

TEMA 4

ECMAScript, DOM, NAVEGADOR

EI1042 - TECNOLOGÍAS Y APLICACIONES WEB

EI1036- Tecnologías Web para Sistemas de Información (2020/2021)

PROFESORADO: DRA. DOLORES MARÍA LLIDÓ ESCRIVÁ



Universitat Jaume I.

TABLA DE CONTENIDOS

1. Introducción EMACScript:
2. JavaScript - Objetos - Variables - Tipos de datos - Funciones - Diferencia Node.js y Javascript Navegador
3. Document Object Model (DOM)
4. Componentes de los navegadores
5. CSSOM y RenderTree
6. JavaScript Navegador - HTML DOM - Window y Document - el objeto `this`
7. Gestión Eventos (Event Target)
8. Template
9. JSON
0. CORS

1. INTRODUCCIÓN EMACSCRIPT

- El lenguaje **JavaScript** se creó inicialmente para los navegadores web.
- En 1997 JavaScript adoptado como estándar de la European Computer Manufacturers Association ECMA.
- Evolucionó en un lenguaje con muchos usos y plataformas:

EMACScript.

- Una **plataforma** puede ser un cliente web (Chrome) o un servidor web (Node.js) u otro anfitrión.
- El **Script Engine** es el programa que ejecuta código en un dialecto del lenguaje de script (JavaScript) proporciona además del núcleo del lenguaje, los objetos y funciones específicas de la plataforma. Cada navegador tiene su propia implementación
(webkit->safari, v8->Chrome, Node.js).

DIALECTOS/API

- JavaScript, JScript y ActionScript son dialectos de ECMAScript.
- Los **dialectos** incluyen extensiones al lenguaje y **APIs**.
- Una aplicación escrita en un dialecto puede ser incompatible con otra, si no utilizan un subconjunto común de características **APIs compatibles**

2. JAVASCRIPT

- Lenguaje interpretado, dialecto de ECMAScript.
- La sintaxis es semejante al C pero las variables no están tipificadas y no distingue entre mayúsculas y minúsculas.
- No existe un cuerpo principal del programa (main), todo lo que no esté dentro de una función es ejecutado.
- En JavaScript todo son objetos.
- Declarar las variables y funciones antes de usarse.
- No es obligado usar ; al final de las sentencias
- El sangrado es para mejorar la lectura.
- Un grupo de sentencias multiples se agrupan con {}

HERRAMIENTAS EN JAVASCRIPT

- JavaScript: Integrado en navegador o Node.
- API del navegador/Servidor: construcciones integradas en el navegador/servidor que se asientan sobre el lenguaje JavaScript y le permiten implementar la funcionalidad más fácilmente.
- API de terceros: construcciones integradas en plataformas de terceros (por ejemplo, Twitter, Facebook) que le permiten usar algunas de las funciones de esas plataformas en sus propias páginas web (por ejemplo, mostrar sus últimos Tweets en su página web).
- JavaScript Library: por lo general, uno o más archivos de JavaScript que contienen funciones personalizadas que puede adjuntar a su página web para acelerar o permitir la escritura de funciones comunes. Los ejemplos incluyen jQuery, Mootools y React .
- Frameworks de JavaScript: (por ejemplo, Angular y Ember - Express, Koa.js..) tienden a ser paquetes de HTML, CSS, JavaScript y otras tecnologías que instala y luego se usa para escribir una aplicación completa desde cero.

OBJETOS

- Una lista es en realidad una clase de objeto.
- Un objeto de JavaScript es un array asociativo formado por sus propiedades y sus métodos.
- Objetos predefinidos en el núcleo de JavaScript:
 - Date: objeto genérico para representar fechas (por defecto es la fecha actual). `let fin = Date.now();`
 - Math: objeto que incorpora las funciones matemáticas y constantes. `var pi = Math.pi;`
 - console: objeto para comunicarnos con la consola `console.log('Hola');`.

