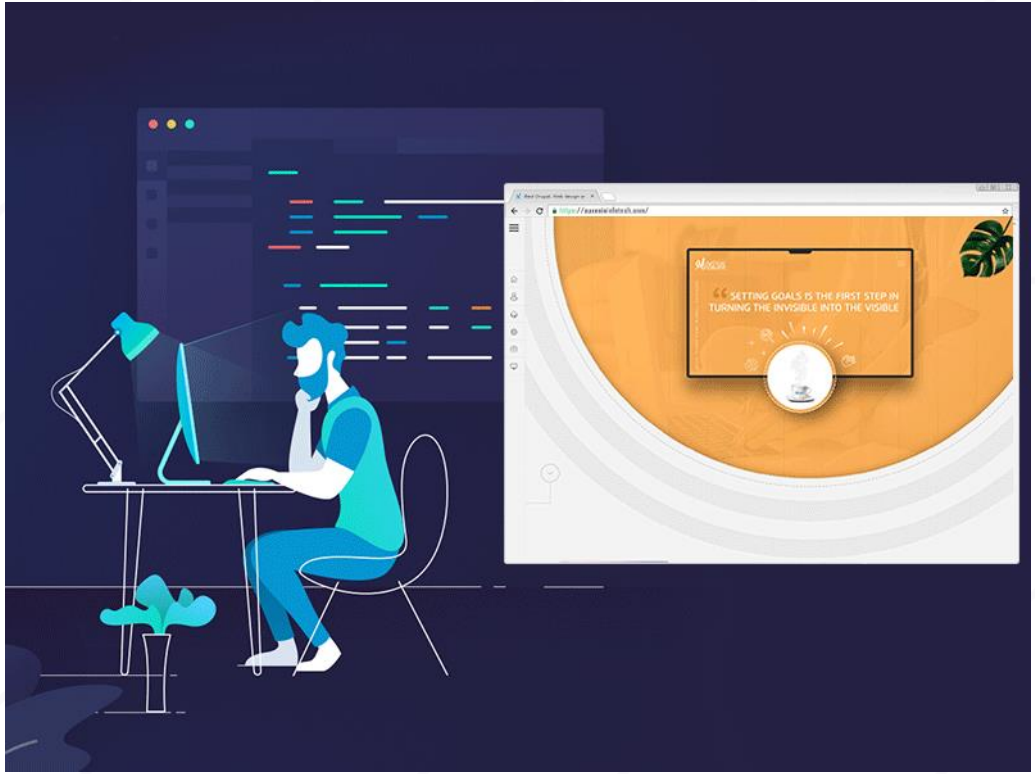
Document Object Model

DEV.F
DESARROLLAMOS(PERSONAS);

JavaScript

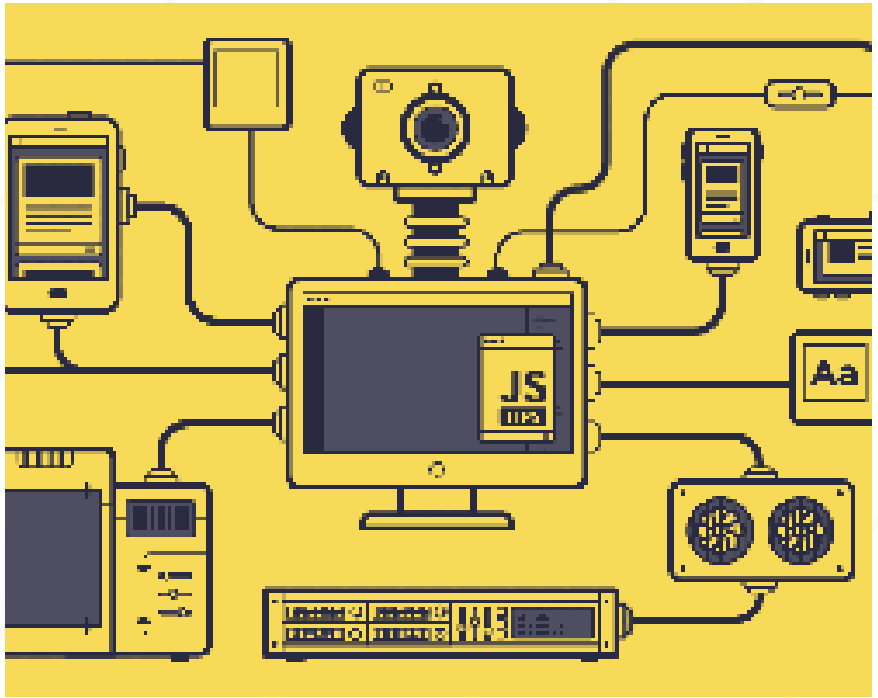
Hace que las páginas web sean más dinámicas e interactivas.



