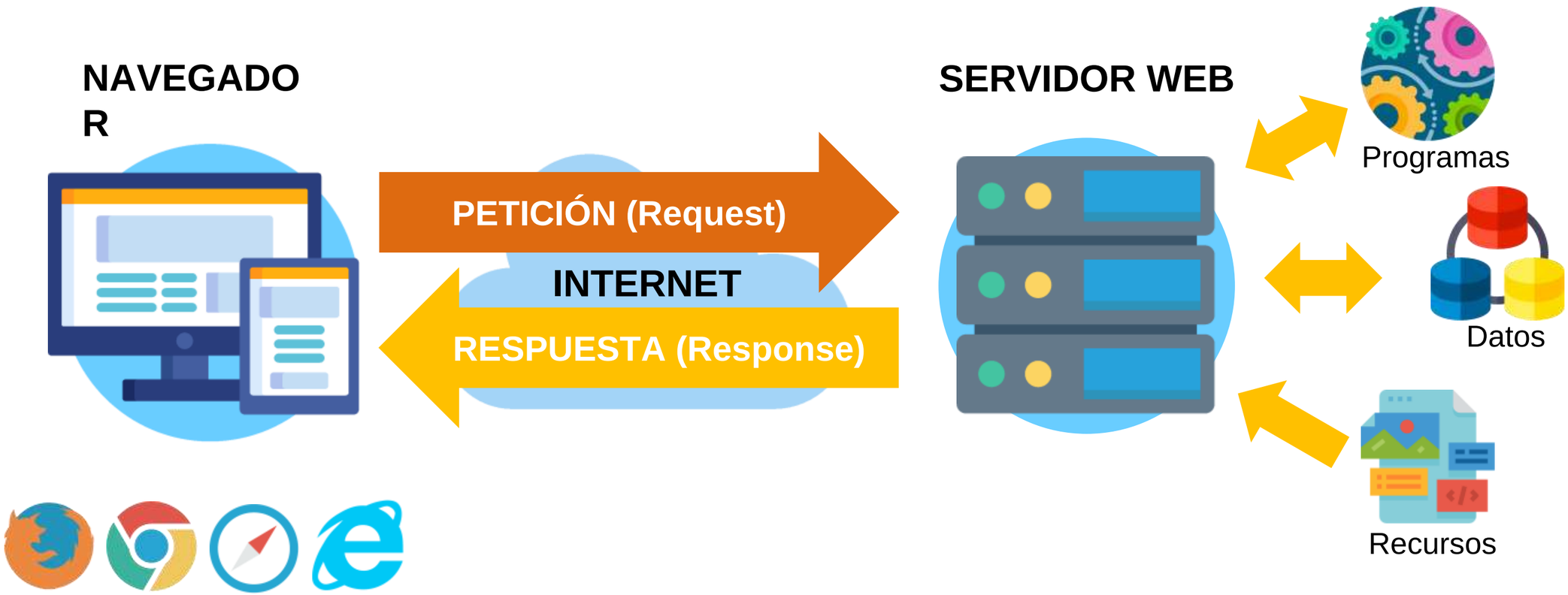