CREACIÓN DE OBJETOS

- Instancia directa formato breve.

```
personObj = {nombre: "Carlos Sempere", apodo: "Doe", edad: 50};
```

- Instancia directa con creando objeto "vacío".

```
var Autor = new Object();  
Autor.nombre = "Carlos Sempere";  
Autor.apodo = "Doe" ;  
Autor.edad = 50;  
//objeto 'Autor' con atributos
```

- Con un constructor.

```
function Persona(nombre, apodo, edad){  
    this.nombre = nombre;  
    this.apodo = apodo;  
    this.age = edad;  
}  
var Autor = new Persona("Anas Sempere", "Doe", 50);
```

DECLARACIÓN VARIABLES / ÁMBITOS

- `var`: permite definir una variable **local** en una función o **global** si está fuera de una función.
- `let`: declara una variable local al ámbito en el que esté definida.
- `const`: Declara una constante de solo lectura en un ámbito.

EJEMPLO JS

```
var z; // sin inicializarla (vale undefined)
var x = 42;
y = 42 // sin declarar variable
let y = 13;
const PI = 3.14;
let arr = [ 3, 5, 7 ];
arr.foo = "hola";
```

TIPOS DE DATOS

- Number: `let x = 5;`
- String: `let cad = '32';`
- Boolean: `let b1 = x > 0;`
- Date: `let fin = Date.now();`
- RegExp: `var re = /\w+/;`
- **null**: `foo = null` // definida no inicializada
- **undefined**: `typeof(y)` //no definida no inicializada

ARREGLOS

- Uso de un array:

```
//Directamente con elementos
let colors = ["red", "green", "blue"];
//Crear un array vacío y luego poner los elementos:
let colors = new Array();
colors[0] = "red"; colors[2] = "blue";
colors[1] = "green";
//Crear un array con sus elementos:
let colors = new Array(3,2,1,0);
```

- Recorrido arrays

```
const numeros = [1, 2, 3, 4, 5];
for (let i = 0; i < numeros.length; i++) {
  console.log(i, numeros[i]);
}
for (let i in numeros) {
  console.log(numeros[i]);
}
for (let i of numeros) {
  console.log(i);
}
```

FUNCIONES

```
function NombreFuncion (parametro1, ..., parámetro N )  
{...  
  return valor;  
}
```

- Algunas Funciones Predefinidas:
 - Tipo de una variable: *typeof(variable)*;
 - Evalúa un código JavaScript : *eval("Primera(p1, p2)")*

Las funciones son un objeto por ello podemos:

- asignar funciones a variables, y referenciarlas utilizando la variable
- pasar funciones como parámetros a otras funciones
- obtener funciones como resultados de la ejecución de la función.

```
function grado() {  
function titulo(name) {  
    return "Dr. " + name; }  
return titulo; //una funcion!  
}  
let phd = grado();  
phd("Turing"); //Dr Turing
```

EJEMPLOS: TRY-THROW-CATH

```
function getMonthName(mo) {  
    mo = mo - 1 ; // Adjust month number for array index (1 = Jan)  
    var months = [ "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",  
        "Aug", "Sep", "Oct", "Nov", "Dec" ];  
    if (months[mo]) { return months[mo]; }  
    else { throw "InvalidMonth"; // Lanzamos una excepción. }  
}  
myMonth=5;  
try { // statements to try  
    monthName = getMonthName(myMonth); // function could throw e  
} catch (e) {  
    monthName = "unknown";  
    console.log(e); // pass exception object to error handler  
}
```


EJEMPLOS

```
function Primera(p1, p2) {  
  p1(p2);  
}  
// Función Literal  
var Segunda = function (m1) {  
  console.log(m1 / 10 );  
}  
// Pasando la Función Segunda como parámetro de Primera  
Primera(Segunda, 20 );
```

¿Que devuelve la llamada `Primera(Segunda, 20)`?

FUNCIÓN ANÓNIMA

```
function (parametro1, ..., parametro N )  
{  
  ...  
return valor;  
}
```

¿Cómo la ejecutamos si no la podemos llamar?

- Poniendo la definición de la función anónima entre paréntesis (), y pasándole parámetros

```
/* funcion anónima sin parámetros */  
(function() { alert("hola mundo") })()  
/* funcion anonima como un parametro */  
(function(quien) {alert("hola "+quien)})( "mundo" )
```

- O bien con return

```
return function(quien) {alert("hola " + quien)} ( 'mundo' )
```

EJEMPLOS JAVASCRIPT

```
const f1 = function(x,y)
{
    var s=x+y;
    return s;
}
console.log(f1( 4 , 6 ));
let f2 = f1;
console.log(f2( 3 , 3 ));
```

¿Que aparece en la pantalla al ejecutar el código anterior?

