Javascript

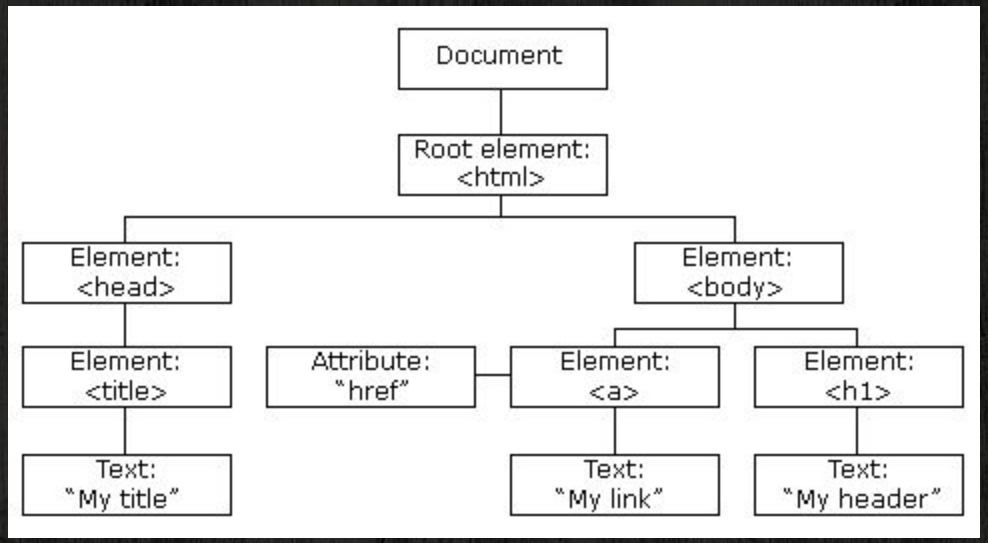
Programa del curso

- Clase 1: Introducción y sintaxis
- Clase 2: Funciones, arrays y objetos
- Clase 3: Javascript integrado a HTML
- **Clase 4: DOM, selectores y elementos**
- Clase 5: Eventos
- Clase 6: Formularios
- Clase 7: Ajax
- Clase 8: Ejercicio integrador

1. DOM (Document Object Model)

DOM - Definición

El **document object model (DOM)** proporciona una representación estructural de un documento html.



DOM - Definición

Con el DOM a su alcance Javascript tiene el poder para:

- Modificar elementos, atributos y estilos de una página.
- Borrar cualquier elemento y atributo.
- Agregar nuevos elementos o atributos.
- Reaccionar a todos los eventos HTML de la página.
- Crear nuevos eventos HTML en la página.

2.
Selectores

Selectores

Para acceder a los **elementos** de una página utilizamos selectores.

Es posible acceder utilizando los nodos, pero puede ser un poco confuso.

Cada selector puede retornar un **solo elemento** o una **lista**.

El objeto document tiene los siguientes selectores como método:

- **document.getElementById(ID)**
- **document.getElementsByClassName(clase)**
- **document.querySelector(cssQuery)**
- **document.querySelectorAll(cssQuery)**

document.getElementById()

```
<html>
<head>
<script>
window.onload = function() { // Carga la página y luego ejecuta:
    document.getElementById("titular").style.display = 'none';
}
</script>
</head>
<body>
    <h1 id="titular">Bienvenido!</h1>
</body>
</html>
```

document.querySelector()

```
<html><head><script>
window.onload = function() { // Carga la página y luego ejecuta:
    document.querySelector(".rojo").style.display = 'none';
}
</script>
</head>
<body>
    <h1 class="rojo">Bienvenido!</h1>
    <p class="rojo">Esto es un texto, me gusta el rojo</p>
</body></html>
```

querySelector() selecciona el **primer** elemento que cumpla la condición.

document.querySelectorAll()

```
<head>
<script>
window.onload = function() { // Carga la página y luego ejecuta:
      document.querySelectorAll(".rojo").forEach(
            function(element){
                  element.style.color = 'red';
      });
}
</script>
</head>
<body>
      <h1 class="rojo">Bienvenido!</h1>
      <p class="rojo">Esto es un texto, me gusta el rojo</p>
</body>
```

2. Modificar elementos

Atributos

En cuanto tenemos seleccionado un elemento podemos acceder a sus atributos utilizando la propiedad **attributes**.

