DOM+HTML+JS (parte 2)

JUAN CARLOS CONDE RAMÍREZ

WEB-TECHNOLOGIES

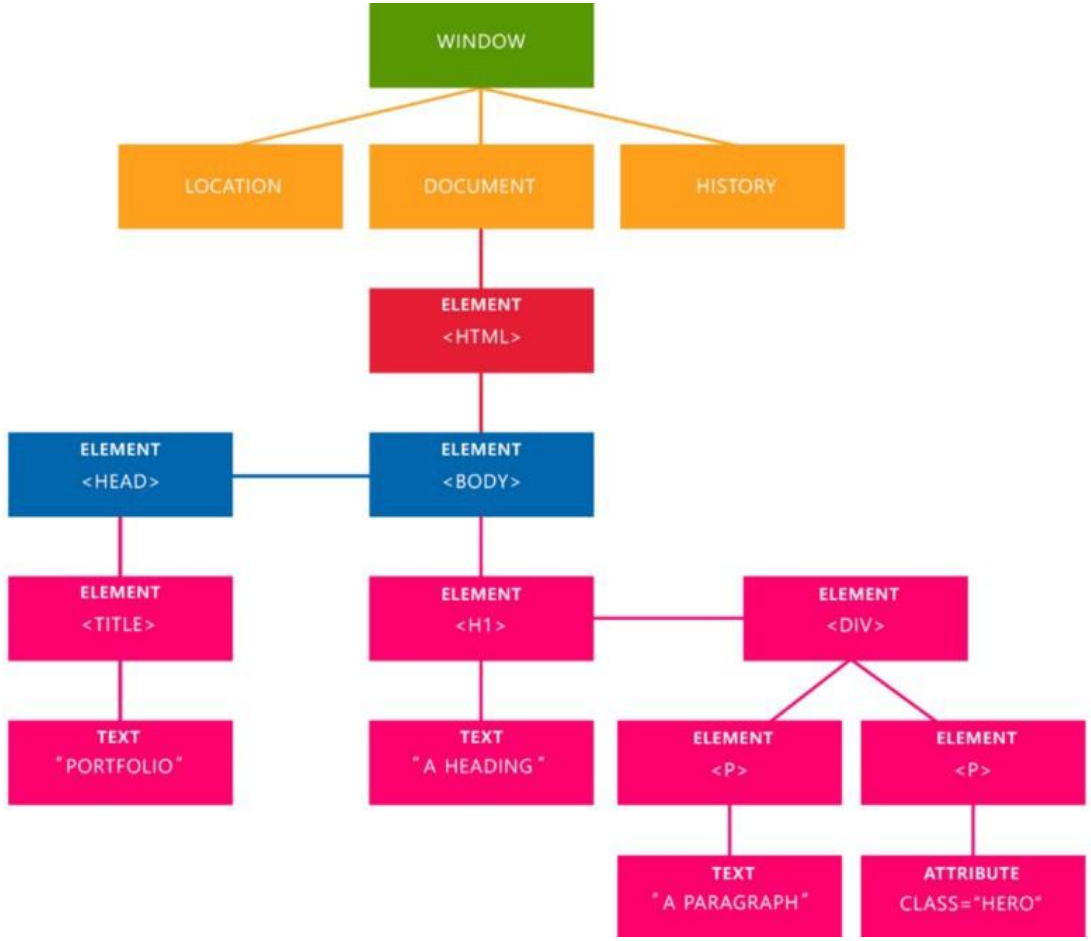
Tipos de datos importantes, I

- A continuación se intentará describir, de la manera más simple posible, los diferentes objetos y tipos.

- Una cantidad de tipos de datos diferentes son utilizados por el API.

- Para simplificarlo, en los ejemplos nos referimos a los *nodos* como `elements`, y a una lista de nodos como `nodeLists` (o simples elementos) y a atributos de nodos como `attributes`.

Jerarquía de Objetos de DOM



Tipos de datos importantes, II

`document`

- Cuando un miembro devuelve un objeto del tipo `document` (por ejemplo, la propiedad `ownerDocument` de un elemento devuelve el documento "document" al cual pertenece), este objeto es la raíz del objeto documento en sí mismo.

`element`

- `element` se refiere a un elemento o a un nodo de tipo de elemento *"element"* devuelto por un miembro del API de DOM. Dicho de otra manera, por ejemplo, el método `document.createElement()` devuelve un objeto referido a un nodo, lo que significa que este método devuelve el elemento que acaba de ser creado en el DOM.
- Los objetos `element` ponen en funcionamiento a la interfaz *Element* del DOM y también a la interfaz "*Node*" más básicas.

Tipos de datos importantes, III

`nodeList`

- Una "nodeList" es una serie de elementos, parecido a lo que devuelve el método `document.getElementsByTagName()`. Se accede a los *items* de la `nodeList` de cualquiera de las siguientes dos formas:
  - `list.item(1)`
  - `lista[1]`
- Ambas maneras son equivalentes. En la primera, `item()` es el método del objeto `nodeList`. En la última se utiliza la típica sintaxis de acceso a listas para llegar al segundo ítem de la lista.

`attribute`

- Cuando un atributo es devuelto por un miembro (por ej., por el método `createAttribute()`), es una referencia a un objeto que expone una interfaz particular (aunque limitada) a los atributos.
- Los atributos son nodos en el DOM igual que los elementos, pero no suelen usarse así.

Tipos de datos importantes, IV

`NamedNodeMap`

- Un `namedNodeMap` es una serie, pero los ítems son accesibles tanto por el nombre como por un índice, este último caso es meramente una conveniencia para enumerar ya que no están en ningún orden en particular en la lista.
- Un `NamedNodeMap` es un método de `item()` por esa razón, y permite poner o quitar ítems en un `NamedNodeMap`.