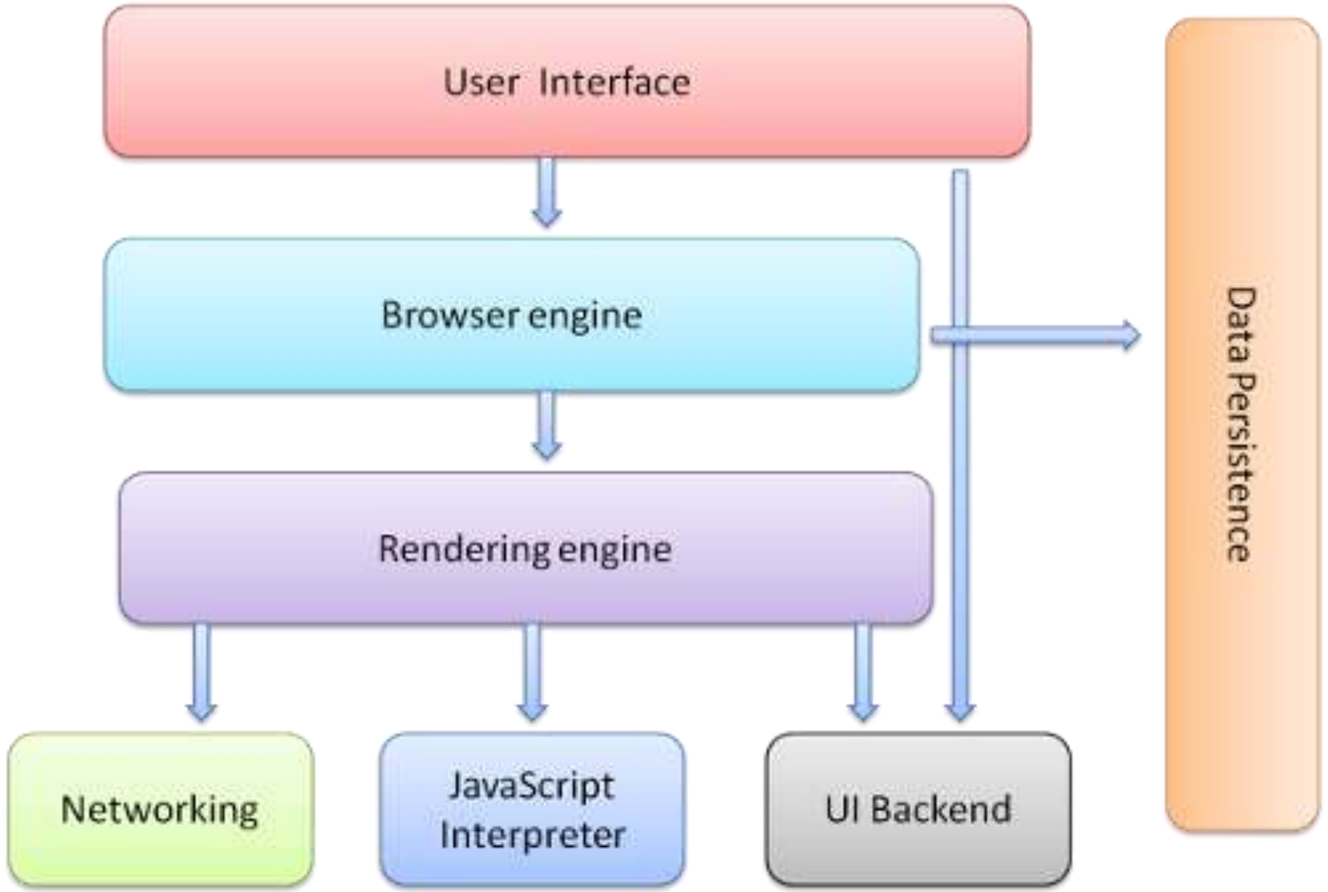
Frontend con JavaScript