Retorna un mapa (es como un array) que tiene valores key/value con los nombres y valores de los atributos de ese elemento.

```
elemento.attributes;
```

getAttribute() / setAttribute()

El método **getAttribute** acepta un string como parámetro con el nombre del atributo que quiero obtener. Retorna el valor del atributo y null en caso de no encontrarlo.

```
elemento.getAttribute("href");
```

El método **setAttribute** permite agregar un nuevo atributo o modificar uno existente.

```
elemento.setAttribute("class","rojo"):
```

hasAttribute() / removeAttribute()

El método **hasAttribute** acepta un string como parámetro con el nombre del atributo que quiero saber si existe en ese elemento.

Retorna un valor booleano.

```
elemento.hasAttribute(nombreAtributoEnString);
```

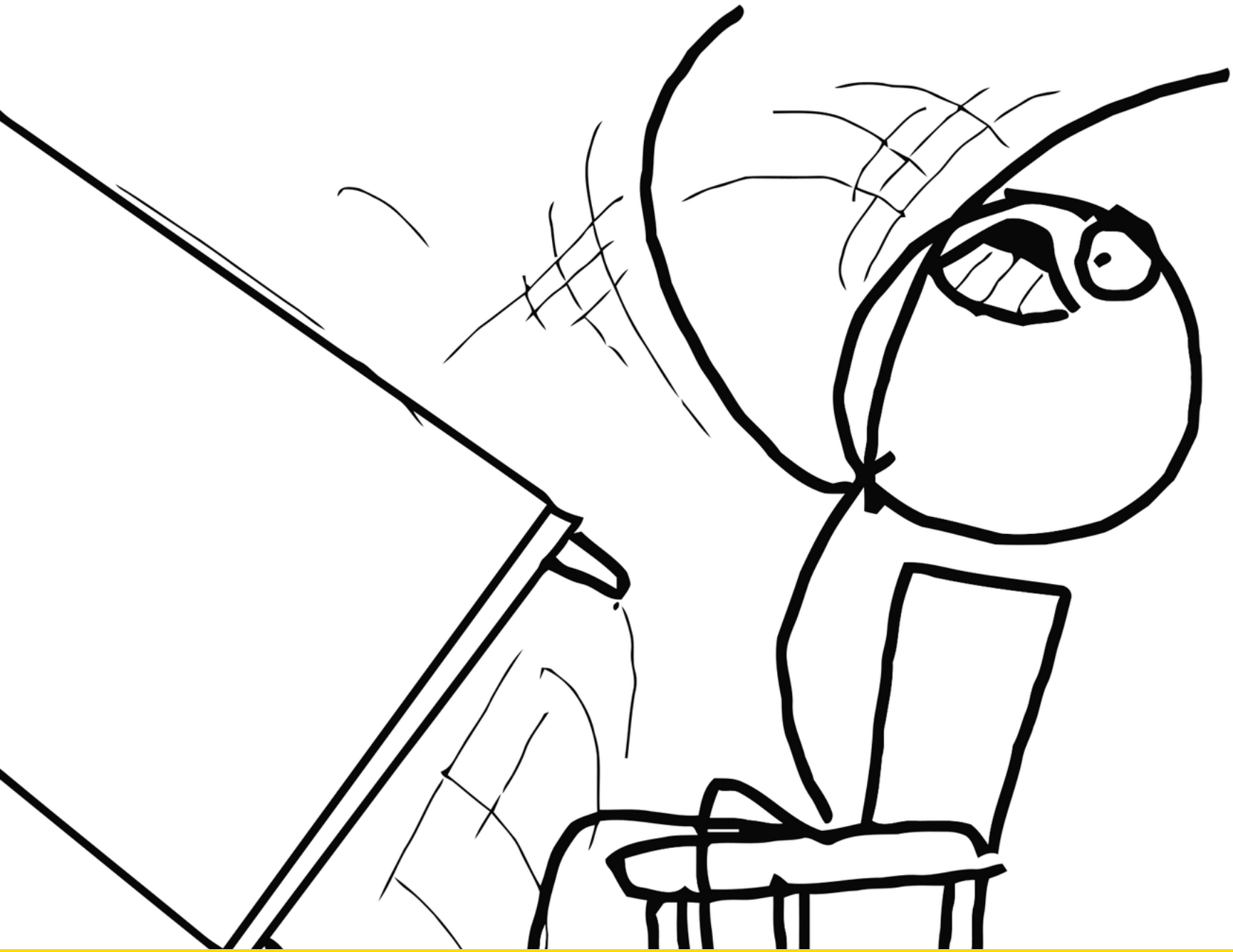
Con el método **removeAttribute** podemos remover un atributo existente.

