y su ubicación en la página y el uso de atributos específicos pueden influir en cuándo se carga y ejecuta, así como en cómo afecta al proceso de renderización de la página web.

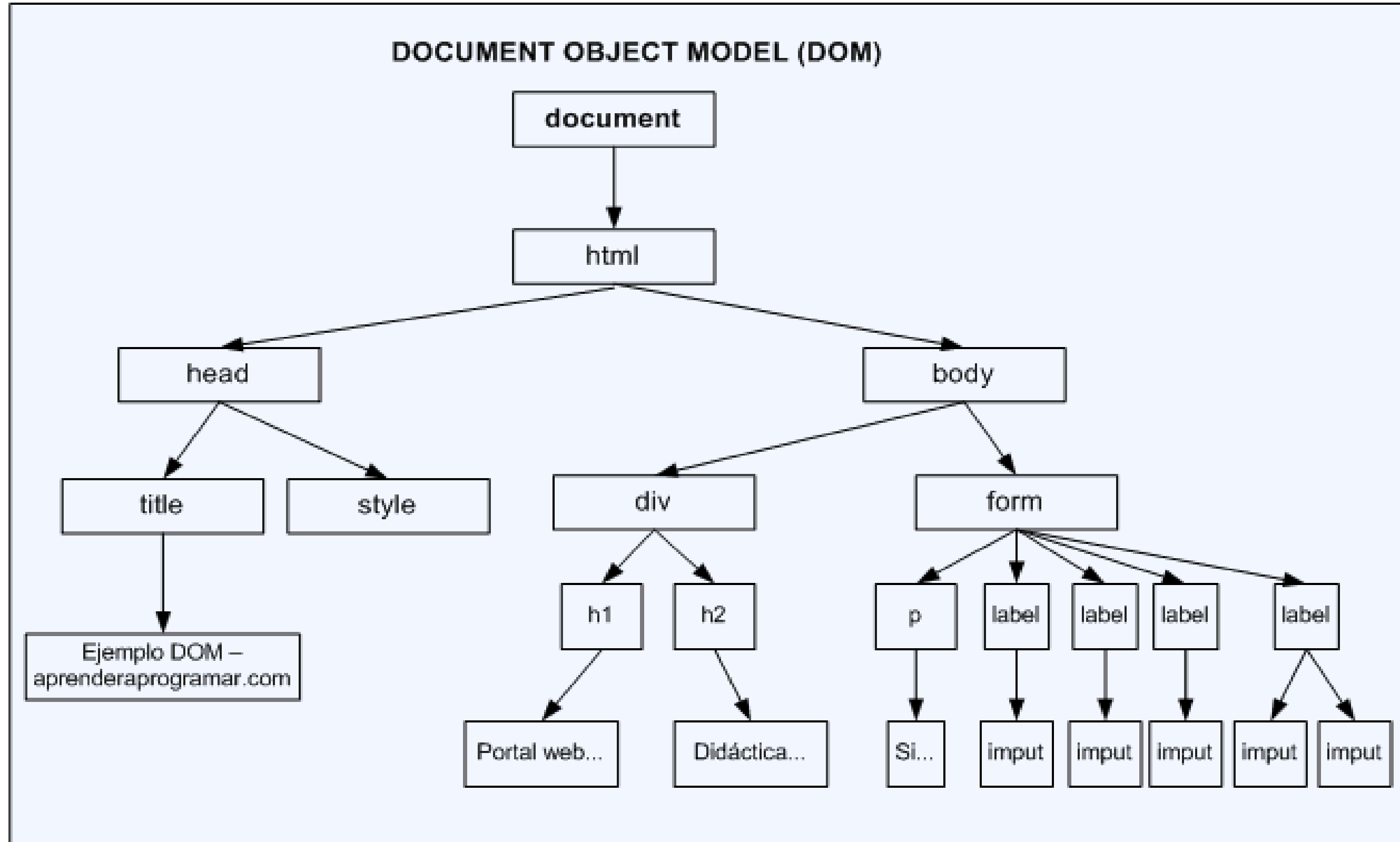
EL DOM

Las siglas **DOM** significan Document Object Model, o lo que es lo mismo, la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol **DOM**.

En Javascript, cuando nos referimos al **DOM** nos referimos a esta estructura, que podemos modificar de forma dinámica desde Javascript, añadiendo nuevas etiquetas, modificando o eliminando otras, cambiando sus atributos HTML, añadiendo clases, cambiando el contenido de texto, etc...

