JavaScript

LIMA UD6 – Tema 1

IES Plurilingüe Antón Losada Diéguez

Tabla de contenido

JavaScript. JS

1. Introducción

JavaScript é unha linguaxe de programación de alto nivel, dinámica, débilmente tipada e interpretada. A súa funcionalidade principal é proporcionar interactividade ás páxinas web, aínda que tamén se usa en outros contextos, como servidores web ou bases de datos. Algunhas das partes esenciais de JavaScript son:

- Variables: As variables son contedores para almacenar valores de datos. Podes usar a palabra clave let para declarar unha variable, como let miVariable = 5;.
- Funcións: As funcións son bloques de código reutilizables. Podes definir unha función usando a palabra clave function, seguida dun nome para a función, e despois os parénteses (), como function miFuncion() {}.
- Estruturas de control: JavaScript ten varias estruturas de control, como if para condicións, for para bucles e switch para seleccionar un bloque de código a executar.
- Obxectos: JavaScript é unha linguaxe orientada a obxectos. Os obxectos son coleccións de pares chave-valor, que poden incluír funcións (métodos) e outras estruturas de datos.
- Manipular a estrutura, o estilo e o contido dunha páxina web a través do DOM (Modelo de Obxectos do Documento)

2. Modelo DOM

O DOM (*Document Object Model*) ou modelo de obxectos do documento é unha API (Application Programming Interface) que proporciona un conxunto estándar de obxectos para representar, acceder e actualizar contidos, estrutura e estilos nun documento HTML (HyperText Markup Language) ou XML (eXtensible Markup Language). Trátase dun estándar do W3C (*World Wide Web Consortium*).

Está separado en tres partes fundamentais:

- ***Core DOM***: modelo estándar para todo tipo de documentos.
- ***XML DOM***: modelo estándar para os documentos XML.
- ***HTML DOM***: modelo estándar para os documentos HTML.

A tarefa máis habitual na programación web adoita ser a manipulación do seu contido para crear novos elementos, facer animacións, manexar eventos, etc. Todas estas tarefas pódense realizar de forma sinxela mediante o DOM. Son os diferentes navegadores webs os encargados de realizar esa transformación do noso documento nunha estrutura xerárquica de obxectos. Desta forma, podemos acceder ao seu contido con métodos máis estruturados.

O DOM transforma todos os documentos XHTML nun conxunto de elementos aos que se chama *nodos*, cada un dos cales é un obxecto. Ditos nodos, que están conectados entre si, representan os contidos da páxina web e a relación que hai entre eles. Cando unimos todos os nodos de forma xerárquica, obtemos unha estrutura similar a unha árbore que se adoita chamar a **árbore DOM** ou **árbore de nodos**.

A continuación, na figura 1 móstrase a estrutura dunha páxina HTML sinxela e a súa árbore DOM asociada.

```
<!DOCTYPE html>
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
        <a href="http://www.edu.xunta.gal">My link</a>
        <h1>My Header</h1>
    </body>
</html>
```

