DOM JS

Document Object Model

DEV.F:

DESARROLLAMOS(PERSONAS);

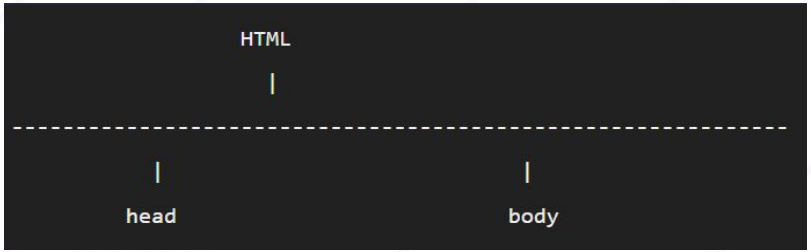
¿Qué es el DOM?

Es la estructura de objetos que genera el navegador en memoria para cada uno de los elementos de la página. Con ello nos referimos al DOM habitualmente con el nombre de jerarquía de objetos, porque realmente es una estructura jerárquica donde existen varios objetos y unos dependen de otros.

Supongamos que nuestra página web se forma del siguiente código:

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

La representación de esta página tan sencilla sería tan simple como se ve a continuación:



DEV.F

DOM y el W3C

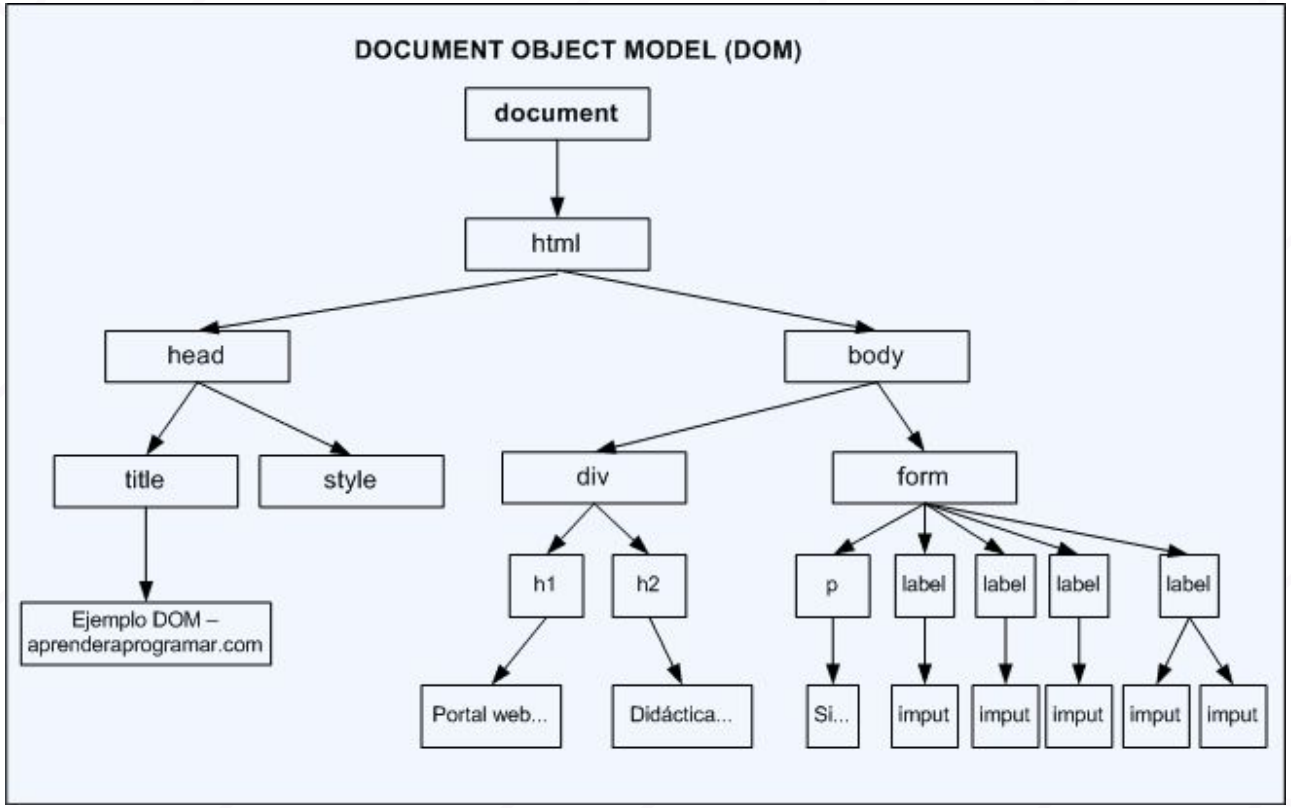
El DOM está definido y administrado por el W3C, por lo que los distintos navegadores simplemente aplican las especificaciones del World Wide Web Consortium, para dar soporte al DOM en sus aplicaciones.