En su interior pueden existir varios tipos de elementos, pero principalmente serán: **ELEMENT** (Etiqueta) o **NODE** (Unidad básica => una etiqueta o nodo texto)

EL DOM



FUNCIONES DEL DOM

getElementsByTagName("etiqueta"): Obtiene todos los elementos de la página HTML cuya etiqueta sea igual que el parámetro que se le pasa a la función

getElementsByTagName("nombre"): Es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo name sea igual al parámetro proporcionado

getElementById("identificador"): Devuelve el elemento HTML cuyo atributo id coincide con el parámetro indicado en la función.

getElementsByClassName("identificador"): Devuelve el elemento HTML cuyo atributo id coincide con el parámetro indicado en la función.

querySelector("selector"): Devuelve el primer elemento que cumple la condición. Si no existe el elemento, el valor retornado es null.

querySelectorAll("selector"): Devuelve un objeto con los elementos que coincidan con el selector.

elemento.children: devuelve una lista de nodos (nodeList) hijos de un elemento.

elemento.parentNode: Devuelve el nodo padre de un nodo.

elemento.nextElementSibling: devuelve el hermano siguiente del elemento

elemento.previousElementSibling: devuelve el hermano anterior del elemento

Resultados:

* **HtmlCollection** : Colección de elementos html. Se obtiene principalmente a través propiedades como document.forms, document.images, document.links, y métodos document.getElementsByTagName()

* **NodeList:** Colección de nodos (etiquetas html, nodos texto, comentarios, etc...) Se obtiene mediante métodos generales del DOM como **querySelectorAll()**, **childNodes**, **parentNode.childNodes**, y otras operaciones que devuelven nodos relacionados con el DOM

Resultados:

Tanto `HtmlCollection` como `NodeList` no se consideran Arrays y por lo tanto, no admiten propiedades y métodos de un Array.

Solución:

Convertir tanto un **NodeList** como un **HTMLCollection** en un array utilizando **`Array.from()`** o el (Spread Operator = **operador de propagación**)

Una vez que se convierten en arrays, puedes usar métodos de array como `forEach()`

NOTA: Se puede utilizar **`for..of`** para recorrer tanto **HTMLCollection** como en **NodeList**

Acceso y modificación de propiedades:

elemento.innerHTML: Accede o modifica el contenido HTML de un elemento.

elemento.textContent: Accede o modifica el contenido de texto de un elemento.

elemento.getAttribute('atributo'); Obtiene el valor de un atributo.

elemento.setAttribute('atributo', 'valor'); Establece el valor de un atributo.

Modificación de estilos:

elemento.style.propiedad = 'valor'; Modifica el estilo de un elemento.

elemento.classList.add('clase'); Agrega una clase CSS a un elemento.

Manipulación de clases:

elemento.classList.remove('clase'): Elimina una clase CSS de un elemento

elemento.classList.remove('clase'): Elimina una clase CSS de un elemento

elemento.classList.toggle('clase'): Agrega o quita una clase CSS según su estado

elemento.classList.contains('clase'): Devuelve true/false dependiendo si el elemento tiene la clase o no.

Creación y eliminación de elementos:

document.createElement('etiqueta'): Crea un nuevo elemento.

elemento.appendChild(nuevoElemento): Agrega un elemento como hijo de otro.

elemento.removeChild(elementoHijo): Elimina un elemento hijo.

Ejemplo:

// Convertir un NodeList en un array y usar forEach

```
var nodeList = document.querySelectorAll("p");
```

```
var nodeListArray = Array.from(nodeList);
```

```
nodeListArray.forEach(function(node) {
```

```
    console.log(node.textContent);
```

```
});
```

// Convertir un HTMLCollection en un array y usar forEach

```
var htmlCollection = document.getElementsByClassName("mi-clase");
```

```
var htmlCollectionArray = [...htmlCollection];
```

```
htmlCollectionArray.forEach(function(element) {
```

```
    console.log(element.textContent);
```

```
});
```

Para insertar elementos adyacentes en el DOM utilizando JavaScript, puedes utilizar las siguientes funciones y métodos:

1.insertAdjacentHTML():

Puedes especificar la posición de inserción utilizando los siguientes valores como argumento:

- **beforebegin**: Antes del elemento en sí
- **afterbegin**: Dentro del elemento, al comienzo.
- **beforeend**: Dentro del elemento, al final.
- **afterend**: Después del elemento.