Cómo funciona la web



¿Cómo funciona un navegador?



https://www.html5rocks.com/es/tutorials/internals/howbrowserswork/

Motores e intérpretes de cada navegador

	Rendering Engine	Js interpreter
Google Chrome	Blink	V8
Mozilla Firefox	Gecko	SpiderMonkey
Safari	WebKit	Nitro
Microsoft Edge	EdgeHTML	ChakraCore
Internet Explorer	Trident	Chakra

CanIUse???

Hello World

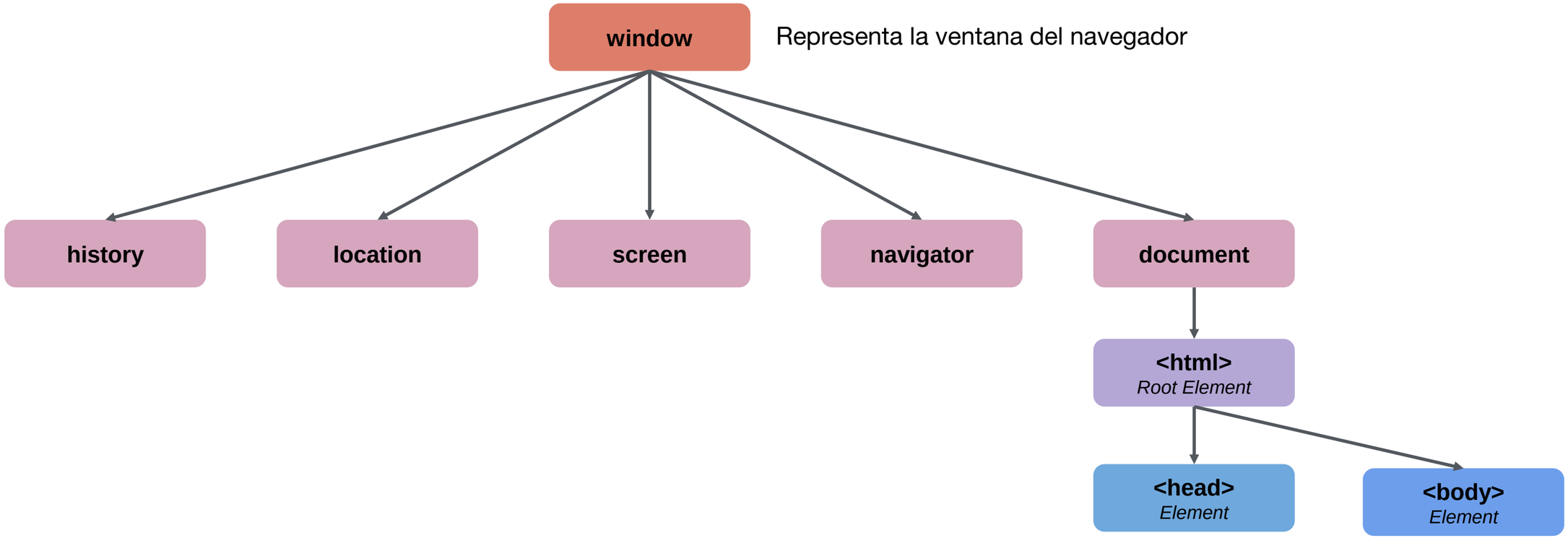
Hello World

```html
<!DOCTYPE html>
<html>
<head>
    <title>Hello World!</title>
</head>
<body>
    <script>
        console.log('Hello World')
    </script>
    <!-- importar archivo JavaScript externo -->
    <script src="hello-world.js"></script>
    <!-- importar módulo JavaScript (para poder usar la instrucción import) -->
    <script type="module" src="hello-world-module.js"></script>
</body>
</html>
```

Browser Object model (BOM)

Browser Object Model (BOM)

El **BOM** publica una serie de objetos en el contexto de JavaScript para poder controlar el navegador web.



https://developer.mozilla.org/es-ES/docs/Web/API/Window

Navigator

appCodeName
appName
appVersion
appMinorVersion
userAgent
product
productSub

browserLanguage
language
systemLanguage
userLanguage
cpuClass
oscpu
platform
userProfile
securityPolicy

mimeTypes
plugins
cookieEnabled
javaEnabled()
onLine
preference()

https://developer.mozilla.org/es/docs/Web/API/Window/navigator

Window

alert(),
confirm()
prompt()
print()
open()
close()
createPopup()
focus()
blur()
stop()

moveBy()
moveTo()
resizeBy()
resizeTo()
scrollBy()
scrollTo()
setInterval()
setTimeout()
clearInterval()
clearTimeout()

