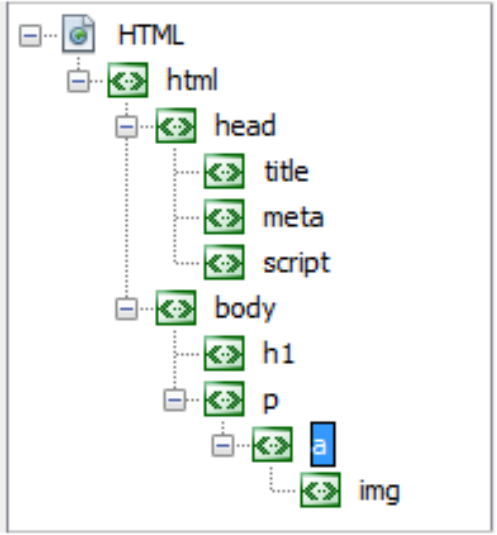
Document Object Model – DOM –

INTERACCIÓN CON EL NAVEGADOR (DOM)

El objeto DOM

- API (Application Programming Interface) para HTML y XML
- **Representación estructurada** del documento (página web) que permite a los lenguajes de programación modificar la estructura, los estilos o el contenido del mismo documento
- El DOM ofrece una representación del documento como un grupo de nodos y objetos que tienen **propiedades** y **métodos**.
- Conecta las páginas web a los scripts o lenguajes de programación



https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

https://developer.mozilla.org/en-US/docs/Web/API

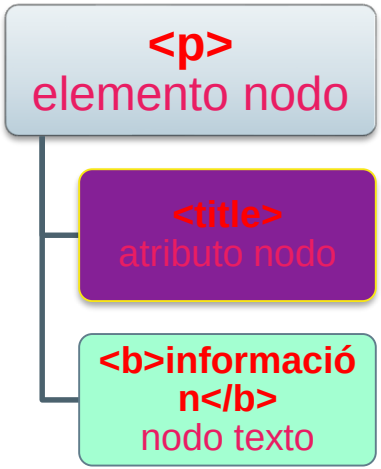
Conociendo el objeto DOM

El árbol DOM está compuesto de un conjunto de “piezas”, como:

- Nodos
- Elementos nodo
- Elemento texto
- Atributos nodo

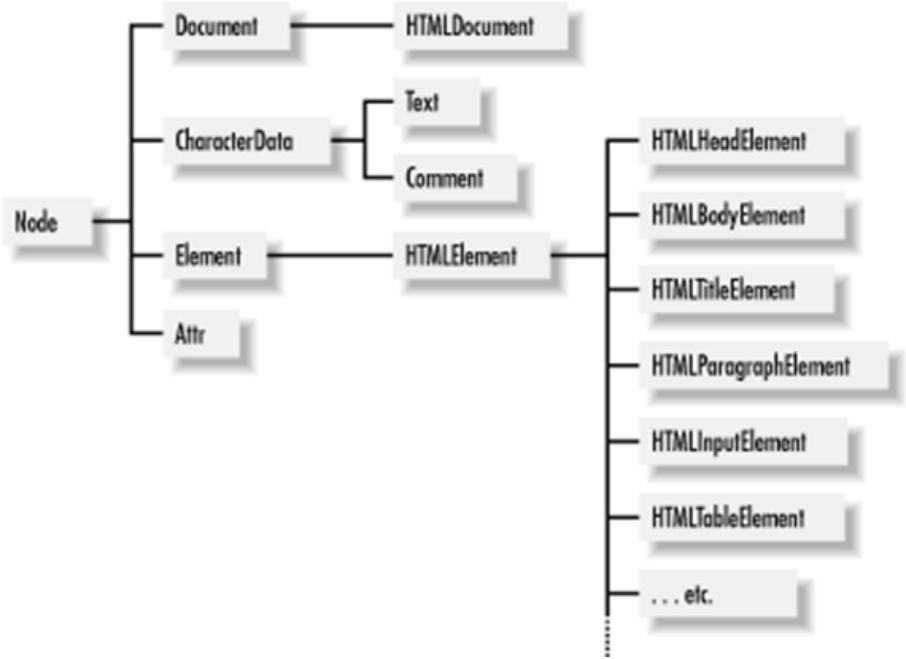
El objeto DOM está compuesto de una colección de nodos, existen diferentes tipos de nodos, algunos nodos contienen a su vez otros nodos.