Interfaces del DOM, I

- Uno de los propósitos de esta guía es minimizar el hablar de *interfaces abstractas*, *heredadas* y otros detalles de funcionamiento.

- Más bien, concentrarse sobre los objetos en el DOM y sobre las herramientas que se pueden usar para manipular la jerarquía de DOM.

- Desde el punto de vista del programador web, es bastante indiferente saber que la representación del objeto del elemento *HTML form* toma la propiedad `nodeName` desde la interfaz `FormElement` o que las propiedades `className` se toman desde la propia interfaz `Element`. En ambos casos, la propiedad está sólo en el objeto `form`.

Interfaces del DOM, II

- Pero puede resultar confuso el funcionamiento base de la relación entre objetos e interfaces en el DOM, por eso esta sección intentará hablar un poquito sobre las interfaces actuales en la especificación del DOM y de como se dispone de ellas.

Interfaces y objetos, I

- En algunos casos un objeto pone en ejecución a una sola interfaz. Pero a menudo un objeto toma prestada una tabla HTML (*table*) desde muchas interfaces diversas.

- El objeto `table`, por ejemplo, pone en funcionamiento una Interfaz especial del elemento *table HTML*, la cual incluye métodos como `createCaption` y `insertRow`. Pero como también es un elemento HTML, `table` pone en marcha a la interfaz de Element.

- Al final, se trata de un nodo en el árbol de DOM de una página web o XML; el elemento `table` hace funcionar la interfaz más básica de *Node*, desde el cual deriva *Element*.

Interfaces y objetos, II

- La referencia a un objeto `table`, como en el siguiente ejemplo, utiliza estas interfaces intercambiables sobre el objeto:

```
var table = document.getElementById("table");
var tableAttrs = table.attributes; // Node/interfaz Element
for (var i = 0; i < tableAttrs.length; i++) {
  // interfaz HTMLTableElement: atributo border
  if(tableAttrs[i].nodeName.toLowerCase() == "border")
    table.border = "1";
}
// interfaz HTMLTableElement: atributo summary
table.summary = "nota: borde aumentado";
```

Interfaces esenciales de DOM, I

- La idea no es describir qué hacen toda el API pero sí dar una idea de las clases de métodos y propiedades relacionadas con el uso del DOM. Muchos ejemplos de uso común de esta API se puede consultar en: Ejemplos DOM.

- `document` y `window` son objetos cuya *interfaces* son generalmente muy usadas en la programación de DOM.

- En términos simples, el objeto `window` representa componentes como los podría ser el navegador, y el objeto `document` es la raíz del *documento* en sí.

Interfaces esenciales de DOM, II

- *Element* hereda de la interfaz genérica *Node*, estas dos interfaces proporcionan muchos métodos y propiedades utilizables sobre los elementos individuales.

- Éstos elementos pueden igualmente tener interfaces específicas según el tipo de datos representados, como en el ejemplo anterior del objeto `table`.

- Lo siguiente es una breve lista de los métodos y propiedades comunes en la Web y en las páginas escritas en XML que utilizan DOM.

Interfaces esenciales de DOM, III

Métodos

- document.getElementById(id)
- *element*.getElementsByTagName(name)
- document.createElement(name)
- *element*.appendChild(*element*)

Propiedades:

- *element*.innerHTML
- *element*.style.left
- *element*.setAttribute
- *element*.element.getAttribute
- *element*.addEventListener
- window.content
- window.onload
- window.dump
- window.scrollTo

Probando el API del DOM, I

- Para probar y ver como trabajan en la plataforma del navegador las interfaces del DOM, a continuación se muestra una página de prueba. Como es de suponerse, el contenido de la función se puede actualizar según la necesidad, para crear más botones o poner más elementos.

```
<html>
  <head>
    <title>Pruebas DOM</title>
    <script type="application/javascript">
    function setBodyAttr(attr, value){
      if (document.body) eval('document.body.'+attr+'="'+value+'"');
      else notSupported();
    }
    </script>
  </head>
```

Probando el API del DOM, II

```
<body>
    <div style="margin: .5in; height: 400;">
      <p><b><tt>texto</tt></b></p>
      <form>
        <select onChange="setBodyAttr('text',
        this.options[this.selectedIndex].value);">
          <option value="black">negro
          <option value="darkblue">azul oscuro
        </select>
        <p><b><tt>bgColor</tt></b></p>
        <select onChange="setBodyAttr('bgColor',
        this.options[this.selectedIndex].value);">
          <option value="white">blanco
          <option value="lightgrey">gris
        </select>
```

Probando el API del DOM, III

```
      <p><b><tt>link</tt></b></p>
      <select onChange="setBodyAttr('link',
      this.options[this.selectedIndex].value);">
        <option value="blue">azul
        <option value="green">verde
      </select>
      <small>
        <a href="http://www.brownhen.com/dom_api_top.html" id="sample">(sample link)</a>
      </small><br>
    </form>
    <form>
      <input type="button" value="version" onclick="ver()" />
    </form>
  </div>
 </body>
</html>
```

Probando el API del DOM, III

Probando el API del DOM, IV

- En este ejemplo, los menús desplegables actualizan dinámicamente componentes de la página Web accesibles desde DOM como: el color de fondo (*bgColor*), de los hiperenlaces (*aLink*) y el texto (*text*).

- El hecho de diseñar páginas y probar las interfaces son una parte importante del aprendizaje para un uso eficaz del DOM.

[Obtenido de *MDN web docs*]

Otras referencias…

- Referencia DOM
- Introducción al DOM
- Eventos y el DOM
- Ejemplos