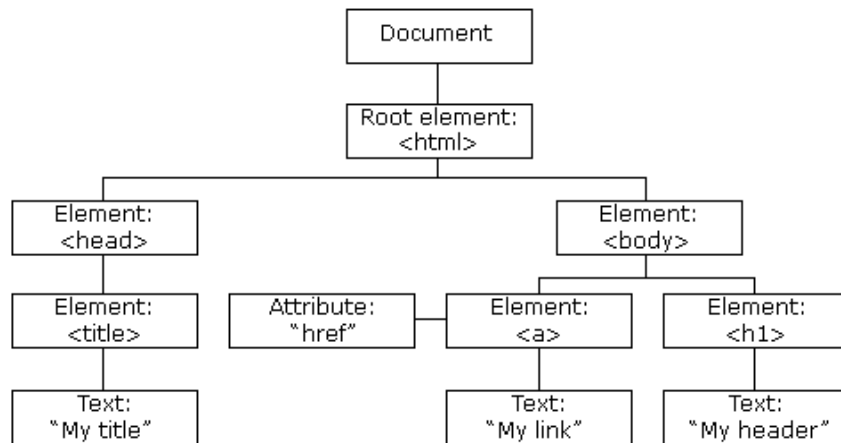
EJEMPLOS: RECORRIDO ARRAYS

```
const numeros = [1, 2, 3, 4, 5];
numeros[5]=6
//Usando un callback
numeros.forEach(function(numero) {
    console.log(numero);
});
numeros.push(7);
//funcion flecha
numeros.forEach(element => console.log(element));
numeros.append(8);
numeros.forEach((numero, index) => {
    console.log('Indice: ' + index + ' Valor: ' + numero);
});
numeros.pop();
//mapeo se genera un nuevo objeto iterador. Pero este no ocupa
nums=numeros.map(function (num) { return (num) })
console.log(nums)
```

3. DOM (DOCUMENT OBJECT MODEL)

- Interfaz de programación para documentos HTML, SVG o XML.
- Conecta documentos (HTML, SVG o XML) con lenguajes de programación, modelado los documentos como objetos.
- Interfaz de lenguaje neutral independiente de plataforma que permite a los programas y scripts el acceso dinámico y la actualización del contenido, la estructura o estilo de documentos.
- Proporciona una representación estructurada del documento como una estructura lógica que es muy parecida a un árbol. Cada elemento es un nodo (document, head, frame, body, p, div) los cuales contienen propiedades, métodos y eventos asociados a los mismos.

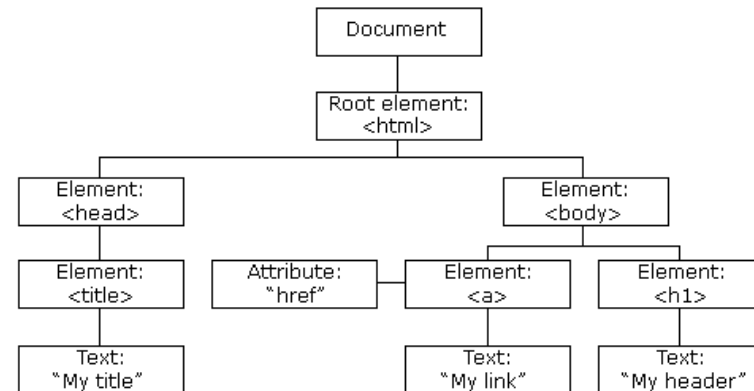
- <https://dom.spec.whatwg.org/>
- <https://www.w3.org/DOM/DOM>



