Fundamentos de Programación

JavaScript

DOM

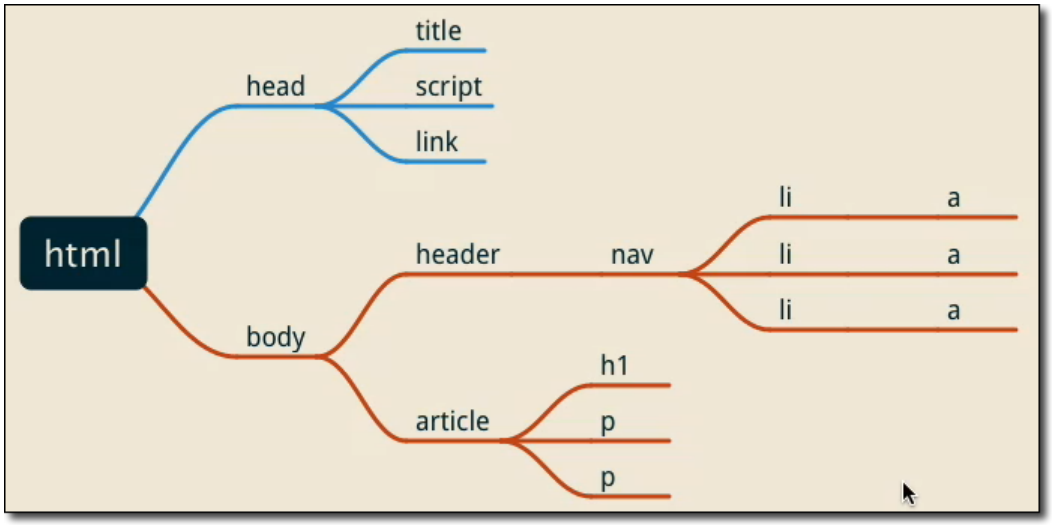
IES La Nía

DOM

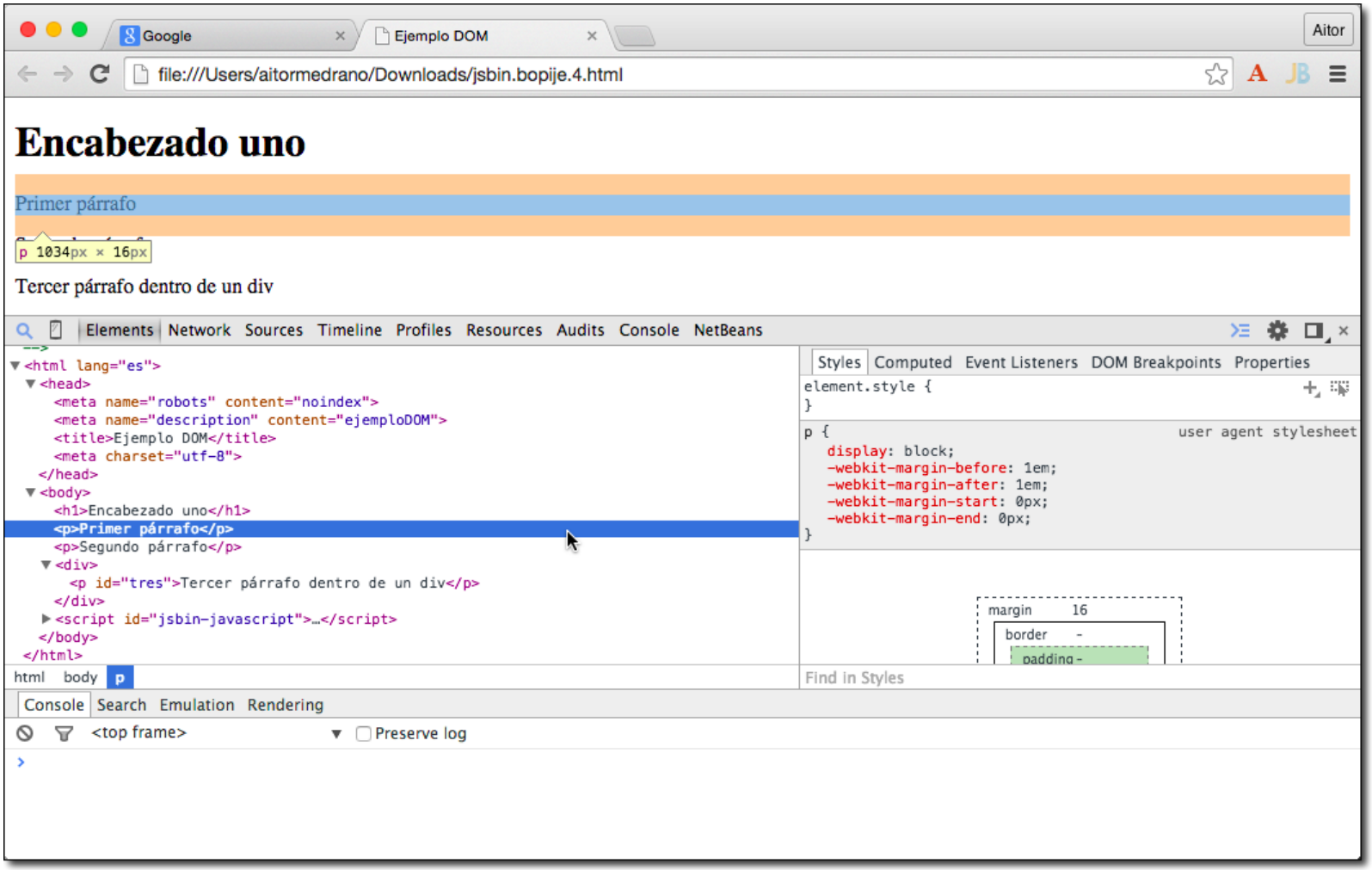
- *Document Object Model*
- Describe las relaciones entre los elementos HTML
- El navegador interpreta y organiza el código HTML como un DOM
- API para *CSS* y *JavaScript*