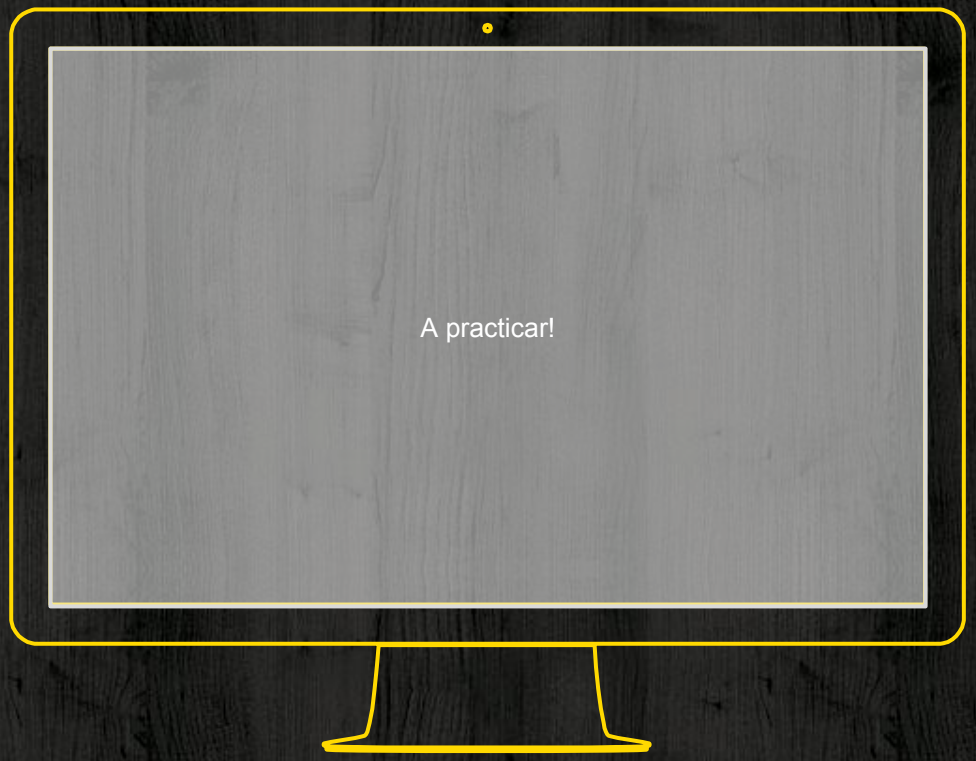
```
elemento.removeAttribute(nombreAtributo):
```

Estilos

- Los elementos HTML tienen una propiedad llamada *style* que retorna un objeto literal que representa los estilos que tiene ese objeto.
- Podemos agregar o modificar sus atributos.
- Los nombres de las propiedades de CSS en Javascript se escriben con el siguiente formato: **nombreDePropiedadDeCss**.

```
elemento.style.color = "red"; // seteo el color a rojo
elemento.style.fontWeight = "bold"; // seteo el weight a bold
```





A practicar!

