U3. Document Object Model

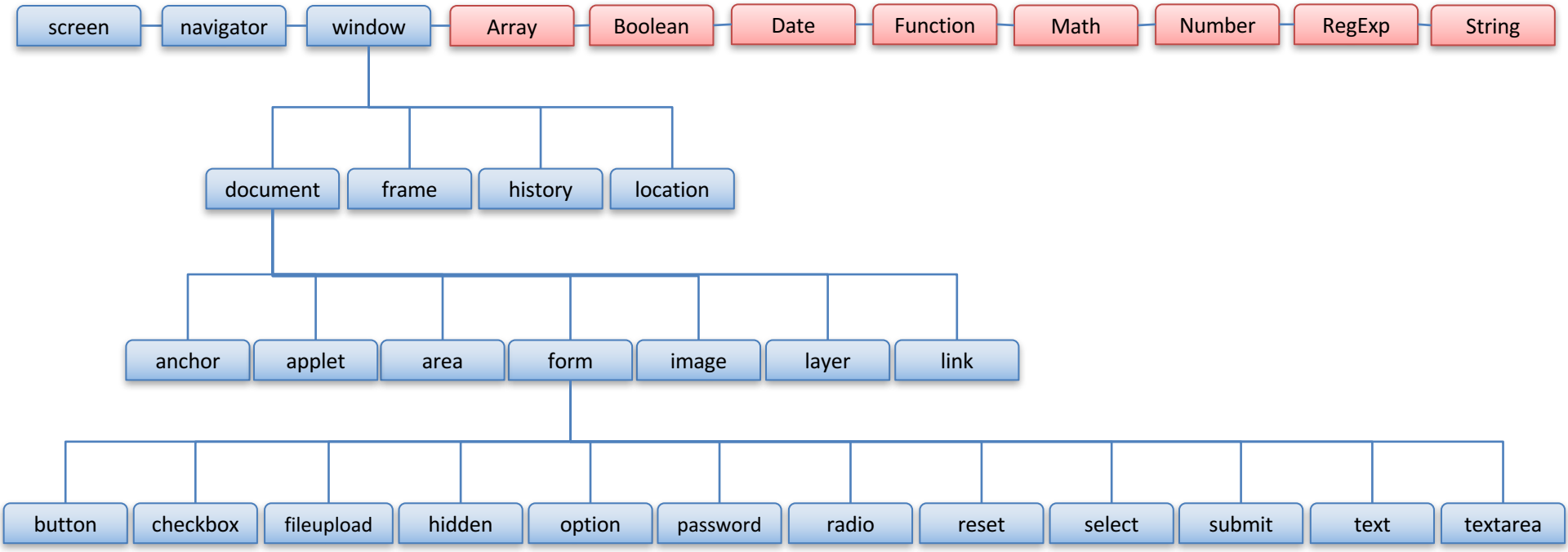
Desarrollo Web en Entorno Cliente

Contenidos

- ¿Qué es el DOM?
  - DOM
  - Estructura de árbol
  - Tipos de nodos DOM
- Acceso al DOM
  - Acceso a los nodos del DOM
  - Acceso a los atributos
- Modificar el DOM
  - Crear elementos
  - Añadir atributos
  - Borrar nodos

¿Qué es el DOM?

- Los objetos de JavaScript se ordenan de modo jerárquico.



Document Object Model (DOM)

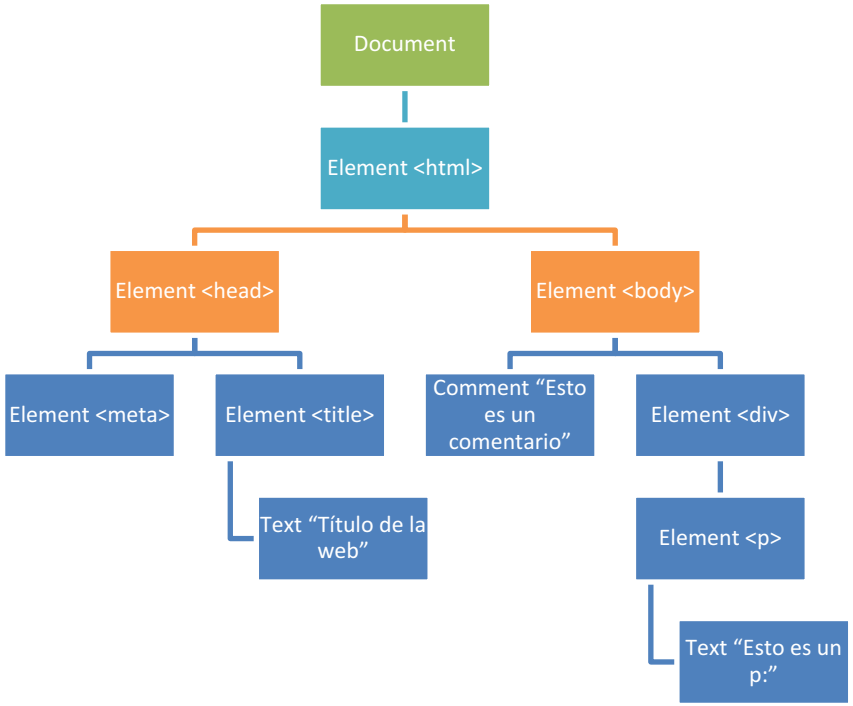
- Se trata de un estándar de W3C que define cómo acceder a documentos XHTML (HTML y XML) para **consultarlos** o **modificarlos** mediante una sintaxis.
- Cada navegador hace su propia implementación del DOM por lo que hay diferencias entre unos y otros.