Los elementos HTML, son en sí mismos nodos, aunque no contengan información.



Conociendo el objeto DOM

Tipos habituales de nodos



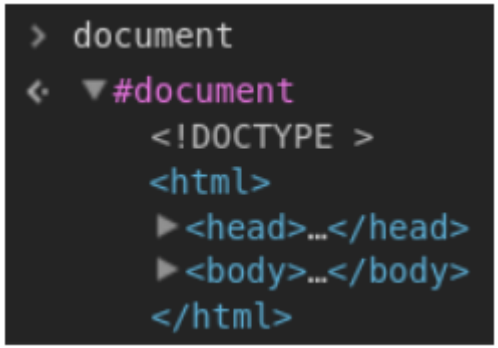
Pongámoslo en práctica

Genera una página web HTML5 que contenga:

```
<h1>Dog</h1>
<div id="cat"></div>
<div class="mouse"></div>
<div class="mouse"></div>
<div class="mouse"></div>
```

DOM – Objeto document

- El DOM es accesible vía **window.document** o simplemente **document**
- Es instanciado automáticamente cuando se renderiza la página.
- Se puede acceder via consola tecleando: **document**



Manipular el DOM

- Para manipular el DOM es necesario, en primer lugar, obtener el elemento a manipular. Para ello existen distintos métodos:
  - Buscar elementos por su ID: document.**getElementById('cat')**
  - Buscar elementos por el atributo class: document. **getElementsByClassName('cat')**
  - Buscar elementos por el tipo: document. **getElementsByTagName('cat')**
  - Acceder al primer selector: document. **querySelector(selector)**
  - Lista de todos los selectores que coinciden con el grupo especificado: document. **querySelector(selector)**

Creando elementos DOM

Método **createElement():**

```
let newNode = document.createElement(<node_name>);
node.appendChild(newNode);
```

Con **innerHTML**

```
node.innerHTML = <html>;
```

Cambiando estilos

- Podemos modificar los estilos de un elemento usando Javascript
- Existen dos aproximaciones:
  - Acceder a las propiedades de estilo mediante **style**

```
Node.style.[color|height|width|border...]=<valor>;
```

  - Modificar la clase del estilo (preferida)

```
Node.className =<class_list>;
let elementClasses = Node.classList.[add()|remove()|contains()];
```

- http://www.w3schools.com/jsref/dom_obj_style.asp

Pongámoslo en práctica

Para una página web:

- Crea un css básico para darle estilo a tu página
- Cambia los estilos del título del artículo:
    - Usando la propiedad style
    - Asignándole una clase
- Añade una animación al css
    - http://www.minimamente.com/example/magic_animations/

EVENTOS

Eventos en Javascript

- Las aplicaciones web creadas con JavaScript pueden utilizar el modelo de programación basada en eventos.

- En este tipo de programación, los scripts se dedican a esperar a que el usuario "haga algo" (que pulse una tecla, que mueva el ratón, que cierre la ventana del navegador).

- A continuación, el script responde a la acción del usuario normalmente procesando esa información y generando un resultado.

