CFGS DAW

Mòdul 6: Desenvolupament web en entorn client

INS Joan d'Àustria

Extensiones DOM

A partir de las características DOM vistas, muchos navegadores desarrollaron sus propias extensiones para ampliar la funcionalidad.

A partir de 1998, el W3C incorporó y estableció dentro del estandard algunas de estas extensiones. De todas estas destacan:

- La API de selectores
- HTML5

Extensiones DOM

LA API DE SELECTORES:

Una de las principales características de las librerias javascript es la de poder seleccionar diferentes elementos del DOM a partir de un patrón determinado.

En concreto se observó que una necesidad era la de poder seleccionar elementos a partir de patrones CSS. Así aparecieron los metodos querySelector y querySelectorAll

Extensiones DOM

LA API DE SELECTORES:

querySelector recibe como parámetro una consulta CSS y retorna el primer elemento hijo que cumple con el patrón o null si no encuentra ninguno

```
//acceder a un elemento, por ejemplo a body
var body = document.querySelector("body");

//acceder a un elemento con ID "myDiv"
var myDiv = document.querySelector("#myDiv");

//acceder al primer elemento de la clase "myClass"
var selected = document.querySelector(".myClass");
```

Extensiones DOM

LA API DE SELECTORES:

querySelectorAll recibe como parámetro una consulta CSS y retorna todos los elementos hijo que cumple con el patrón o null si no encuentra ninguno

```
//obtener todos los <p> dentro del div “myDiv”
var pes= document.getElementById(“myDiv”).querySelectorAll(“p”);

//obtener todos los elementos de la clase  “selected”
var selecteds = document.querySelectorAll(“.selected”);

//obtener todos los elementos <strong> elements dentro de elementos <p>
var strongs = document.querySelectorAll(“p strong”);
```

Extensiones DOM

HTML5:

Una de las características de HTML5 es el mayor uso de clases para indicar aspectos estilísticos y semánticos. También se han realizado ampliaciones y extensiones para tratar diferentes aspectos

Extensiones DOM

HTML5:

getElementsByClassName recibe como parámetro un string con el nombre de una o más clases y retorna una lista de nodos que corresponden a todas las clases indicadas

```
//acceder a los elementos de la clase “pie”
var pies= document.getElementsByClassName(“pie”);

//acceder a los elementos de la clase “pie” dentro de “myDiv”
var pies= document.getElementById(“myDiv”).getElementsByClassName(“pie”);
```

Extensiones DOM

HTML5:

Para gestionar el foco, se ha desarrollado:

document.activeElement: contiene un puntero al elemento que tiene el foco

```
var button = document.getElementById(“myButton”);
button.focus();
alert(document.activeElement === button); //true
```

Extensiones DOM

HTML5:

Para conocer el estado de carga, se ha desarrollado:

document.readyState: retorna 2 posibles valores: loading indicando que el documento se está cargando o complete indicando que ya se ha cargado

```
if (document.readyState == “complete”){
        //do something
}
```

Extensiones DOM

HTML5:

Para gestionar y modificar los nodos de forma más sencilla se han desarrollado:

- innerHTML: permite consultar o modificar todo el contenido de un elemento en formato HTML en lugar de lista de nodos

```
<div id="ejemplo">
<p>Lista ejemplo</p>
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
</div>
```

```
var x=document.getElementById('ejemplo');
alert (x.innerHTML);
```

```
<p>Lista ejemplo</p>
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
```

Extensiones DOM

HTML5:

- outerHTML: funciona igual que innerHTML pero retorna o sobreescribe también el propio elemento

```
<div id="ejemplo">
<p>Lista ejemplo</p>
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
</div>
```

```
var x=document.getElementById('ejemplo');
alert (x.outerHTML);
```

```
<div id="ejemplo">
<p>Lista ejemplo</p>
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
</div>
```

Extensiones DOM

HTML5:

- insertAdjacentHTML: acepta como parámetros la posición y el nuevo texto HTML. La posición puede ser:
  - beforebegin: se introducirá justo antes del elemento como si fuera previousSibling
  - afterbegin: se introducirá dentro del elemento como primer hijo (o serie de hijos)
  - beforeend: se introducirá dentro del elemento como ultimo hijo (o serie de hijos)
  - Afterend: se introducirá justo después del elemento como si fuera nextSibling

Extensiones DOM

HTML5:

```
<style>
div {
color:red;
}
</style>
<div id=“ejemplo">
<p>Lista ejemplo</p>
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
</div>
```

```
var x=document.getElementById('content');
x.insertAdjacentHTML("beforebegin", "<p>Hola mundo!</p>");
```

```
var x=document.getElementById('content');
x.insertAdjacentHTML(“afterbegin", "<p>Hola mundo!</p>");
```

```
var x=document.getElementById('content');
x.insertAdjacentHTML("beforeend", "<p>Hola mundo!</p>");
```

```
var x=document.getElementById('content');
x.insertAdjacentHTML(“afterend", "<p>Hola mundo!</p>");
```

Extensiones DOM

HTML5:

Para tratar el tema del scroll se ha desarrollado:

- scrollIntoView: se puede aplicar a todos los elementos HTML y hace que el elemento sea visible en el viewport. Recibe un parámetro (por defecto es true) que puede ser:
  - True: el top del elemento coincide con el top del viewport
  - False: el elemento es visible pero no necesariamente en el top del viewport, por lo que si el elemento es grande, puede que el inicio del elemento no sea visible

```
document.forms[0].scrollIntoView(true);
```

Extensiones DOM

OTRAS EXTENSIONES PROPIETARIAS:

Cada navegador ha desarrollado otras extensiones, pero no forman parte del standard. Por ejemplo en Safari y Chrome podemos encontrar:

- scrollIntoViewIfNeeded
- scrollByLines
- scrollByPages

Microsoft por su parte dispone de:

- innerText
- outerText