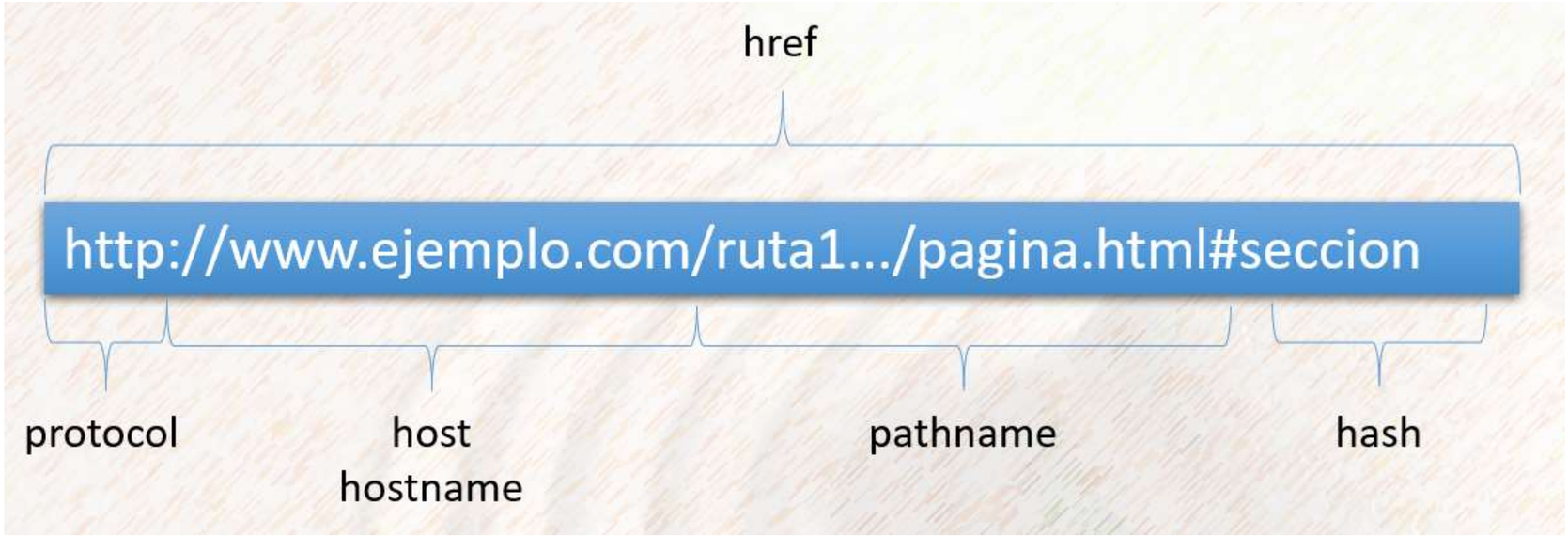
screenX
screenY
innerWidth
innerHeight
outerWidth
outerHeight

(APIS HTML5)
localStorage
sessionStorage

https://developer.mozilla.org/es/docs/Web/API/Window

location

href
protocol
host name
pathname
hash
port
search
assign()
replace()
reload()



https://developer.mozilla.org/es/docs/Web/API/Window/location

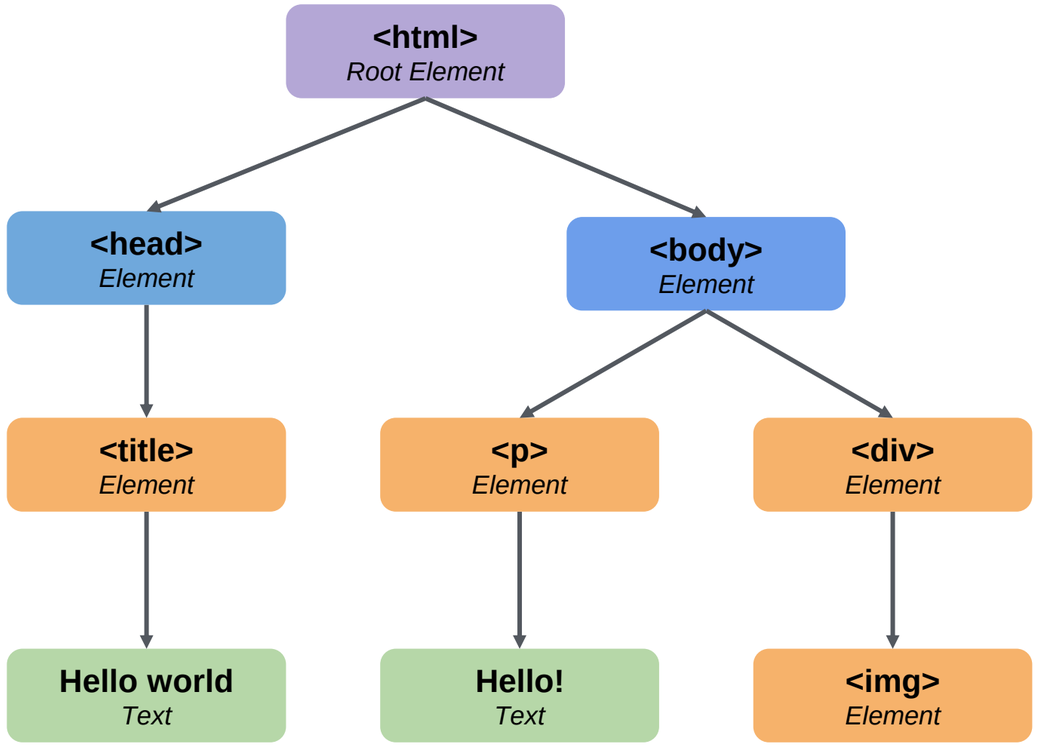
Document Object Model (DOM)