¿Por qué JS es el rey de la web?

JavaScript fue diseñado con el propósito de escribir scripts - *también conocidos como piezas de código o programas* - que pueden agregar interactividad a tu sitio web.

Por ejemplo, un script puede generar un mensaje en una caja de alerta, o proveer una lista de opciones en un menú desplegable.



Con JavaScript puedes escribir pequeñas funciones, llamadas **event handlers** (*manejadores de eventos*) y hacer que éstas se activen empleando **atributos HTML**.

```
function Saludar() {
    alert("¡Hola Mundo!");
}
```

```
<html>

    <head>
        <title>Vinculando mi JavaScript Externo</title>
        <script src = "/miproyecto/script.js" type = "text/javascript"></script>
    </head>

    <body>
        <input type = "button" onclick = "Saludar();" name = "ok" value = "Haz click" />
    </body>

</html>
```

Document Object Model

El **DOM** de un HTML es un **modelo de objetos** estándar y una **interfaz** de programación para HTML. Este define:

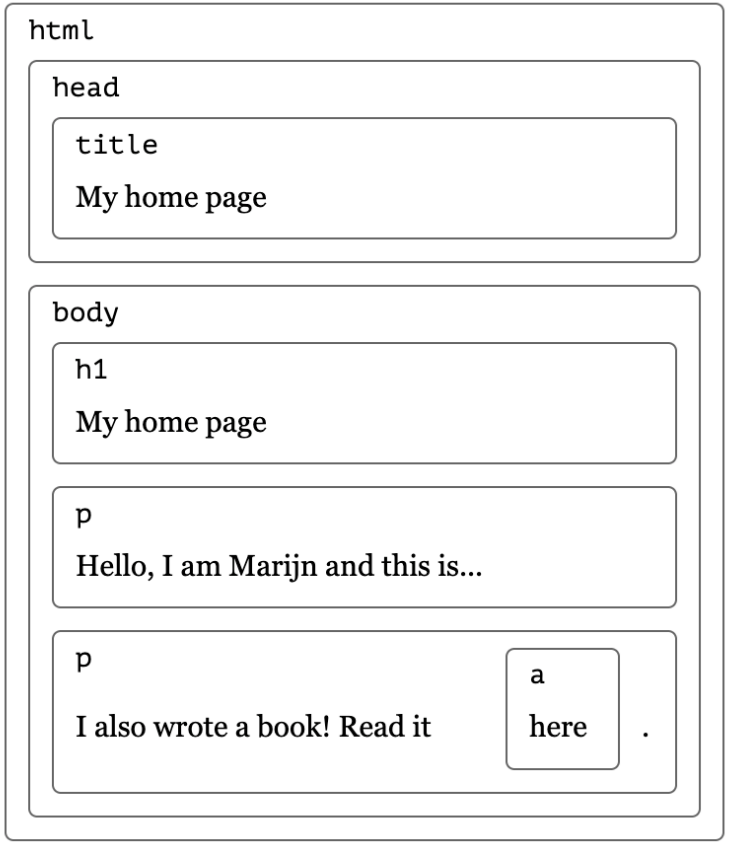
- Los elementos **HTML como objetos**
- Las propiedades de todos los elementos HTML
- Los métodos para acceder a todos los elementos HTML
- Los eventos para todos los elementos HTML

En otras palabras: El DOM de HTML es el estándar para cómo **obtener**, **modificar**, **cambiar** o **borrar** elementos HTML

¿Cómo funciona?