Nodo

- Cada uno de los elementos del DOM.
- Un nodo tiene un padre (*parent*) y puede tener varios hijos (*children*)
- Los nodos de un mismo nivel son hermanos (*siblings*), y comparten padre.



Dev-Tools



http://jsbin.com/bopije/4/edit?html

Uso de la Consola

- Podemos trabajar con el DOM desde la Consola (ESC) de las *Dev-Tools*
- Mediante el nombre de los nodos
  - `document`
  - `document.body`
  - `document.head`
- O mediante selectores CSS
  - `document.getElementById('tres');`
  - `document.querySelector('#tres');`
  - `$('#tres');`

Seleccionando Elementos I

- `document.getElementById(`*`id`*`)`
  - Devuelve un nodo a partir del elemento con atributo `id`
  - Sólo espera uno por página

- `getElementsByTagName(`*`etiqueta`*`)`
  - Obtiene un array con los elementos de la etiqueta
  - Se puede aplicar sobre `document`, o a partir de un nodo
    - `document.getElementsByTagName('li');`
    - `var menu = document.getElementById('menu');`
    - `var elems = menu.getElemensByTagName('li');`

Ejercicios

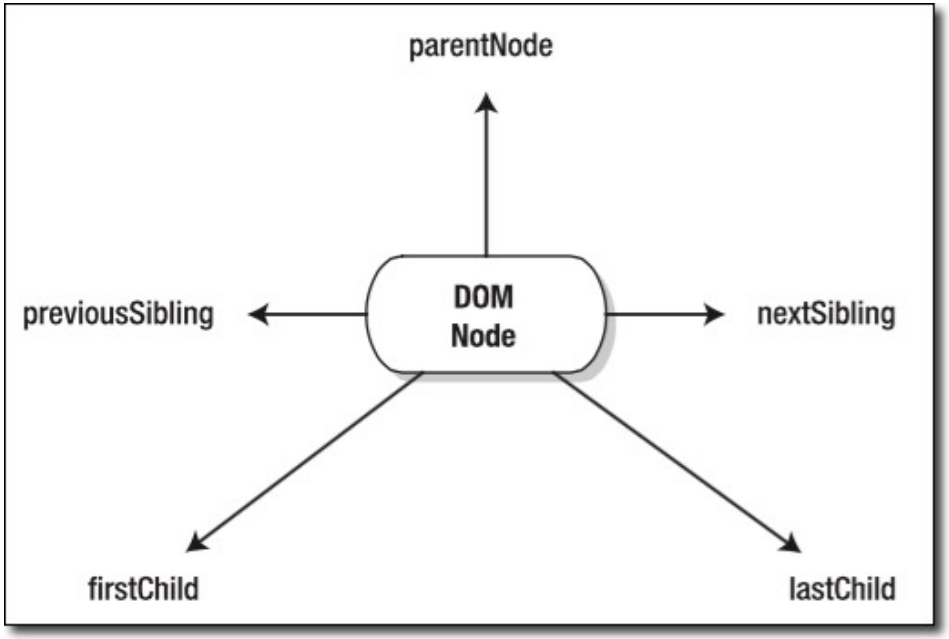
- A partir del contenido de `DOM01`
- En el archivo `_/js/myscript.js`
- (`ej601Seleccion`) Asignar cada selección a una variable :
  - El elemento cuyo `id` es `main`
  - Todos los enlaces
  - Los enlaces dentro de `main`