Definir un evento

- En JS podemos asociar un evento a cualquier elemento del DOM
- Existen distintas aproximaciones:
  - En línea en la etiqueta de HTML que se quiera:

```
<input type="button" value="Calcula" onClick=“calcular(2,3,4,5)">
```

    - Es importante escapar los caracteres. Por ejemplo si es un string usar comillas simples ' ‘.
    - Este método de asociación no se recomienda! Ya que acopla HTML y JS

  - Mediante una función de **callback**:

```
boton.onclick = function(e){…}
```

  - Mediante un **listener:**

```
boton.addEventListener('click', function(){…}, false);
```

Definir un evento – atributo de evento

- Podemos programa un evento asociación con funciones anónimas de callback.
- Estas se llamarán cuando suceda el evento

```
<button id="mi_boton">Enviar</button>
```

```
document.getElementById('mi_boton').onclick = function(e){
                    ....
}
```

- Cuando asociamos un evento a un elemento, podemos acceder al propio elemento dentro de la función de callback con la palabra reservada **this**

```
<div id="mi_div">Texto del div</ div >
```

```
document.getElementById('mi_div').onclick = function(e){
                    let elcuerpo= this.innerHTML; //devolverá el cuerpo del propio div
                    ...
}
```

Modelo básico de eventos

Atributo	Descripción	IE	F	O	W3C
onblur	An element loses focus	3	1	9	Yes
onchange	The content of a field changes	3	1	9	Yes
onclick	Mouse clicks an object	3	1	9	Yes
ondblclick	Mouse double-clicks an object	4	1	9	Yes
onerror	An error occurs when loading a document or an image	4	1	9	Yes
onfocus	An element gets focus	3	1	9	Yes
onkeydown	A keyboard key is pressed	3	1	No	Yes
onkeypress	A keyboard key is pressed or held down	3	1	9	Yes
onkeyup	A keyboard key is released	3	1	9	Yes
onmousedown	A mouse button is pressed	4	1	9	Yes
onmousemove	The mouse is moved	3	1	9	Yes
onmouseout	The mouse is moved off an element	4	1	9	Yes

Modelo básico de eventos

Atributo	Descripción	IE	F	O	W3C
onmouseover	The mouse is moved over an element	3	1	9	Yes
onmouseup	A mouse button is released	4	1	9	Yes
onresize	A window or frame is resized	4	1	9	Yes
onselect	Text is selected	3	1	9	Yes
onunload	The user exits the page	3	1	9	Yes

Pongámoslo en práctica

Asocia un evento click a botones: colorea, blanco-negro, normal, que harán los efectos respectivos cambiando el aspecto de un elemento section.

Pongámoslo en práctica

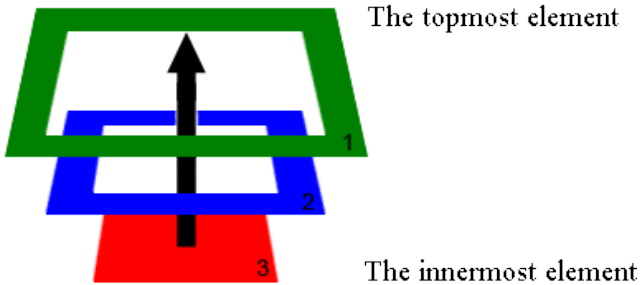
Asocia un evento al botón de enviar comentario del formulario, debe llamar a una función que:

- Verifique que cada campo tiene valor
- Recoja y concatene los valores de los campos en un único string y haga alert del mismo.
- Si hay un error, que lo indique en un alert

* Si hay un error, poner el campo enmarcado en rojo.

Event Bubbling

- En este modelo de flujo de eventos se produce primero el evento en el elemento más interno de la estructura de árbol y va subiendo jerárquicamente hasta llegar al nodo raíz



- Para evitar la propagación usar el método stopPropagation sobre el evento

```
elem.onclick=function(evnt){
    …
     evnt.stopPropagation();
}
```

Atributos de evento para ratón y teclado

- Detectando eventos de ratón y teclado, según versiones

Atributo	Descripción	IE	F	O	W3C
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	6	1	9	Yes
button	Returns which mouse button was clicked when an event was triggered	6	1	9	Yes
clientX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
clientY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered	6	1	9	Yes
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered	6	1	9	Yes

Atributos de evento para ratón y teclado

- Detectando eventos de ratón y teclado, según versiones

Atributo	Descripción	IE	F	O	W3C
relatedTarget	Returns the element related to the element that triggered the event	No	1	9	Yes
screenX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
screenY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	6	1	9	Yes