Document Object Model (DOM)

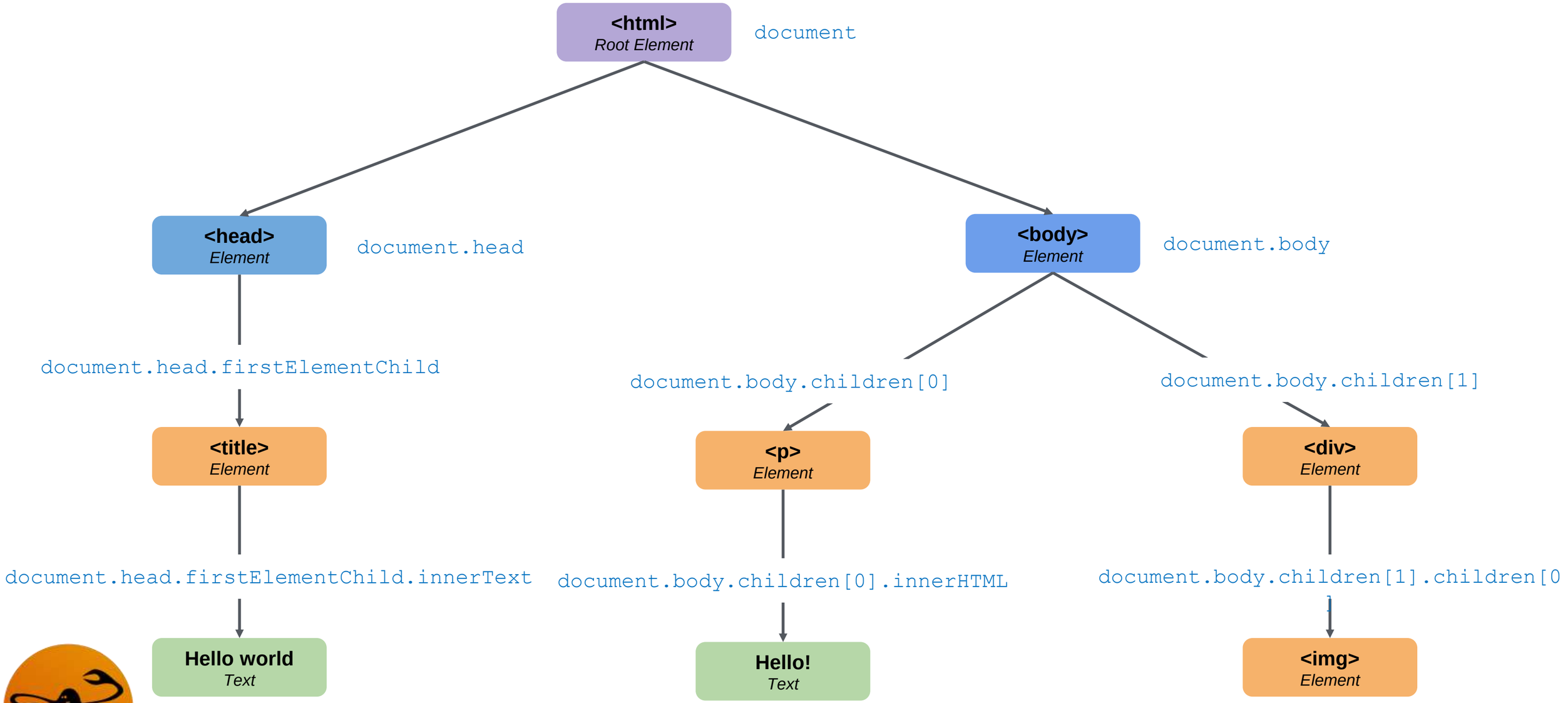
El DOM es un árbol que crea el navegador y lo pone en el contexto de JavaScript a través de la variable **document**.