Práctica 1 - **DOM - Selectores y Atributos**

3.
Elemento

Creando elementos - createElement

El método **createElement** nos permite crear nuevos elementos HTML.

createElement acepta como parámetro un string con el nombre de una etiqueta de HTML (a, div, span, li, ul, etc).

```
var btn = document.createElement("BUTTON");
```

El método **createTextNode** nos permite crear nuevos textos HTML.

```
var texto = document.createTextNode("Hola soy un texto");
```

Insertando elementos - appendChild

appendChild nos permite insertar un nodo dentro de otro.

```
<script>
	var node = document.createElement("LI");
	var textnode = document.createTextNode("Water");
	node.appendChild(textnode);
	document.getElementById("myList").appendChild(node);
</script>
<body>
	<div id="myList"></div>
</body>
```

textContext / innerHTML

textContent nos permite leer o escribir contenido en modo de **texto**.

```
elemento.textContent = "texto";

elemento.textContent;       // texto
```

innerHTML nos permite leer o escribir contenido en **HTML** de un nodo.

```
<div id="titular"></div>

var elemento = document.getElementById("titular");

elemento.innerHTML = "<h1>Mi elemento HTML</h1>";
```

Resultado:

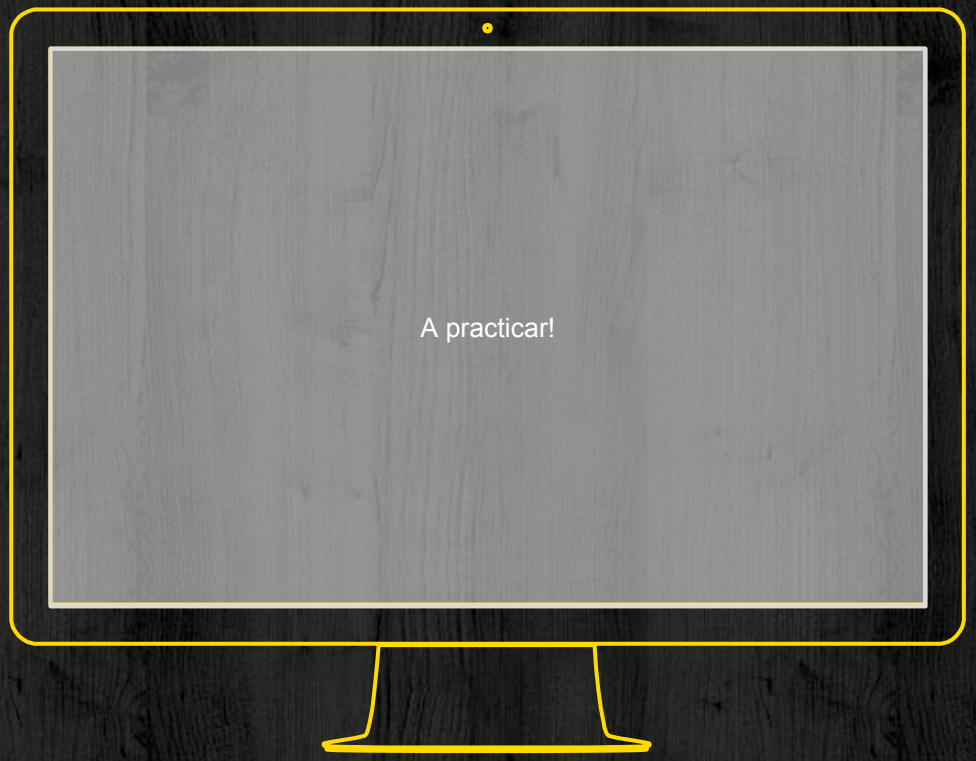
```
<div id="titular"><h1>Mi elemento HTML</h1></div>
```

Removiendo elementos - removeChild

El objeto nodo tiene un método llamado **removeChild** que nos permite remover nodos hijos.

Para poder remover un nodo tenemos que primero seleccionarlo.

```
var elemento = document.getElementById(ID);
var nodo = elemento.children.item(nroItem);
elemento.removeChild(nodo);
```



A practicar!

Práctica 2 - **Elementos**

Gracias!

Preguntas?