ÁRBOL DEL DOM: CORE DOM

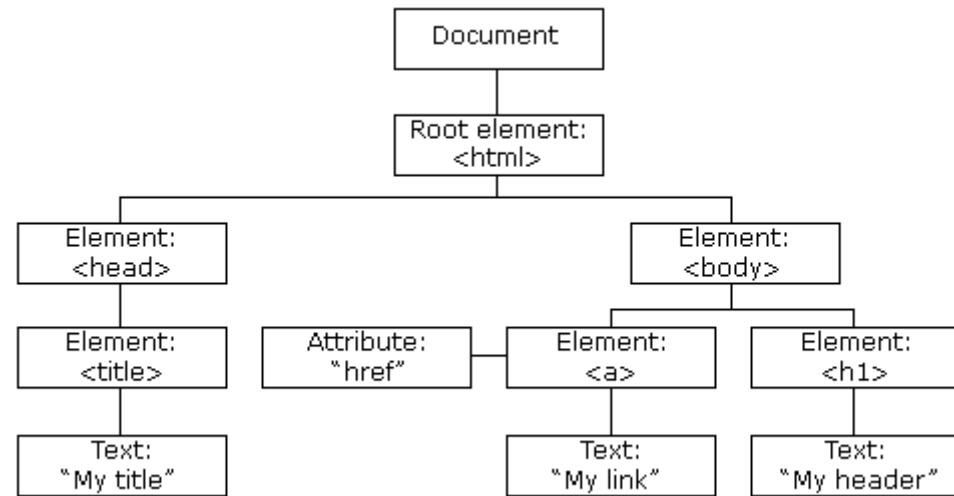
- **root node:** Nodo raíz (no tiene padre).
- **father node:** Todos los nodos (excepto raíz) tienen padre.
- **child node:** Un nodo puede tener varios hijos.
- **leaf node:** Un nodo hoja es un nodo sin hijos.
- **sibling nodes:** Los nodos hermanos comparten el mismo padre.

Ejemplos de métodos:



- `x.appendChild(y)`: añade un hijo y al nodo x
- `x.removeChild(y)`: elimina el hijo y del nodo x
- `x.replaceChild(y, z)`: cambia el hijo y por z en el nodo x

TIPOS DE NODOS



- DOCUMENT_NODE: Representa el documento entero.
- ELEMENT_NODE: Las etiquetas son nodos tipo elemento.
- ATTR_NODE: Los atributos de las etiquetas. Se asocia a un nodo elemento.
- TEXT_NODE: Contiene el texto entre las etiquetas de un elemento.
- COMMENT_NODE: Representa un comentario.
- DOCUMENT_TYPE_NODE: Contiene el docType del documento.
- DOCUMENT_FRAGMENT_NODE: Representa un fragmento de documento que debe estar bien formado ("template").

ATRIBUTOS DE LOS NODOS

- `x.nodeName` – nombre del nodo `x`
- `x.nodeValue` – valor del nodo `x`
- `x.nodeType` – tipo del nodo `x`
- `x.parentNode` – nodo padre
- `x.firstChild` - primer hijo
- `x.lastChild` - último hijo
- `x.previousSibling` - anterior nodo hermano
- `x.nextSibling` - siguiente nodo hermano
- `x.childNodes` – lista de los nodos hijos.
- `x.textContent` – contenido textual del nodo.

DOCUMENT/ELEMENT NODE

- `X.createAttribute(att_name)`: crea un nodo de tipo atributo con nombre *att_name*
- `X.createElement(tag_name)`: crea un elemento con nombre *tag_name*
- `document.createTextNode(text_as_string)`: crea un nodo de tipo texto
- `X.getElementsByTagName(tag_name)`
- `X.getElementsByClassName(class_name)` List of elements
- `X.getElementById(element_id)`
- `x.getAttribute(att)`: valor del atributo *att*
- `x.setAttributeNode(att_node)`, asigna el atributo correspondiente a *att_node*
- `x.setAttribute(att, value)`, asigna *value* al atributo *att*.

PHP DOM

Biblioteca libxml (Parser DOM):

- Core DOM
- No incluye la etiqueta Doctype.
- Compatible con DOM level 3 core.