Propiedades de un Nodo

- `nodeType` → valor numérico con el tipo
  - 1 elem, 2 atributo, 3 texto, etc...
- `nodeName` → nombre del nodo
- `atributes` → array de atributos del nodo
- `nodeValue` → elemento dentro del nodo

Relaciones entre Nodos

- A partir de un nodo, mediante notación punto:

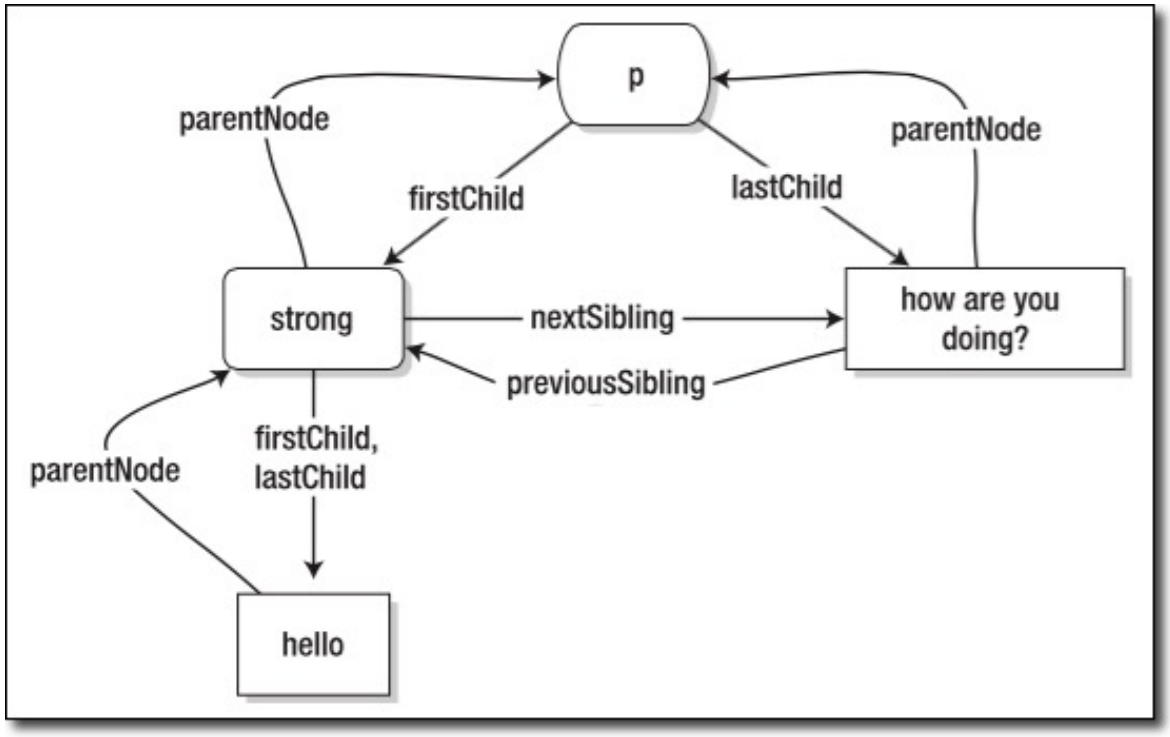


- `parentNode`: padre
- `previousSibling`: hermano anterior
- `nextSibling`: siguiente hermano
- `firstChild`: primer hijo
- `lastChild`: último hijo
- `childNodes`: array con los hijos

```
var primerHijo = document.body.firstChild;
var primerHijo2 = document.body.childNodes[0];
```