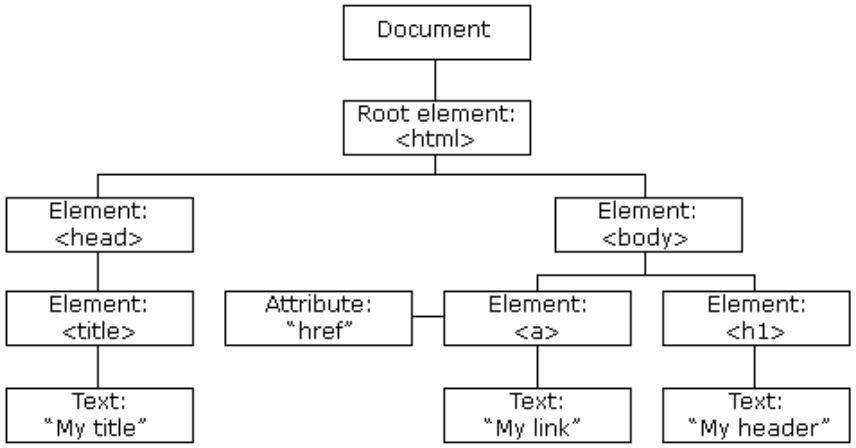

Figura 1: Obxectos da árbore DOM do anterior documento HTML

Cada rectángulo da imaxe representa un nodo do DOM. As liñas indican como se relacionan os nodos entre si. A raíz da árbore de nodos é un nodo especial, denominado *document*. A partires dese nodo, cada etiqueta HTML transfórmase en nodos do tipo elemento ou texto. Os nodos de tipo texto, conterán o texto que existe nesa etiqueta. A conversión realízase de forma xerárquica, é dicir, o nodo inmediatamente superior é o nodo pai e todos os nodos que están no seguinte nivel por debaixo son os nodos fillo.

Desta forma, JavaScript pode manipular de maneira dinámica calquera operación sobre o documento: modificación/eliminación/inserción de elementos HTML, atributos, estilos CSS e manexo de eventos.
As versións do DOM foron evolucionando dende DOM 1 que proporcionaba un modelo completo para todo un documento HTML ou XML incluíndo as ferramentas para cambiar calquera parte do documento, DOM 2 introduciu a función `getElementById`, así como un modelo de eventos e soporte para espazos de nomes XML e CSS, DOM 3 que é a que se emprega na maioría dos navegadores que permitiu agregar soporte para o manexo de eventos de teclado e XPath, así como unha interface para serializar documentos como XML e finalmente DOM 4.1 que é a versión na que o WHATWG (Web Hypertext Application Technology Working Group) está traballando na actualidade.

2.1. Tipos de Nodos

A especificación do DOM define 12 tipos de nodos dos cales 5 son os máis empregados:

- ***Document***: é o nodo raíz do que derivan todos os demais nodos da árbore.
- ***Element***: representa cada unha das etiquetas XHTML. Trátase do único que pode conter atributos e o único do que poden derivar outros nodos.
- ***Attr***: representa cada atributo das etiquetas XHTML polo que haberá un nodo por cada par atributo-valor.

- ***Text*:** é o nodo que contén o texto incluído na etiqueta XHTML correspondente.
- ***Comment*:** representa os comentarios incluídos na páxina XHTML.
- **Outros:** *CdataSection*, *DocumentFragment*, *DocumentType*, *EntityReference*, *Entity*, *Notation* e *ProcessingInstruction*.

Unha vez coñecida a árbore DOM, podemos empregar funcións para acceder a calquera nodo da árbore. Podemos acceder a un nodo específico (elemento XHTML) de dúas formas: (1) a través dos nodos pai, (2) usando un método de acceso directo.

Para acceder dende os nodos pais, partimos do nodo raíz e imos accedendo a través dos fillos ata chegar ao elemento que precisamos. No caso do método de acceso directo (que é o máis empregado) facemos uso das funcións do DOM que nos permiten ir directamente a un elemento sen ter que ir nodo a nodo. Non obstante, para ter acceso a calquera nodo da árbore DOM, esta ten que estar completamente construída, é dicir, a páxina XHTML tivo que ser cargada correctamente na súa totalidade.

Na árbore DOM debemos distinguir tamén entre *métodos* que son accións que podemos realizar sobre elementos HTML (por exemplo: engadir ou borrar un elemento HTML) e *propiedades* que son valores dos elementos HTML que poden ser modificados (por exemplo: cambiar o contido dun elemento HTML).

Se observamos o seguinte exemplo podemos ver un método `getElementById` que nos permite ter acceso a un determinado nodo mediante o seu ID, e a propiedade `innerHTML` que accede ao seu contido e o modifica.

```
<script>
    document.getElementById("exemplo").innerHTML = "A todo porquiño lle chega o
seu san Martiño";
</script>
```

2.2. Obxecto Document, acceso a nodos tipo elemento:

O obxecto *Document* é o propietario de todos os demais obxectos da nosa páxina web polo que se precisamos acceder a calquera elemento no HTML sempre se fará mediante o uso do obxecto *Document*.