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Scope</title>
</head>
<body>
    <h1>DOM</h1>
    <p>
        Cuando llega el HTML al browser, este lo empieza a pasar:
        va leyendo etiqueta por etiqueta y va creando el DOM.
        Cuando este proceso termina por completo es cuendo
        obtenemos el evento <code>DOMContentLoaded</code>
    </p>
</body>
</html>
```

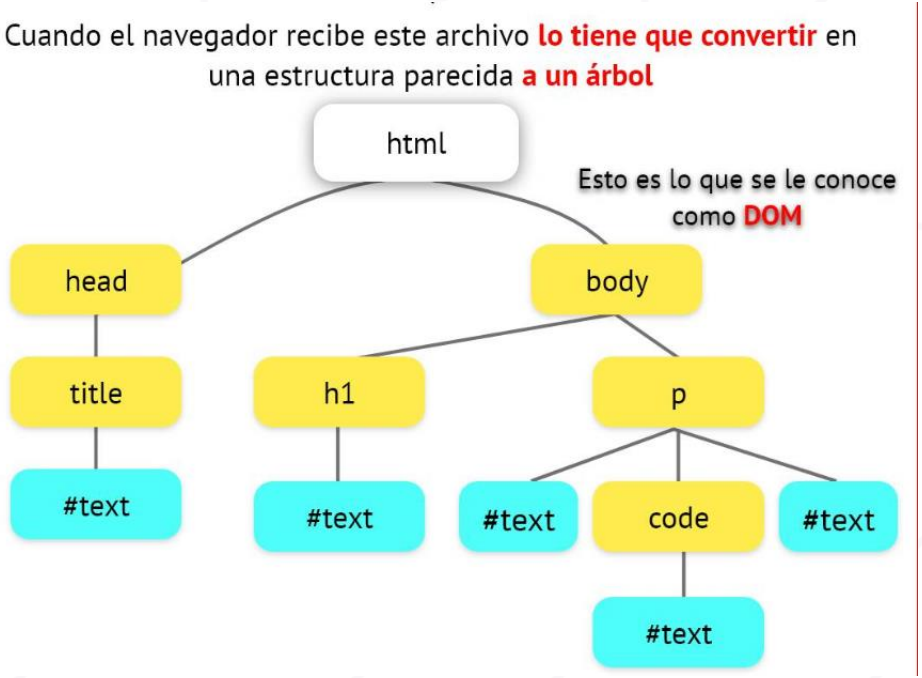
Estructura del documento

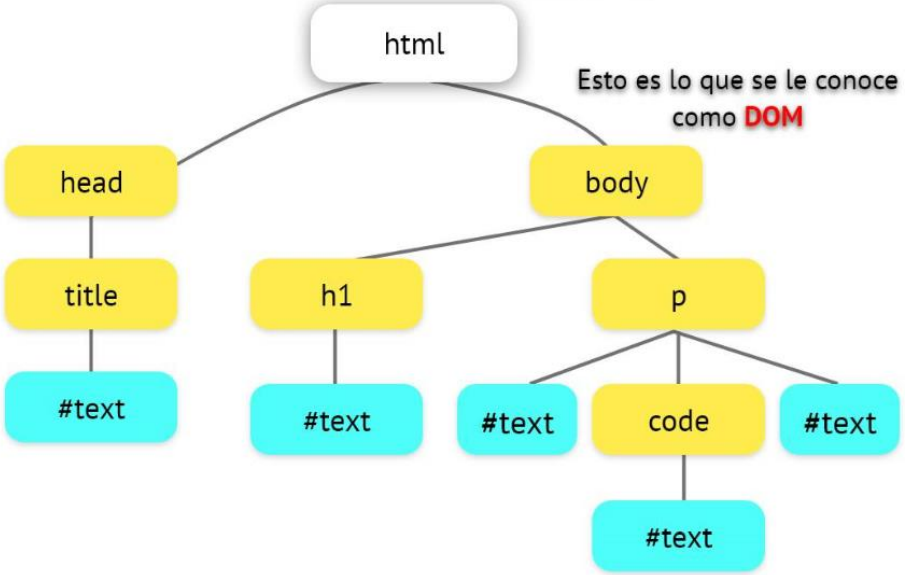



```
<!doctype html>
<html>
  <head>
    <title>My home page</title>
  </head>
  <body>
    <h1>My home page</h1>
    <p>Hello, I am Marijn and this is my home page.</p>
    <p>I also wrote a book! Read it
      <a href="http://eloquentjavascript.net">here</a>.</p>
  </body>
</html>
```

- **Mediante el DOM, JavaScript puede acceder y modificar todos los elementos de un documento HTML.**

- **Cuando se carga una página web, el navegador crea un Document Object Model de la página. El DOM se construye como un árbol de objetos:**