Representación DOM

```
<p><strong>Hello</strong> how are you doing?</p>
```



Relaciones de Nodo – Sólo Elementos

- Sólo considera los nodos que son elementos `(nodeType = 1)`
- `previousElementSibling`: hermano anterior
- `nextElementSibling`: siguiente hermano
- `firstElementChild`: primer hijo que es un elemento
- `lastElementChild`: último hijo que es un elemento
- `children`: array con los hijos elementos

Ejercicios

- A partir del contenido de `DOM01`
  - En el archivo `_/js/myscript.js`
- (`ej602SeleccionRelaciones`) Mediante las relaciones entre los nodos:
  - Selecciona el primer párrafo del artículo.
  - Selecciona el segundo elemento del menú (*artistas*).
  - Selecciona el texto después del enlace del artículo (*in the heart of Seattle*).

Seleccionando Elementos II

- `getElementsByClassName(claseCSS)`
  - Devuelve un *array* con los elementos cuya clase CSS coincida
  - HTML 5 → no funciona en navegadores antiguos

```
var agua = document.getElementsByClassName('agua');
var tablero = document.getElementById('tablero');
var agua2 = tablero.getElementsByClassName('agua');
```

Seleccionando Elementos III

- `querySelector(selectorCSS)`, `querySelectorAll(selectorCSS)`
  - Recupera nodos mediante un selector CSS
  - Devuelven un nodo o un array de nodos (`All`)
  - Similar a *jQuery*

```
var tablero = document.querySelector('#tablero');
var agua = tablero.querySelectorAll('.agua');
var agua2=document.querySelectorAll('#tablero.agua');
```

Selectores CSS Básicos

Selector	Propósito	Ejemplo
etiqueta	Encuentra todos los elementos de etiqueta	p
#identificador	Encuentra el elemento cuyo id es identificador	#contenido
.nombreClase	Encuentra todos los elementos cuyo class es nombreClase	.anuncio
etiqueta.nombreClase	Encuentra todos los elementos de tipo etiqueta cuyo class es nombreClase	div.anuncio
etiqueta#ident.nombreClase	Encuentra el elemento de tipo etiqueta cuyo id es ident y su class es nombreClase	div#banner.anuncio
*	Encuentra todos los elementos de la página	*

Selectores CSS Jerárquicos

Selector	Propósito	Ejemplo
selector1, *selector2*, …	Encuentra todos los selectores especificados	`p, div`
.clase1.clase2	Encuentra todos los elementos cuya `class` son *clase1* y *clase2*	`.anuncio.vip`
padre > hijo	Encuentra todos los elementos *hijo* que son hijos directos del tipo *padre*	`div > img`
ascendiente descendiente	Encuentra todos los elementos *descendiente* contenidos dentro del tipo *ascendiente*	`div img`
anterior + posterior	Encuentra todos los elementos *posterior* que están después de *anterior*	`img + p`
anterior ~ hermanos	Encuentra todos los hermanos (*siblings*) que están tras *anterior* y que cumplen el selector de *hermanos*	`div.carrusel ~ img`

Ejercicios

- A partir del contenido de `DOM01`
  - En el archivo `_/js/myscript.js`
- (`ej603SeleccionSelectores`) Mediante los selectores CSS:
  - Vuelve a realizar las selecciones de los dos ejercicios anterior pero haciendo uso de `querySelector`

Atributos

- Para acceder a los atributos de un elemento → notación punto
- Propiedades de lectura y escritura
  - `img.src` o `img.src=''`
- Podemos añadir atributos en caliente
  - `img.alt = 'foto bonita'`
- Para añadir una clase CSS, atributo `className`