Para encontrar calquera elemento HTML ou propiedade CSS no noso documento existen catro métodos principais:

- ***document.getElementById(id)*:** devolve o elemento que corresponda con ese *id*.
- ***document.getElementsByTagName(tagName)*:** devolve unha colección HTML cos elementos que se correspondan con esa etiqueta *tagName*.
- ***document.getElementsName(nameValue)*:** devolve unha colección HTML cos elementos que teñan como atributo *name* o correspondente valor *nameValue*.
- ***document.getElementsByClassName(classsName)*:** devolve os elementos que se correspondan con ese nome de clase *className*.

Supoñamos o seguinte exemplo dun *<input>* de tipo texto:

```
<input type="text" id="user" name="usuario" class="inputClass" >
```

Para acceder a ese elemento a función mais empregada é a de `getElementById`:

```
<script>
    let user= document.getElementById("user");
</script>
```

Sen embargo poderiamos empregar a de `getElementsByName` en base ao atributo `name`. Esta función devolve unha colección polo que é preciso indicar a posición á que queremos acceder. Se soamente obtemos un elemento sería a posición 0.

```
<script>
    document.getElementsByName("usuario")[0];
</script>
```

Se por exemplo tivésemos 5 elementos, para acceder ao terceiro teriamos que indicar o índice 2:

```
<script>
    document.getElementsByName("usuario")[2];
</script>
```

Podemos acceder a todos os elementos cun simple bucle:

```
<script>
    for(let i=0; i<document.getElementsByName("usuario").length;i++){
        let elemento=document.getElementsByName("usuario")[i];
        console.log(elemento);
    }
</script>
```

Outra forma de acceder é a través do nome de etiqueta mediante o método `getElementsByTagName` o cal devolve unha colección de elementos que coincidan con esa etiqueta. Neste caso, igual que no anterior, precisamos indicar a posición do elemento que queremos obter de dita colección. Supoñendo que é o primeiro, indicaremos o índice 0.

```
<script>
    let elemento=document.getElementsByTagName("input")[0];
</script>
```

Por último, se o que queremos é acceder a unha propiedade CSS podémolo facer a través da clase correspondente empregando o método `getElementsByClassName`. Dita función devolve unha colección con todos os elementos que se corresponden con dito nome de clase.

```
<script>
    let elemento=document.getElementsByClassName("inputClass")[0];
</script>
```

É importante destacar neste punto que unha colección HTML (`HTMLCollection`) non é un *array*. Podemos acceder aos elementos polo seu índice e recorrelos mediante un bucle pero non dispoñemos de métodos de *arrays* como é o caso de `valueOf()`, `pop()`, `push()`, `join()`.
Tamén existen dous métodos moi empregados para o acceso aos elementos que son:

- `document.querySelector("selector")`: devolve o primeiro elemento que emparella co selector CSS indicado como parámetro.
- `document.querySelectorAll("selector")`: devolve todos os elementos que emparellan co selector CSS indicado. Trátase dunha lista de nodos (`NodeList`) que analizaremos máis adiante.

```
<script>
    let primeiroElemento=document.querySelector("p");
    let elementos=document.querySelectorAll("p");
</script>
```

2.3. Acceso a nodos de tipo atributo:

Unha vez que xa coñecemos como acceder aos nodos de tipo elemento é preciso tamén ter acceso aos seus atributos.

Propiedade	Descrición
nodeName	Valor da propiedade *nodeName* do elemento. Por exemplo: o tipo de etiqueta HTML
nodeValue	Valor da propiedade *nodeValue* do elemento. Por exemplo: si é unha etiqueta HTML será null.
nodeType	Constante numérica que identifica o tipo de nodo.
attributes	Referencia ao obxecto que contén os atributos.
ownerDocument	Referencia ao obxecto documento propietario.

Supoñamos un exemplo no que queremos acceder ao atributo `type` do noso `<input>`. Podemos empregar a propiedade `attributes` que nos devolverá unha colección con tantos pares atributo-valor como teña o noso nodo elemento accesibles mediante `nodeValue` e `nodeName`.