A través **document**, podemos acceder a sus elementos, que son los elementos que hay en nuestro HTML.

```
<html>
      <head>
    <title>Hello World</title>
  </head>
  <body>
    <p>Hello!</p>
    <div>
       <img src="example.png"/>
    </div>
  </body>
</html>
```



Navegando por el árbol



Element

Los nodos que cuelgan de document.body o document.head, son de instancias **Element**.

La clase **Element** nos proporciona atributos y métodos para manipular estos nodos:

- **attributes:** array de atributos del nodo
- **innerHTML:** HTML contenido dentro del nodo
- **textContent:** texto incluído dentro del nodo
- **classList:** array de clases CSS que tiene el nodo
- **appendChild(<Element>):** permite añadir un elemento como hijo
- **removeChild(<Element>):** permite eliminar un elemento hijo
- **replaceWith(<Element>):** reemplaza el elemento actual por el que se pasa como parámetro

https://developer.mozilla.org/es/docs/Web/API/Element

Seleccionando nodos

Desde cualquier nodo, podemos realizar búsquedas de elementos hijos para ir más rápido:

- `nodo.getElementById(<id>) -> [<Element>]`
- `nodo.getElementsByTagName(<tag>) -> [<Element>]`
- `nodo.getElementsByName(<name>) -> [<Element>]`
- `nodo.getElementsByClassName(<clase CSS>) -> [<Element>]`
- `nodo.querySelector(<selector CSS>) -> 1er <Element>`
- `nodo.querySelectorAll(<selector CSS>) -> [<Element>]`

Crear y eliminar elementos

Desde cualquier nodo, podemos realizar búsquedas de elementos hijos para ir más rápido:

- `document.createElement('tag name') -> <Element>`
- `nodo.appendChild(<Element>)`
- `nodo.innerHTML = '<html code>'`
- `nodo.outerHTML = '<html code>'`
- `nodo.remove()`

Manipulando atributos

Desde cualquier nodo, podemos realizar modificaciones de sus atributos:

- `nodo.attributes`
- `nodo.setAttribute(“<name>”, “<value>”)`
- `nodo.getAttribute(“<name>”) - > <value>`
- `nodo.hasAttribute(“<name>”) -> Boolean`
- `nodo.removeAttribute(“<name>”)`

Manipulando estilos y clases CSS

Desde cualquier nodo, podemos realizar modificaciones de sus atributos y clases CSS:

- `nodo.style.<cssAttributeInCamelCase>`
- `nodo.className // valor del atributo class de HTML`
- `nodo.classList - > [<CSS name>]`
  - `nodo.classList.add(<CSS name>) // añade`
  - `nodo.classList.remove(<CSS name>) // elimina`
  - `nodo.classList.toggle(<CSS name>) // alterna`
  - `nodo.classList.contains(<class name>) - > Boolean`

Twitter Clon

Eventos

Eventos

JavaScript es un lenguaje orientado a eventos.

La filosofía general es poner callbacks o handlers a los eventos, de manera que todo funciona de manera reactiva.

Estos callbacks recibirán siempre como parámetro el objeto que identifica el evento.

https://developer.mozilla.org/es/docs/Web/Events

Manejando eventos in-line

```
<button onclick='getElementById("demo").innerHTML=Date()'>The time is?</button>
```

```
<button onclick='doSomething()'>The time is?</button>

<script>
function doSomething() {…}
</script>
```

Manejando eventos in-line

Manejando eventos desde JavaScript

```
<button id='magicButton'>The time is?</button>

<script>
function doSomething() {…}

document.getElementById('magicButton').addEventListener('click', doSomething);

document.getElementById('magicButton').removeEventListener('click', doSomething);

</script>
```

Manejando eventos desde JavaScript

Comportamiento por defecto

Comportamiento por defecto

Hay componentes de HTML a los que el navegador les asigna un comportamiento por defecto.

Por ejemplo, los <a> nos suelen llevar a donde indica su atributo href, pero podemos decirle al navegador que no ejecute ese comportamiento.

```
document.getElementById("a").addEventListener("click", function(ev){
    ev.preventDefault(); // no nos envía al link especificado
});
```

Event Bubbling & Capturing

Event Bubbling & Capturing

Como el DOM es jerárquico (hay unos elementos dentro de otros), cuando hacemos click en un elemento, se realiza una propagación del mismo.