Atributos mediante métodos

- `nodo.getAttribute(nombreAtrib)`
  - Obtiene el valor del atributo
- `nodo.setAttribute(nombreAtrib, valorAtrib)`
  - Asigna el valor al atributo
- `nodo.hasAttribute(nombreAtrib)`
  - Indica si el nodo contiene el atributo
- `nodo.removeAttribute(nombreAtrib)`
  - Elimina el atributo

Estilos CSS

- `nodo.`**`style`**`.propiedad`
  - `miNodo.style.color = 'red';`
- `nodo.`**`className`**
  - `miNodo.className = 'resaltado';`
- Operaciones sobre listas de clases:
  - `nodo.classList.add(), nodo.classList.remove(), .toggle()...`

Filtros basado en los atributos

Filtro	Propósito
[*atrib*]	Incluye los elementos que tienen el atributo *atrib*
[*atrib*=*valor*]	Incluye los elementos que tienen el atributo *atrib* con el valor *valor*
[*atrib*!=*valor*]	Incluye los elementos que tienen el atributo *atrib* y no tiene el valor *valor*
[*atrib*^=*valor*]	Incluye los elementos que tienen el atributo *atrib* y su valor comienza por *valor*
[*atrib*$=*valor*]	Incluye los elementos que tienen el atributo *atrib* y su valor termina con *valor*
[*atrib**=*valor*]	Incluye los elementos que tienen el atributo *atrib* y su valor contiene *valor*
[*filtroAtrib1*][*filtroAtrib2*]	Incluye los elementos que cumplen todos los filtros especificados, es decir, *filtroAtrib1* y *filtroAtrib2*

Ejercicios

- A partir del contenido de `DOM02`
  - En el archivo `_/js/myscript.js`
- (`ej604SelectorAtributos`) Mediante los selectores CSS y el uso de atributos:
  - Intercambia las caras del primer y segundo artista

Modificando Contenido

- `nodo.`**`innerHTML`** // `nodo.`**`innerText`**
  - Contenido interno
- `nodo.`**`outerHTML`** // `nodo.`**`innerText`**
  - Incluye la etiqueta del nodo
- `nodo.`**`insertAdjacentHTML`**`(puntoInsercion, textoHtml)`
  - PuntoInsercion → `beforebegin, afterbegin, beforend, afterend`

Ejercicios

- A partir del contenido de `DOM02`
  - En el archivo `_/js/myscript.js`
- (`ej605ModificandoContenido`) Haciendo uso de `querySelector`, la propiedad `style` e `innerHTML`:
  - Traduce los títulos de los artículos (los que estan en rojo)
  - Cambia el color de los titulos a naranja

Creando Contenido I

- `document.createElement(elemento)`
  - Crea un nuevo elemento que se tiene que añadir al DOM
- `nodo.appendChild(elemento)`
  - Añade un elemento dentro del nodo

```
var imagen = document.createElement('img');
imagen.src = 'imagenes/logo.jpg';
var miNodo = document.querySelector('#destino');
miNodo.appendChild(imagen);
```

Creando Contenido II

- `document.createTextNode(texto);`
  - Crea un elemento de tipo texto con el texto indicado
- `nodoPadre.insertBefore(nuevoNodo, nodoReferencia)`
  - Inserta un nodo en el lugar indicado

```
var nodoTexto = document.createTextNode('Hola Aspe');
miNodo.insertBefore(nodoTexto, miNodo.childNodes[5]);
```

Clonado y Borrado

- `var copia = nodo.`**`cloneNode`**`(bool)`
  - Copia el contenido del nodo en un nodo nuevo
  - El *booleano* indica si tiene que clonar los hijos
- `nodoPadre.`**`removeChild`**`(nodoHijo)`
  - A partir del nodo padre, permite eliminar un nodo hijo

Sustituyendo nodos

- `nodoPadre.`**`replaceChild`**`(nuevoNodo, viejoNodo)`
  - Reemplaza un nodo por otro.
  - Ahorra tener que mover y borrar.

```
viejoNodo.parentNode.replaceChild(nuevoNodo, viejoNodo)
```

Ejercicios

- A partir del contenido de `DOM02`
  - En el archivo `_/js/myscript.js`
- (`ej606BorrandoContenido`) Mediante las operaciones de manipulación de nodos:
  - Borra la última fila de fotos