Dado o código do `<input>` xa empregado anteriormente:

```
<input type="text" id="user" name="usuario" class="inputClass">
```

Podemos imprimir todos os seus atributos mediante un bucle:

```
<script>
    for(let i=0; i<document.getElementById("user").attributes.length;i++){
        let elemAtt=document.getElementById("user").attributes[i];
        console.log(elemAtt.nodeName+" - "+elemAtt.nodeValue);
    }
</script>
```

Dito exemplo daranos como resultado os catro atributos co seu valor (`type`, `id`, `name` e `class`). Ademais, tamén poderiamos modificar facilmente os seus valores asignando un novo valor ao correspondente atributo. Por exemplo:

```
<script>
    document.getElementById("user").attributes[2].nodeValue="usuarioNuevo";
    for(let i=0; i<document.getElementById("user").attributes.length;i++)
    {
        let elemAtt=document.getElementById("user").attributes[i];
        console.log(elemAtt.nodeName+" - "+elemAtt.nodeValue);
    }
</script>
```

No caso de non saber a posición que ocupa o atributo que queremos na colección de `attributes` podemos indicalo expresamente polo seu nome. No exemplo mostrado a continuación obsérvase por consola o valor do atributo `type`:

```
<script>
    console.log(document.getElementById("user").attributes["type"].nodeValue);
</script>
```

Outra forma de visualizar dita información sería empregando directamente a propiedade `type` de ese nodo elemento ou a función `getAttribute` pasando como parámetro o nome do atributo que queremos consultar:

```
<script>
    console.log(document.getElementById("user").type);
    console.log(document.getElementById("user").getAttribute("type"));
</script>
```

A propiedade `nodeType` dun nodo como se comentou con anterioridade, é unha constante numérica que indica o tipo de nodo. Trátase dunha propiedade soamente de lectura.

Propiedade	Tipo	Exemplo
ELEMENT_NODE	1	<h1 id="titulo"> Exemplo de texto en h1 </h1>
ATTRIBUTE_NODE	2	id="titulo" (obsoleto, *deprecated)*
TEXT_NODE	3	Exemplo de texto en h1
COMMENT_NODE	8	<!-- Isto é un comentario -->
DOCUMENT_NODE	9	O HTML por si mesmo (o pai de <html>)
DOCUMENT_TYPE_NODE	10	<!DOCTYPE HTML>

2.4. Acceso a nodos de tipo texto e navegación polo DOM:

Por último, falta ter acceso aos nodos de tipo texto, que son os que realmente teñen o contido do que se visualiza na páxina.

Dado un exemplo de código HTML como o que se mostra a continuación:

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Unidade 6</title>
</head>
<body>
    <h1> Cancións galegas populares </h1>
    <div id="cancion1">
        <h2> A Rianxeira </h2>
        <p class="paragrafo1">
            Ondiñas veñen
            ondiñas veñen,
            ondiñas veñen e van.
            Non te vaias rianxeira,
            que te vas a marear.
        </p>
        <p class="paragrafo2">
            A Virxe de Guadalupe
            cando vai pola ribeira,
            descalciña pola area,
            parece unha rianxeira.
        </p>
    </div>
    <div id="cancion2">
        <h2> Polo Río Abaixo </h2>
        <p class="paragrafo1">
```

```
            Polo río abaixo vai
            unha troita de pé,
            corre que te corre,
            quen a puidera coller,
            quen a puidera coller,
            quen a puidera pillar.
        </p>
        <p class="paragrafo2">
            Polo río abaixo vai,
            unha troita de pé.
        </p>
    </div>
</body>
</html>
```

Neste punto é útil poder navegar dende un nodo da nosa árbore DOM a outro, feito que é posible a través dunha serie de propiedades:

Propiedade	Descrición
parentNode	Referencia ao obxecto pai
childNodes	Referencia a unha lista de nodos que contén aos nodos fillo. Se engadimos o índice accederemos a ese fillo concreto. Por exemplo para acceder ao segundo fillo: *childNodes[1]*
firstChild	Referencia ao primeiro nodo fillo
lastChild	Referencia ao último nodo fillo
previousSibling	Referencia ao nodo irmán previo.
nextSibling	Referencia ao nodo irmán seguinte.

Hai que ter en conta que se hai espazos dentro dunha etiqueta é considerado texto, polo tanto convértese nun nodo mais de tipo texto. Isto vese mais claro cun exemplo:

```
<p id="proba1">
    <span> proba </span>
</p>
```

¿Cantos nodos fillos ten a etiqueta `<p>`? Ten tres fillos: (1) nodo texto cos espazos en branco, (2) nodo elemento coa etiqueta `<span>` e (3) nodo texto cos espazos en branco do final.

Sen embargo se quitamos todos os espazos pasaremos a ter un único nodo fillo que é a etiqueta `<span>`:

```
<p id="proba1"><span> proba </span></p>
```

Podemos comprobar ambos exemplos observando o número de nodos fillos:

```
<script>
    console.log(document.getElementById("proba1").childNodes.length)
</script>
```

`childNodes` devolve unha lista de nodos (`NodeList`) extraídos do documento. Pode ser entendida como unha colección HTML (`HTMLCollection`) como cando empregábamos `getElementsByTagName`, pero non é o mesmo. Tanto nunha lista de nodos como nunha colección HTML accedemos aos elementos que conteñen, a través dun índice, como se se tratase dun *array* (pero non o son) e podemos recorrelos cun bucle. Sen embargo, nunha colección HTML podemos acceder polo seu nome, id ou nº de índice e na lista de nodos unicamente polo número de índice. Ademais, os obxectos `NodeList` son os únicos que poden conter nodos atributo e nodos texto.

A mellor forma de comprender cada unha das propiedades de navegación polos nodos do DOM é facendo probas sobre o HTML empregado neste apartado, recordando que os espazos en branco crean un novo nodo de texto.

- Selección do * * con id *cancion1*, o texto do parágrafo 1 ("Ondiñas veñen...").
- O mesmo caso co anterior pero empregando *firstChild* no canto de *childNodes[0]*.
- Dado o parágrafo 1 do * * con id *cancion1* accedemos ao anterior irmán (nodo de texto cos espazos en branco), ao anterior irmán dese nodo de texto (nodo do elemento *<h2>*). Polo tanto, obtemos o contido do *<h2>* *Rianxeira*.
- O mesmo que o caso anterior pero en lugar de acceder aos dous irmáns anteriores, accedemos a dous irmáns posteriores dende a etiqueta *<h2>*. Polo tanto accedemos ao contido do parágrafo 1 ("Ondiñas veñen...").
- Acceso ao *<body>* (nodo pai do * * con id *cancion2*), primeiro fillo (nodo de texto baleiro) e seguinte irmán o cal nos devolve o contido da etiqueta *<h1>* ("Cancións galegas populares").
- Acceso ao *<body>* (nodo pai do * * con id *cancion2*), último fillo que é o contido de *<script>*.

Estes exemplos son mostrados por consola como se indican no seguinte código:

```
<script>
    console.log(document.getElementById("cancion1").childNodes[3].childNodes[0].
textContent);
    console.log(document.getElementById("cancion1").childNodes[3].firstChild.tex
tContent);
    console.log(document.getElementById("cancion1").childNodes[3].previousSiblin
g.previousSibling.textContent);
    console.log(document.getElementById("cancion1").childNodes[1].nextSibling.ne
xtSibling.textContent);
    console.log(document.getElementById("cancion2").parentNode.firstChild.nextSi
bling.textContent);
    console.log(document.getElementById("cancion2").parentNode.lastChild.textCon
tent);
</script>
```

2.5. Acceso a propiedades, colección ou elementos predefinidos:

No primeiro nivel do DOM (DOM 1) especificáronse 11 obxectos HTML, coleccións e propiedades, que son válidas aínda con HTML5, pero no nivel 3 de DOM (DOM 3) engadíronse moitos mais, dos cales destacamos os principais na seguinte táboa:

Propiedades, coleccións, obxectos	Descrición	DOM
document.anchors	Devolve un listado de tódalas anclas <a> no documento	1
document.baseURI	Devolve o URI absoluto do documento	3
document.body	Devolve o elemento <body>	1
document.cookie	Devolve as *cookies* do documento	1
document.doctype	Devolve o *doctype* do documento	3
document.documentElement	Devolve o elemento <html>	3
document.documentURI	Devolve o URI do documento	3
document.domain	Devolve o nome de dominio do servidor do documento	1
document.embeds	Devolve todos os elementos <embed>	3

document.forms	Devolve todos os formularios, elementos <form>	1
document.head	Devolve o elemento <head>	3
document.images	Devolve tódalas imaxes <img>	1
document.implementation	Devolve a implemetación do DOM	3
document.inputEncoding	Devolve o código de caracteres do documento	3
document.lastModified	Devolve a data (día e hora) na que o documento foi actualizado por última vez	3
document.links	Devolve todos os elementos <a> e <area> que teñen un atributo href	1
document.readyState	Devolve o estado de carga do documento	3
document.referrer	Devolve a URI da páxina que enlazou coa páxina actual	1
document.scripts	Devolve todos os elementos <script>	3
document.title	Devolve o elemento <title>	1
document.URL	Devolve a URL completa do documento	1

A mellor forma de visualizar ditas propiedades e coleccións é a través da consola mediante o uso de calquera HTML:

```
<script>
    console.log(document.anchors);
    console.log(document.baseURI);
    console.log(document.body);
    console.log(document.cookie);
    console.log(document.doctype);
    console.log(document.documentElement);
    console.log(document.documentURI);
    console.log(document.domain);
    console.log(document.embeds);
    console.log(document.forms);
    console.log(document.head);
    console.log(document.images);
    console.log(document.implementation);
    console.log(document.inputEncoding);
    console.log(document.lastModified);
    console.log(document.links);
    console.log(document.readyState);
    console.log(document.referrer);
    console.log(document.scripts);
    console.log(document.title);
    console.log(document.URL);
</script>
```

2.6. Creación e borrado de nodos:

No caso de querer crear ou borrar elementos existen os seguintes métodos no obxecto *document*:

- ***document.createElement(elemento)*:** crea ese nodo elemento correspondente.
- ***document.removeChild(elemento)*:** elimina o nodo elemento correspondente.
- ***document.appendChild(elemento)*:** engade o nodo elemento correspondente.
- ***document.textNode(texto)*:** crea un nodo de texto co contido indicado en *texto*.
- ***document.remove()***: elimina o nodo.
- ***document.replaceChild(elemNew, elemOld)*:** cambia o nodo *elemOld* polo nodo *elemNew*.

O proceso sempre será crear primeiro o nodo elemento, despois o nodo texto e finalmente engadir o nodo texto ao nodo elemento.

Dado o seguinte código HTML que queremos obter:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
        <a href="http://www.edu.xunta.gal">My link</a>
        <h1>My Header</h1>
    </body>
</html>
```

Partimos do texto sen ningún contido dentro de <body> e temos que crealo mediante JavaScript:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
    </body>
</html>
```

Primeiro crearemos os nodos elemento, engadiremos os correspondentes atributos, engadiremos os nodos texto neses nodos elemento e finalmente incluiremos os dous nodos elementos (`<a>`, `<h1>`) ao `<body>`.

Ademais, tamén se indica como intercambiar un nodo elemento por outro (cambio do `<h1>` co contido `My Header` por un `<h2>` co contido `My New Header`). Finalmente móstrase en comentarios como borrar dito `<h2>`.

```
<script>
    let nodoElem1 = document.createElement("a");
    nodoElem1.href = "http://www.edu.xunta.gal";
    let nodoElem1Texto = document.createTextNode("My link");
    nodoElem1.appendChild(nodoElem1Texto);
    let nodoElem2 = document.createElement("h1");
    let nodoElem2Texto = document.createTextNode("My Header");
    nodoElem2.appendChild(nodoElem2Texto);

    let elemento = document.getElementsByTagName("body")[0];
    elemento.appendChild(nodoElem1);
    elemento.appendChild(nodoElem2);

    //exemplo de intercambio dun <h1> por outro <h2>
    let nodoElem3=document.createElement("h2");
    let nodoElem3Texto = document.createTextNode("My New Header");
    nodoElem3.appendChild(nodoElem3Texto);
    let parent = document.getElementsByTagName("body")[0];
    let old = document.getElementsByTagName("h1")[0];
    parent.replaceChild(nodoElem3, old);

    //exemplo de borrado
    //nodoElem3.remove();
</script>
```