```
<!DOCTYPE html>
<html>

    <head>
        <title>My title</title>
    </head>

    <body>
        <h1>My header</h1>
        <a href="">My link</a>
    </body>

</html>
```

Cuando termina el **navegador** de convertirlo al **DOM** ocurre el **evento:**

DOMContentLoaded

A partir de este punto tenemos la garantía de que **todo nuestro documento se ha cargado**.

SCRIPT EXTERNOS O EMBEBIDOS

Todo script que carguemos en nuestra página tiene un llamado y una ejecución

cuando el DOM se este procesando va a detener todo el procesamiento cuando se encuentre la etiqueta **<script>**

```
<body>
    <script>
        Estos script pueden ser externos o embebidos.
    </script>

    Cuando esto ocurre no se va a leer ningún otro elemento HTML, hasta que acabemos con el script.

    <p>
        Soy un párrafo, que no se puede ejecutar todavía por culpa del script de arriba 😡
    </p>
</body>
```

por lo tanto, es importante en que lugar colocamos nuestros **scripts**

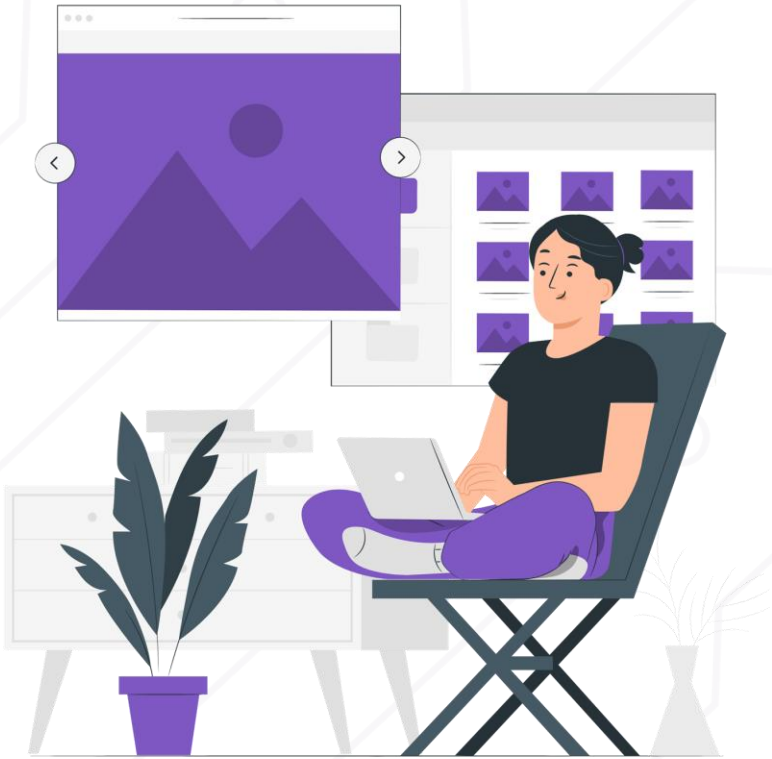
```
<body>
    <h1> titulo </h1>
    <p>  texto  </p>
    <script>El mejor lugar para colocarlo es al final del body<script>
</body>
```

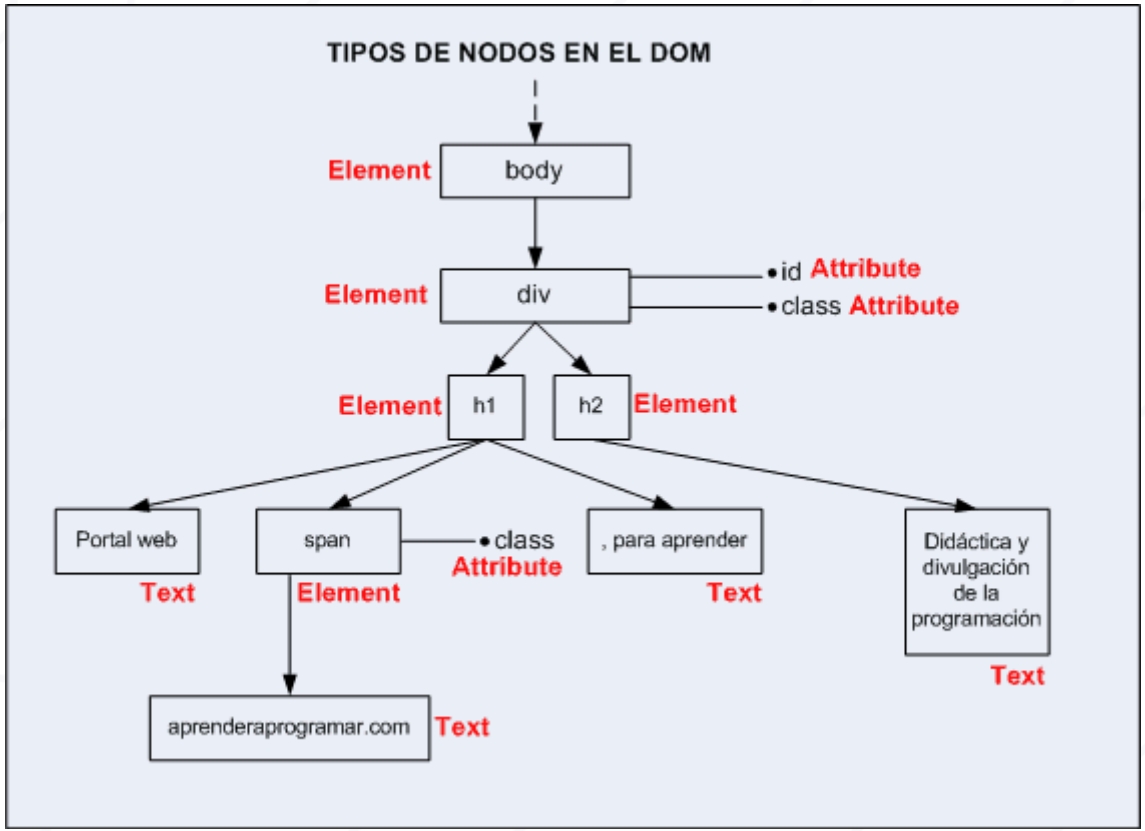
Con este modelo de objetos, JavaScript obtiene todo el poder que necesita para **crear páginas HTML dinámicas**:

- Puede modificar todos los elementos HTML de una página
- Puede modificar todos los atributos HTML de una página
- Puede modificar todos los estilos CSS de una página
- Puede remover elementos y atributos HTML de una página