Ejemplo:

```
const element = document.getElementById("myElement");  
element.insertAdjacentHTML("beforebegin", "<p>Antes del elemento</p>");
```

2. insertAdjacentElement():

Similar a `insertAdjacentHTML()`, esta función te permite insertar un elemento adyacente a otro elemento en el DOM. Puedes usar un elemento HTML como argumento.

Ejemplo:

```
const element = document.getElementById("myElement");  
const newElement = document.createElement("p");  
newElement.textContent = "Nuevo párrafo";  
element.insertAdjacentElement("beforebegin", newElement);
```

3. insertAdjacentText():

Esta función te permite insertar texto adyacente a un elemento en el DOM en una posición específica.

Ejemplo:

```
const element = document.getElementById("myElement");  
element.insertAdjacentText("beforebegin", "Texto antes del elemento");
```

Asegúrate de reemplazar `"myElement"` con el ID del elemento al que deseas insertar contenido y selecciona la posición de inserción adecuada según tus necesidades. Estos métodos son útiles cuando necesitas agregar elementos o texto adyacente a un elemento existente en el DOM de manera fácil y flexible.

LOS EVENTOS DEL DOM

El navegador desencadena muchos eventos. Los más comunes:

- **Eventos del ratón (MouseEvent):** mousedown, mouseup, click, dblclick, mousemove, mouseover, mousewheel, mouseout, contextmenu
- **Eventos del teclado (KeyboardEvent):** keydown, keypress, keyup
- **Eventos de formularios:** focus, blur, change, submit
- **Eventos de la ventana:** scroll, resize, load, unload

EVENTOS DEL **MOUSE**

Nombre del evento	Descripción del evento
<i>click/dblclick</i>	Cuando hacemos click/doble click de ratón sobre un elemento de la página. (mouse down / mouse down en el mismo elemento). Botón izquierdo.
<i>mouseenter / mouseleave</i>	Cuando el cursor del ratón entra o sale de un elemento.
<i>mouseover / mouseout</i>	Cuando el cursor del ratón entra o sale de un elemento o de alguno de sus hijos.
<i>mousedown / mouseup</i>	Cuando presiono / libero cualquier botón del ratón, da igual que no sea el mismo elemento.
<i>drag / dragstart / dragend / dragenter / dragleave / dragover</i>	Eventos relacionado con el movimiento de elementos “arrastrables” (draggable = true) a lo largo de la pantalla.
<i>drop</i>	Cuando “libero” el elemento “arrastrable” (draggable) en su destino.

EVENTOS DE **TECLADO**

Nombre del evento	Descripción del evento
<i>keypress / keyup</i>	Cuando el usuario presiona/libera un tecla .
<i>keydown</i>	Cuando el usuario mantiene una tecla presionada .

EVENTOS DE FORMULARIO

Nombre del evento	Descripción del evento
<i>focus / blur</i>	Cuando el elemento obtiene/pierde el foco .
<i>submit/reset</i>	Cuando se envían o se borran los datos de un formulario.
<i>change</i>	Cuando el contenido/selección/estado de input/select/textarea cambia.
<i>input</i>	Cuando cambia el contenido de un input.
<i>select</i>	Cuando selecciono algún texto de un input/textarea .

EVENTOS DE NAVEGADOR

Nombre del evento	Descripción del evento
<i>load</i>	Cuando la página se ha acabado de cargar (si es asocia a <body>).
<i>unload</i>	Cuando cerramos una página o nos dirigimos a otra (si se asocia a <body>)
<i>resize</i>	Cuando se redimensiona la ventana del navegador.
<i>scroll</i>	Se activa cuando se realiza un desplazamiento (scroll) en la página.

FORMAS DE CAPTURAR EVENTOS

- Desde el **HTML** (**inline** - no recomendado)
- Desde el Script con **Método Propio**.
- Desde el Script con **addEventListener(...)** /**removeEventListener()**.