Estructura del árbol DOM

Página

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Título de la web</title>
 </head>
 <body>
  <!—Esto es un comentario-->
  <div>
          <p>Esto es un p</p>
  </div>
 </body>
</html>
```

Árbol DOM



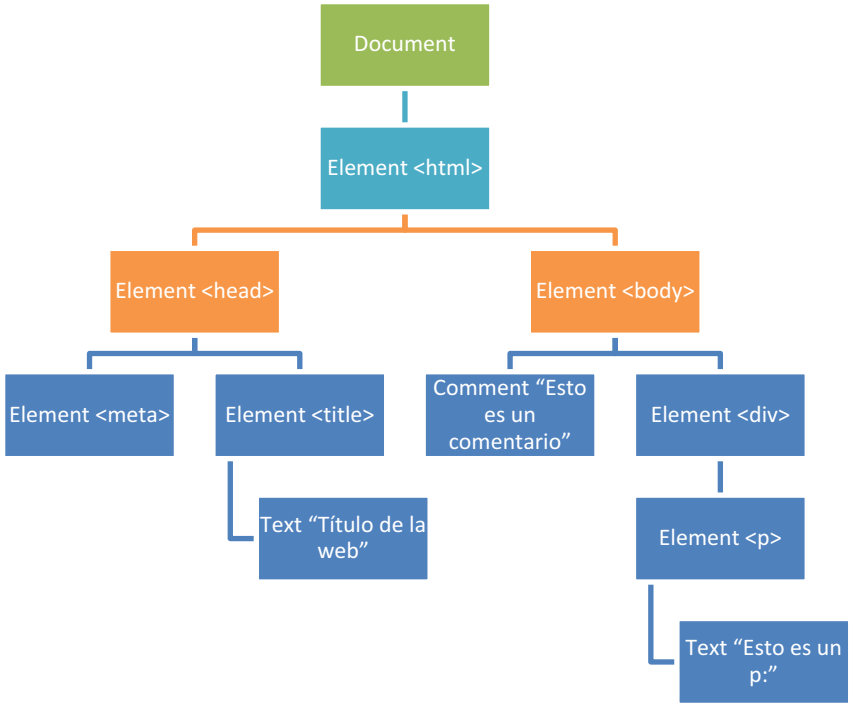
Estructura del árbol DOM

Explicación

- Cada etiqueta html que tiene un contenido se ha transformado en 2 nodos:
  - **Element:** la propia etiqueta
  - **Text:** el texto que la contiene

- Los nodos pueden tener descendientes que pasan a ser sus hijos. Podemos acceder a los hijos a partir del padre.

Árbol DOM



Tipos de nodos DOM

- Existen 12 tipos de nodos en el DOM. Los más habituales son:
  - **document:** es el nodo raíz del que cuelgan el resto y representa el documento XHTML.
  - **element:** representa las etiquetas XHTML. Puede contener atributos y otros nodos hijos.
  - **attr**: representa los atributos (pareja atributo=valor) de las etiquetas XHTML.
  - **text**: contiene el texto de los contenidos de una etiqueta HTML
  - **comment**: representa los comentarios de la página.

Acceso a los elementos del DOM

IMPORTANTE

No podemos hacer referencia al DOM si **no ha sido cargado por completo**. Para asegurarnos de que se carga podemos poner el script dentro de body o podemos emplear el evento onload.

- Document.title: permite acceder a <title>
- Document.body: permite acceder a <body>
  - nodeName: nombre del nodo (sólo lectura)
  - innerHTML: permite obtener el texto + etiquetas
  - innerText: permite obtener sólo el texto

Acceso a los elementos del DOM

- No todos los elementos son accesibles por su nombre. Algunos de los que sí nos permiten este acceso son:
  - Document.head (devuelve los elementos head)
  - Document.images (devuelve los elementos img)
  - Document.anchors (devuelve los elemento a)
  - Document.forms (devuelve los elementos form)
  - Document.links (devuelve los elementos con un href)

Acceso a los elementos del DOM

Página