Es decir, si haces click en un elemento que está dentro de otro, estás a su vez haciendo click en el elemento contenedor, por tanto hay un orden de propagación del evento.

Event Bubbling

Ejecuta primero el evento en el objeto interior y luego lo propaga hacia arriba.

Como ocurre con las burbujas debajo del agua.

```
<div id="div">
      <p id="p">I'm batman</p>
</div>

document.getElementById("p").addEventListener("click", function(){ alert("P"); });
document.getElementById("div").addEventListener("click", function(){ alert("DIV"); });
```

Primero hace el alert “P” y luego con “DIV”

Event Capturing

Ejecuta primero el evento en el objeto exterior y luego lo propaga hacia abajo.

```
<div id="div"><p id="p">I'm batman</p></div>

document.getElementById("p").addEventListener("click", function(){ alert("P"); }, true);
document.getElementById("div").addEventListener("click", function(){ alert("DIV"); }, true);
```

Primero hace el alert “DIV” y luego con “P”

Parar la propagación

Podemos evitar que el evento se propague.

```
document.getElementById("p").addEventListener("click", function(event){
    event.stopPropagation();
});
```

Promesas

Involucrados en una promesa

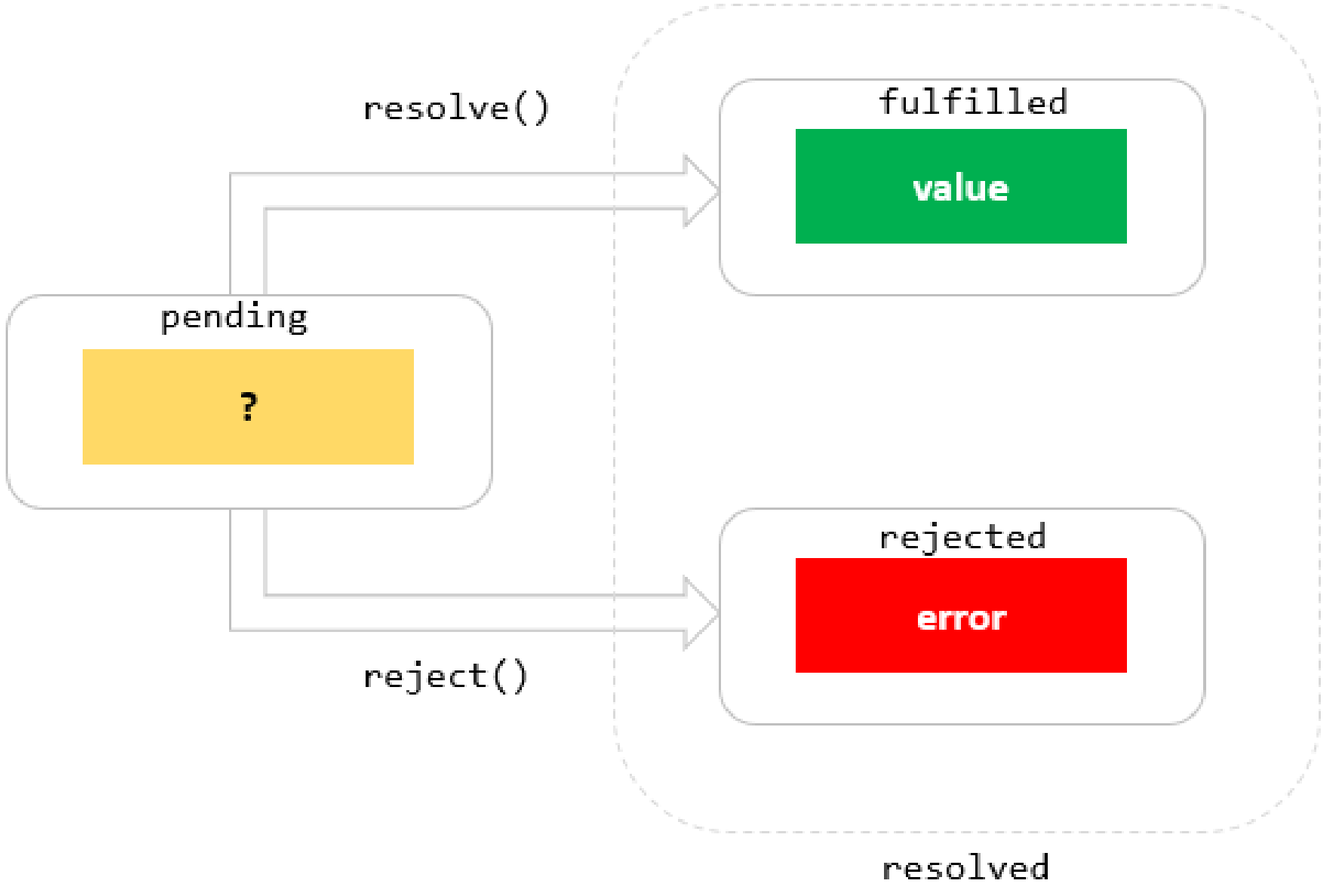
Hay 2 involucrados en una promesa. Podemos identificarlos como productor y consumidor.