INLINE - DESDE HTML

```
<div onclick="miFuncion(event)"></div>
```

```
<script>
```

```
    function miFuncion(e) {  
        console.log("Has hecho click");  
    }
```

```
</script>
```

CON PROPIEDADES PROPIAS

```
<div id="caja"></div>
```

```
<script>
```

```
    function miFuncion(e){  
        console.log("Hashecho click");  
    }
```

```
    let caja = document.getElementById("caja");  
    caja.onclick = miFuncion
```

```
</script>
```

CON LA FUNCIÓN **addEventListener**

```
<div id="caja"></div>
```

```
<script>
```

```
    function miFuncion(e){  
        console.log("Hashecho click");  
    }
```

```
    let caja = document.getElementById("caja");  
    caja.addEventListener("click",miFuncion);
```

```
</script>
```

CON LA FUNCIÓN `removeEventListener()`

```
<div id="caja"></div>
```

```
<script>
```

```
let caja = document.getElementById("caja");
```

```
// Para deshabilitar un manejador de eventos
```

```
caja.removeEventListener("click",miFuncion);
```

```
</script>
```

EL OBJETO **EVENT** / Algunas propiedad y métodos

.preventDefault()	Evita el comportamiento por defecto.
.stopPropagation() / .stopImmediatePropagation()	Para la propagación / Para también otros listeners que puedan estar capturando dicho evento
.target / .currentTarget	Elemento que lanzó el evento / cuyos manejadores lanzaron el evento
.pageX / .pageY	Posición X/Y de donde se produjo el evento
...	

EJEMPLOS

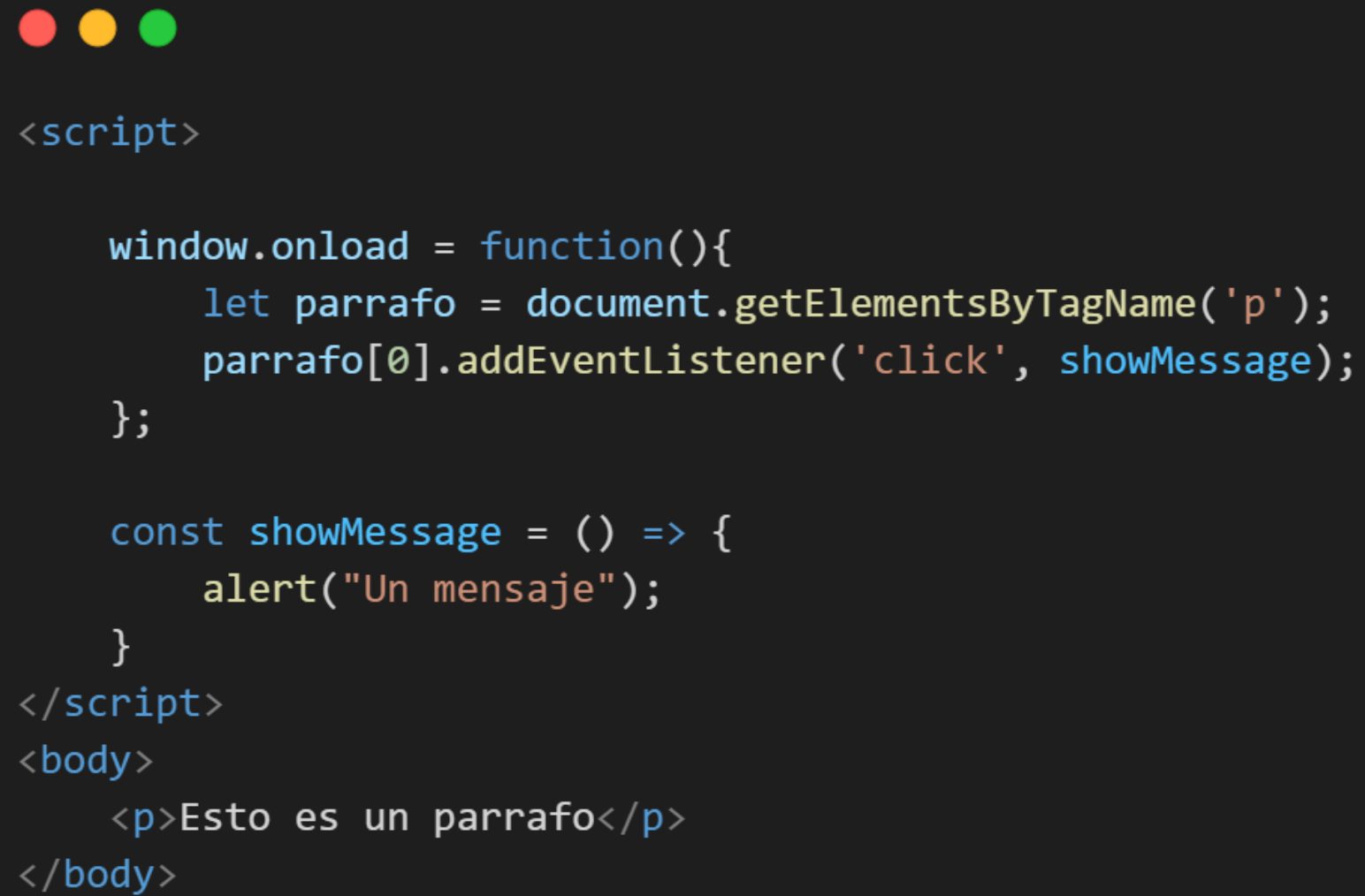
El evento **window.onload** se ejecuta cuanto toda la página se haya cargado, incluidos estilos y scripts.



```
<script>
  function load() {
    alert("Se ha cargado la página completamente");
  }
  window.onload = load;
</script>
```