```
<html>

  <head>
     <title>Título</title>
  </head>

  <body>
     <h1>DOM Lección 1</h1>
     <p>Hello world!</p>
  </body>

</html>
```

Árbol DOM

<head> tiene 1 hijo: <title>

<title> tiene un hijo (Text node): "Título“

<body> tiene 2 hijos: <h1> y <p>

<h1> tiene 1 hijo: "DOM Lección 1"

<p> tiene 1 hijo: "Hello world!"

<h1> y <p> son gemelos

Recorrer el árbol DOM

- Podemos recorrer el DOM empleando las etiquetas siguientes:
  - Acceso a padres: parentNode
  - Acceso a hijos: childNodes[númeroNodo], firstChild, lastChild, children (acceso solo a hijos tipo element).
  - Acceso a hermanos: nextSibling, previousSibling

- En el ejemplo anterior prueba

  document.body.childNodes

  document.body.childNodes[1]

  document.body.childNodes[1].nodeName

  document.body.childNodes[1].parentNode

  document.body.childNodes[1].nextElementSibling

Acceso a los elementos del DOM

- Acceso directo:
  - getElementsByTagName()

    JS:

    Var divs = Document.getElementsByTagName(“div”);

    Var primerDiv = divs[0];

  - getElementsByName()

    HTML

    ** … **

    JS:

    Var div1 = Document.getElementsByName(“primero”);

    Si hubiera varios elementos con el mismo name, sería div1[0], div1[1]…

Acceso a los elementos del DOM

- Acceso directo:
  - getElementsByClassName()

  HTML

  …

  JS

  var imp = document.getElementsByClassName("destacado");

  A partir de aquí accedemos a todos los elementos de la clase imp: imp[0], imp[1]…

Acceso a los elementos del DOM

- Acceso directo:
  - getElementById()

    **HTML: Esto es un p **

    JS:

    mip = Document.getElementById("destacado");

    mip.innerHTML = "Ahora el p es otro";
  - querySelector() permite seleccionar el primer elemento que coincida con el indicado mediante selectores CSS y querySelectorAll() todos.

Document.querySelector("p")
Document.querySelector("#id")
Document.querySelector(".class")

Acceso a los atributos

- Los atributos de los elementos XHTML se convierten en propiedades de los nodos del árbol DOM que podemos consultar y modificar.

```
HTML
<a id="google_link" href="http://www.google.com">Enlace</a>

JS
var glink = document.getElementById("google_link");
console.log (glink.href);
```

- getAttribute() permite consultar el valor de un atributo.

Acceso a los atributos

- De la misma manera, podemos acceder a las propiedades CSS a través de la propiedad style.
- Para acceder a las propiedades CSS, se hace una adaptación de los nombres con guión, convirtiéndolos en CamelCase.

```
HTML
<div id="menu">Enlace</a>

CSS
#menu{ background-color: #000000; color: #ffffff; }

JS
var divMenu = document.getElementById("menu");
console.log (divMenu.style.color);
console.log (divMenu.style.backgroundColor);
```

Modificar el DOM

- Podemos añadir elementos al árbol de la siguiente manera:
  - **createElement(),** crea un nodo tipo elemento
  - **createTextNode(),** crea un nodo tipo texto
  - elementoPadre.**appendChild**(elementoHijo), cuelga el nodo elementoHijo del nodo elementoPadre

```
var paragraf = document.createElement("p");
var txt= document.createTextNode("Hola!");
paragraf.appendChild(txt);
document.body.appendChild(paragraf);
```

Modificar el DOM

- También es posible añadir atributos empleando **setAttribute**
- La sintaxis es:

    setAttribute(nombreClase, valorClase)

```
document.getElementsByTagName("body")[0].setAttribute("class", "exemple");
```

Modificar el DOM

- También podemos suprimir nodos empleando removeChild.
- removeChild() se debe ejecutar desde el nodo padre del elemento que queremos borrar.

```
var paragraf = document.getElementById("p1");
paragraf.parentNode.removeChild(paragraf);
```