Otros atributos de eventos

- Resto de atributos de eventos

Atributo	Descripción	IE	F	O	W3C
bubbles	Returns a Boolean value that indicates whether or not an event is a bubbling event	No	1	9	Yes
cancelable	Returns a Boolean value that indicates whether or not an event can have its default action prevented	No	1	9	Yes
currentTarget	Returns the element whose event listeners triggered the event	No	1	9	Yes
eventPhase	Returns which phase of the event flow is currently being evaluated				Yes
target	Returns the element that triggered the event	No	1	9	Yes
timeStamp	Returns the time stamp, in milliseconds, from the epoch (system start or event trigger)	No	1	9	Yes
type	Returns the name of the event	6	1	9	Yes

Listeners de eventos

- Los métodos addEventListener y removeEventListener, permiten vincular eventos a objetos dinámicamente

```
function holaMundo() { alert('Hola mundo.'); }

let boton = document.getElementById('boton');

boton.addEventListener('click', holaMundo, false);
boton.removeEventListener('click', holaMundo, false);

<input type="button" value="Pulsa" id="boton" />
```

Validar formularios con JavaScript

- JavaScript reduce la carga en el servidor ya que si se envían los datos ya validados, el servidor solo tiene que validarlos una vez y no tiene que estar mandando páginas de error constantemente.
- Los datos han de validarse también en el servidor.
- Se pueden validar datos de dos maneras:
    - Dinámicamente: por ejemplo con onChange.
    - Al finalizar la entrada de datos mediante un onSubmit.
- Para evitar el comportamiento por defecto de botones en un formulario (y en general de cualquier elemento, como enlaces), se puede devolver en la función callback del evento un false o prevenir dicho el comportamiento:

```
elem.onclick=function(){
                    ….
                    return false;
}
```

```
elem.onclick=function(){
                    ….
                    event.preventDefault();
}
```

Las API de Validación

- Un nuevo conjunto de mecanismos para validación basados en API
- El objeto fundamental para esa estructura es el **ValidityState**
- Cada elemento de un formulario puede llamar al método **checkValidity**, y analizar su contenido con la idea de mostrar información pertinente de validación al usuario
- Este objeto se expone a través del atributo **Validity** que contiene cada formulario
- Puede adoptar una lista de valores dependiendo del tipo de error de validación que se haya producido

Las API de Validación

- Los posibles valores son:
  - valueMissing (campo requerido)
  - typeMismatch (error de tipo)
  - patternMismatch (patrón incorrecto)
  - tooLong (demasiado largo)
  - rangeUnderflow (rango por debajo de lo esperado)
  - rangeOverflow (rango excedido)
  - stepMismatch (paso intermedio incorrecto)
  - customError (error personalizado)
- Estos valores pueden ser consultados con posterioridad utilizando funciones JavaScript

Las API de Validación

```
<label>Tarjeta de Crédito<input name="f" id="f" type="text" /></label>
<script>
document.getElementById('f').oninput=function check() {
if (this.value == "4B" ||
    this.value == "A.Express" ||
    this.value == "Visa") {
    this.setCustomValidity("" + this.value +"' no es un tipo de tarjeta válido.');
} else {
    // Si la entrada es correcta borramos el mensaje
    this.setCustomValidity('');
}
}
</script>
```

Pongámoslo en práctica

Crea un formulario con los siguientes campos:

- Nombre (solo texto y min size 3)
- Apellido (solo texto y min size 3)
- Edad (número menor de 100)
- Email (Formato email /\S+@\S+\.\S+/)

Añade al formulario un botón para enviar los datos. Al hacer clic en el botón se debe verificar el tipo de datos que se han introducido.
Si alguno no es válido, se debe mostrar un mensaje de error en un div. Si todo es correcto, enviar el formulario.

Pongámoslo en práctica

Añadir un campo al formulario: edad

- Solo debe permitir números.
- *Debe ser menor que 100