EJEMPLOS

Asignar un evento por **TAG**



```
<script>

  window.onload = function(){
    let parrafo = document.getElementsByTagName('p');
    parrafo[0].addEventListener('click', showMessage);
  };

  const showMessage = () => {
    alert("Un mensaje");
  }
</script>
<body>
  <p>Esto es un parrafo</p>
</body>
```

EJEMPLOS

Asignar un evento por **ID**

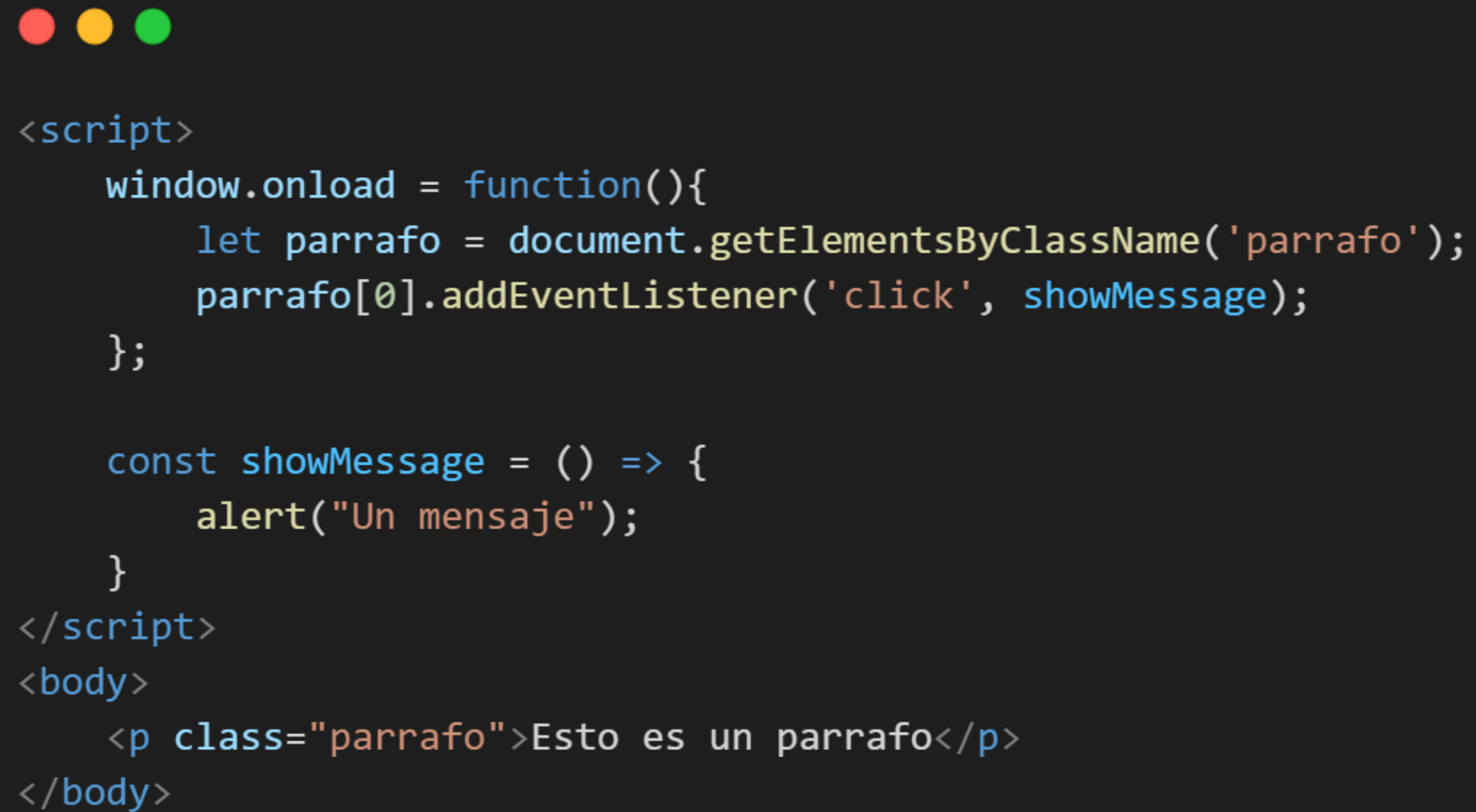


```
<script>
  window.onload = function(){
    let parrafo = document.getElementById('parrafo');
    parrafo.addEventListener('click', showMessage);
  };

  const showMessage = () => {
    alert("Un mensaje");
  }
</script>
<body>
  <p id="parrafo">Esto es un parrafo</p>
</body>
```