- Productor: es quien gestiona un proceso asíncrono mediante una promesa y quien decide si la operación concluye bien o no.
- Consumidor: es quien reacciona ante el cambio de estado de la promesa.

Estados

Los estados de una promesa son los siguientes

- *pending*
- *fulfilled*
- *rejected*



Nos vamos a comprar patatas

Nuestra madre nos manda a comprar patatas. Nos vamos a la compra, no sabemos cuándo volveremos ni si habrá patatas en el mercado o no.

Mientras, nuestra madre sigue a sus cosas, esperándonos. Nos ha comentado que si traemos patatas, hará una tortilla. En caso contrario, pediremos una pizza por teléfono.

Nos vamos a comprar patatas

```
function irAPorPatatas() {
  return new Promise(function(resolve, reject) {
    const hayPatatas = buscarPatatasEn3Sitios()
    if (hayPatatas) {
      resolve('🥔🥔')
    } else {
      reject('😢😢')
    }
  })
}
```

```
irAPorPatatas
  .then(hacerTortilla)
  .catch(pedirPizza)
```

Peticiones HTTP con fetch

Peticiones HTTP con fetch

Desde el navegador podemos realizar peticiones HTTP a servidores utilizando el API de fetch, basada en promesas.

```
fetch('https://jsonplaceholder.typicode.com/todos')
  .then(response => response.json())
  .then(data => console.log(data));
```

Y si queremos parsear el JSON...nos devuelve otra promesa.

https://developer.mozilla.org/es/docs/Web/API/Fetch_API

Almacenamiento local

Almacenamiento local

El estándar de HTML5 ofrece APIs JavaScript para el almacemiento local datos:

- **Cookies:** almacenamiento clave-valor que se envía y recibe del server
- **Storage:** almacenamiento clave-valor (localStorage & sessionStorage)
- **Indexed DB:** base de datos

localStorage & sessionStorage

- **localStorage:** los valores almacenados están disponibles mientras el usuario no borre los datos de navegación.
- **sessionStorage:** los valores almacenados están disponibles hasta que el usuario cierra la pestaña/ventana.

```
Storage.getItem(<key>) // devuelve <value> en <key>
Storage.setItem(<key>, <value>) // guarda <value> en <key>
Storage.removeItem(<key>) // elimina el <value> de <key>
Storage.clear() // elimina todos los valores
```

https://developer.mozilla.org/es/docs/Web/API/Storage

localStorage & sessionStorage

- **Sólo podemos guardar valores primitivos** (no podemos guardar objetos o arrays).
- **Alternativa:** serializar/de-serializar

```
let obj = {a: 1, b: true, c: “hey”};
localStorage.setItem(‘serializedObj’, JSON.stringify(obj));
const serializedObj = localStorage.getItem(‘serializedObj’);
obj = JSON.parse(serializedObj);
```

¿Por dónde sigo?

¿Por dónde sigo?

Existen infinidad de Web APIs que puedes utilizar desde el navegador:

- Drag and Drop
- File System
- Fullscreen
- Geolocation
- Payment Request
- Network Information
- Web Workers
- Web Notifications
- Web Animations

GRACIAS

www.keepcoding.io