Uno de los objetivos principales del W3C es generar estándares: documentos donde se definen las sintaxis de lenguajes y protocolos que intervienen en el desarrollo de internet

Ejemplo de representación DOM

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo DOM - aprenderaprogramar.com</title><meta charset="utf-8">
<style type="text/css">
body {background-color:yellow; font-family: sans-serif;}
label {color: maroon; display:block; padding:5px;}
</style>
</head>
<body>
<div id="cabecera">
<h1>Portal web aprenderaprogramar.com</h1>
<h2>Didáctica y divulgación de la programación</h2>
</div>
<!-- Formulario de contacto -->
<form name ="formularioContacto" class="formularioTipo1" method="get" action="accion.html">
<p>Si quieres contactar con nosotros envíanos este formulario relleno:</p>
<label for="nombre"><span>Nombre:</span> <input id="nombre" type="text" name="nombre" /></label>
<label for="apellidos"><span>Apellidos:</span> <input id="apellidos" type="text" name="apellidos" /></label>
<label for="email"><span>Correo electrónico:</span> <input id="email" type="text" name="email" /></label>
<label>
<input type="submit" value="Enviar">
<input type="reset" value="Cancelar">
</label>
</form>
</body>
</html>
```

La representación del documento conforme al estándar del DOM sería esta:



Herramientas nativas de JS

Métodos de búsqueda

getElementById().

El primer método, .getElementById(id) busca un elemento HTML con el id especificado en id por parámetro.

getElementsByClassName().

El método .getElementsByClassName(class) permite buscar los elementos con la clase especificada en class.

Métodos modernos

querySelector().

Devuelve el primer elemento que encuentra que encaja con el selector CSS suministrado en selector. Al igual que su «equivalente» .getElementById().

querySelectorAll().

Realiza una búsqueda de elementos como lo hace el anterior, sólo que como podremos intuir por ese All(), devuelve un array con todos los elementos que coinciden con el selector.

Tipos de datos importantes

Para simplificar, los ejemplos de sintaxis en esta API se refieren a nodos como elements, a una lista de nodos como nodeLists (o simples elementos) y a nodos de atributo como attributes.

- document.getElementById(id)
- element.getElementsByTagName(name)
- document.createElement(name)
- parentNode.appendChild(node)
- element.innerHTML
- element.style.left
- element.setAttribute
- element.element.getAttribute
- element.addEventListener
- window.content
- window.onload
- window.dump
- window.scrollTo

Conclusión sobre el DOM

- **El DOM es una utilidad esencial de los navegadores para el trabajo con la página web.**
- **Trabajar con el DOM se ha hecho más fácil, ya que todos los navegadores actualmente responden a un API muy consistente de funcionalidades sobre los elementos de la página.**
- **El trabajar con el DOM directamente con el lenguaje Javascript es posible, aunque además muchos desarrolladores prefieren usar librerías que simplifican la labor. Incluso, últimamente se estilan librerías más avanzadas, como React o Lit que permiten manipular la página por medio de componentes encapsulados.**