EJEMPLOS

Asignar un evento por **CLASE**




```
<script>
  window.onload = function(){
    let parrafo = document.getElementsByClassName('parrafo');
    parrafo[0].addEventListener('click', showMessage);
  };

  const showMessage = () => {
    alert("Un mensaje");
  }
</script>
<body>
  <p class="parrafo">Esto es un parrafo</p>
</body>
```

EJEMPLOS

Cambiar el texto de un **TAG**



```
<script>
  window.onload = function(){
    let parrafo = document.getElementsByClassName('parrafo');
    parrafo[0].addEventListener('click', changeMessage);
  };

  const changeMessage = function () {
    this.innerText = 'Un mensaje';
  }
</script>
<body>
  <p class="parrafo">Esto es un parrafo</p>
</body>
```

EJEMPLOS

Modificar código **HTML**



```
<script>
  window.onload = function(){
    let container = document.getElementsByClassName('container');
    container[0].addEventListener('click', changeMessage);
  };
  const changeMessage = function () {
    this.innerHTML = '<h1>Un mensaje</h1>';
  }
</script>
<body>
  <div class="container">
    <p class="parrafo">Esto es un parrafo</p>
  </div>
</body>
```

EJEMPLOS

Añadir código **HTML**



```
<script>
  window.onload = function(){
    let container = document.getElementsByClassName('container');
    container[0].addEventListener('click', addMessage);
  };
  const addMessage = function () {
    let h1 = document.createElement("h1");
    h1.innerText = "Un mensaje";
    this.append(h1);
  }
</script>
<body>
  <div class="container">
    <p class="parrafo">Esto es un parrafo</p>
  </div>
</body>
```

EJEMPLOS

Añadir código **HTML**



```
<script>
  window.onload = function(){
    let container = document.getElementsByClassName('container');
    container[0].addEventListener('click', addMessage);
  };
  const addMessage = function () {
    let div = document.createElement("div");
    let h1 = document.createElement("h1");
    h1.innerText = "Un mensaje";
    div.append(h1);
    let h2 = document.createElement("h2");
    h2.innerText = "Un mensaje más pequeño";
    div.append(h2);
    this.append(div);
  }
</script>
```