- Puede agregar nuevos elementos y atributos HTML a una página
- Puede reaccionar a todos los eventos HTML existentes en una página
- Puede crear nuevos eventos HTML en una página

NODOS

Es un objeto que devuelve un conjunto de nodos. Podríamos pensar que se parece a un array, por ser un conjunto de datos (Array: lista ordenada de datos) pero no lo es. De entrada, porque no podemos trabajar con él como si lo fuese.




TIPOS DE NODOS EN EL DOM
Element
body
Element
div
•id Attribute
•class Attribute
Element
h1
h2
Element
Portal web
Text
span
•class
Attribute
Element
, para aprender
Text
Didáctica y divulgación de la programación
Text
aprenderaprogramar.com
Text

- **Una peculiaridad de los NodeList es que son colecciones dinámicas cuyo contenido se actualiza automáticamente cuando la página web cambia dinámicamente**

- **Por tanto los NodeLists podemos decir que tienen ciertas similitudes con los arrays y ciertas diferencias con estos. Los NodeList no tienen algunas posibilidades que tienen los arrays, pero a cambio son dinámicos, lo que puede resultar de gran interés en determinadas circunstancias.**

Leer nodos

`document.getElementById`

Para obtener un elemento por su Id

`document.getElementsByTagName()`

Para obtener un elemento por su Nombre de la etiqueta

`document.getElementsByClassName`

Para obtener un elemento por el nombre de la clase

Leer nodos

```
document.querySelector
```

```
document.querySelectorAll
```

Leer nodos

`nodo.innerHTML (escribir)`

LOS NODELIST PODEMOS CONVERTIRLOS A ARRAYS

```
const nodeListAsArray = [...nodeList]
```