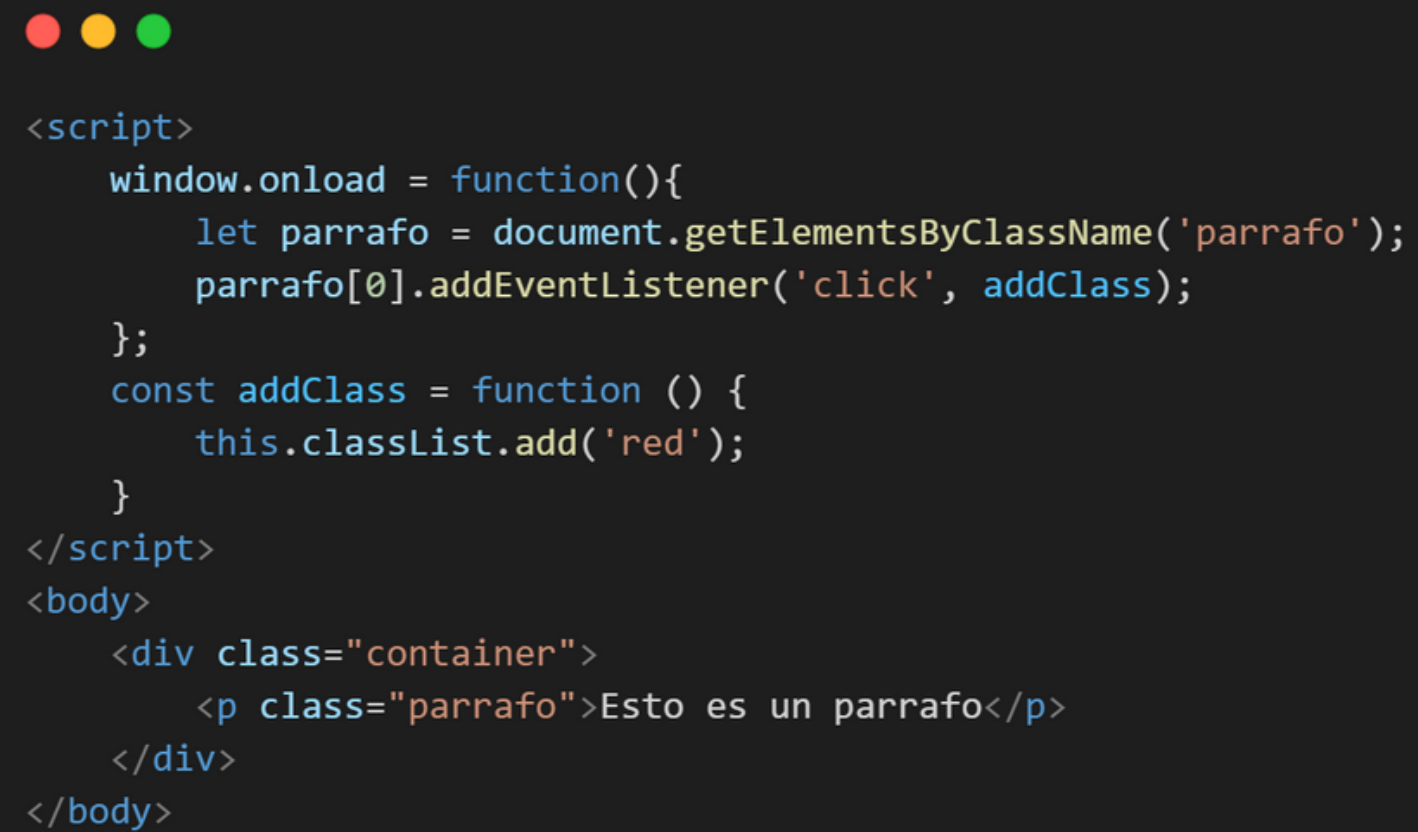
EJEMPLOS

Cambiar el estilo de un elemento

```
<script>
  window.onload = function(){
    let parrafo = document.getElementsByClassName('parrafo');
    parrafo[0].addEventListener('click', changeColor);
  };
  const changeColor = function () {
    this.style.color = 'red';
  }
</script>
<body>
  <div class="container">
    <p class="parrafo">Esto es un parrafo</p>
  </div>
</body>
```


EJEMPLOS


Añadir una clase



```
<script>
  window.onload = function(){
    let parrafo = document.getElementsByClassName('parrafo');
    parrafo[0].addEventListener('click', addClass);
  };
  const addClass = function () {
    this.classList.add('red');
  }
</script>
<body>
  <div class="container">
    <p class="parrafo">Esto es un parrafo</p>
  </div>
</body>
```

EJEMPLOS

Quitar una clase



```
<script>
  window.onload = function(){
    let parrafo = document.getElementsByClassName('parrafo');
    parrafo[0].addEventListener('click', removeClass);
  };
  const removeClass = function () {
    this.classList.remove('red');
  }
</script>
<body>
  <div class="container">
    <p class="parrafo red">Esto es un parrafo</p>
  </div>
</body>
```