EJEMPLO DE DOM EN PHP

```
<?php
$doc = new DOMDocument();
$root = $doc->createElement('html');
$doc->appendChild($root);
$head = $doc->createElement('head');
$root->appendChild($head);

$title = $doc->createElement('title');
$title->appendChild($doc->createTextNode('Este es el título'));
$head->appendChild($title);
$body = $doc->createElement('body');
$root->appendChild($body);
$h1 = $doc->createElement('h1');
$root->appendChild($h1);
$h1->appendChild($doc->createTextNode('Esto es el cuerpo'));
$doctype = "<!DOCTYPE html >";
echo $doctype.$doc->saveHTML();
?>
```

<http://piruletas0.scienceontheweb.net/PHP/T4/HTMLDOM.php>

EJEMPLO DE DOM CON PHP (LOAD FROM STRING)

```
<?php
$html='<html><head> <meta
charset="utf-8"><title>PHP WEB</title></head><body>
<div><h1>Web page parsing</h1>
<p>This is an example Webpage.</p></div></body></html>';
$doc = new DOMDocument();
$doc->loadHTML ($html);
$h2 = $doc->createElement('h2');
$h1= $doc->getElementsByName("h1")[ 0 ];
$h1->parentNode->appendChild($h2);
$h2->appendChild($doc->createTextNode('Esto es el H2'));

$doctype="<!DOCTYPE html >";
echo $doctype.$doc->saveHTML();
?>
```

http://piruletas0.scienceontheweb.net/PHP/T4/HTMLDOM_Load.php

DOM JS NAVEGADOR

Ejemplo:

```
var father, child;
father = document.querySelector('head');
child = document.createElement('title');
child.textContent = 'HTMLDOM';
father.appendChild(child);
father = document.querySelector('body');
child = document.createElement('h1');
child.textContent = 'Bienvenido <span > Pepe </span>';
child.innerHTML = 'Adios <span > Pepe </span>';
child = document.createElement('h2');
child.innerHTML = 'Adios <span > Pepe </span>';
father.appendChild(child);
```

¿Qué hace este ejemplo? Comprueba el funcionamiento en :

<http://piruletas0.scienceontheweb.net/PHP/T4/HTMLDOM.html> ¿Por qué se visualizan las etiquetas HTML?

¿Diferencia innerHTML, textContent?

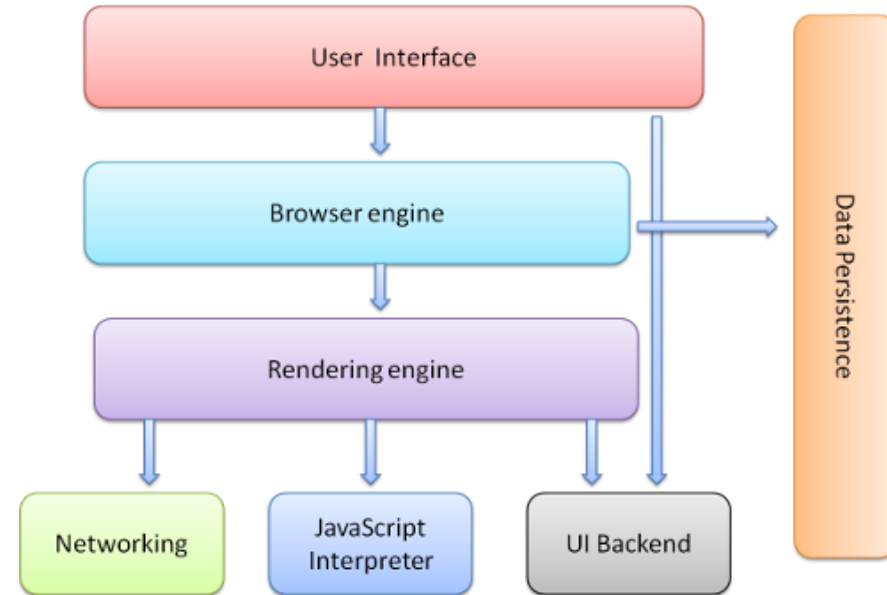
CUESTIÓN:

¿Qué error hay en el código de esta página web?:

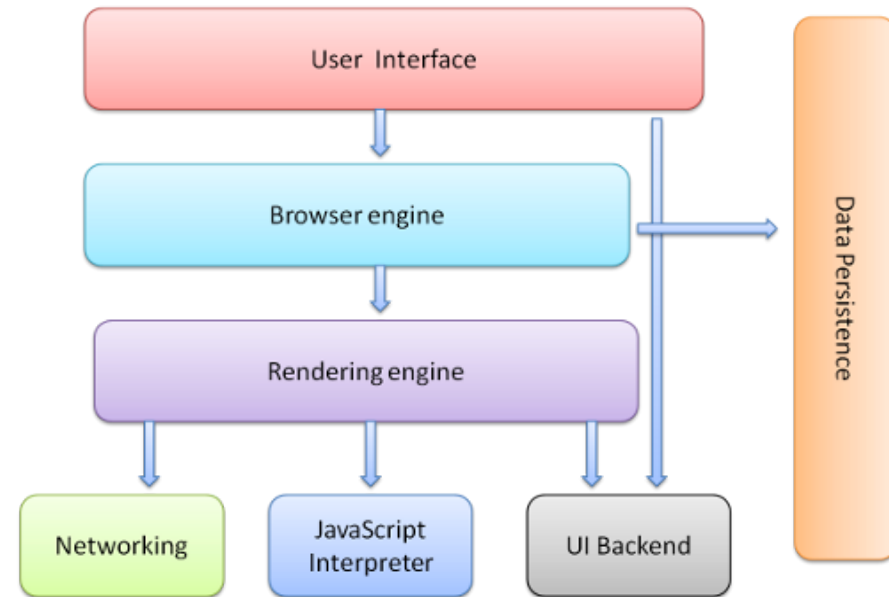
<http://piruletas0.scienceontheweb.net/PHP/T4/HTMLDOMError.html>

EJERCICIO: CREAR UNA TABLA UTILIZANDO EL DOM

4. COMPONENTES NAVEGADOR WEB



- *Interfaz de usuario* : incluye la barra de direcciones, el botón de avance/retroceso...Todo excepto la ventana principal donde se muestra la página web.
- *Motor del Navegador* : coordina las acciones entre la interfaz y el motor de renderización. Carga URL, mensajes error, botón atrás.
- *Motor de renderización* : responsable de mostrar el contenido.
- *Red* : responsable de las llamadas de red, como las solicitudes HTTP (Caché de documentos).




- *Intérprete de JavaScript* : permite analizar y ejecutar el código Javascript. El resultado se pasa al motor de renderización.
- *Almacenamiento de datos* : capa de persistencia. Gestiona los datos de usuario, tales como cookies, indexDB, WebStorage.
- *Backend de interfaz de usuario*: proporciona primitivas de dibujo, widgets de la interfaz de usuario, fuentes, etc. (utiliza métodos sistema operativo)

<https://www.html5rocks.com/es/tutorials/internals/howbrowserswork/>

MOTOR DE RENDERIZADO

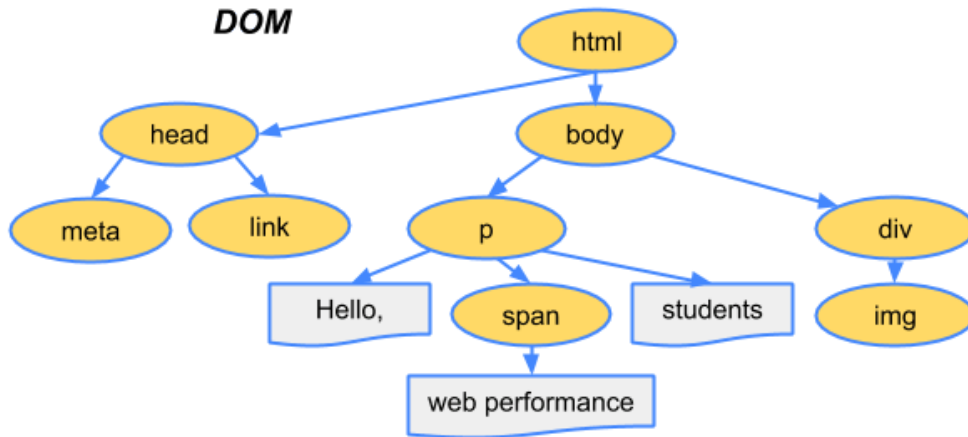
- Un motor de renderizado (web browser engine) es el software que toma contenido marcado (como HTML, XML, archivos de imágenes, etc.) e información de formateo (como CSS, XSL, etc.) y luego muestra el contenido ya formateado en la pantalla.
 - Los motores de renderizado lo usan los navegadores web, clientes de correo electrónico, u otras aplicaciones que deban mostrar contenidos web.
 - Cada motor de renderizado del navegador suele tener su propio intérprete Javascript (**script engine**).

COMPONENTES NAVEGADORES

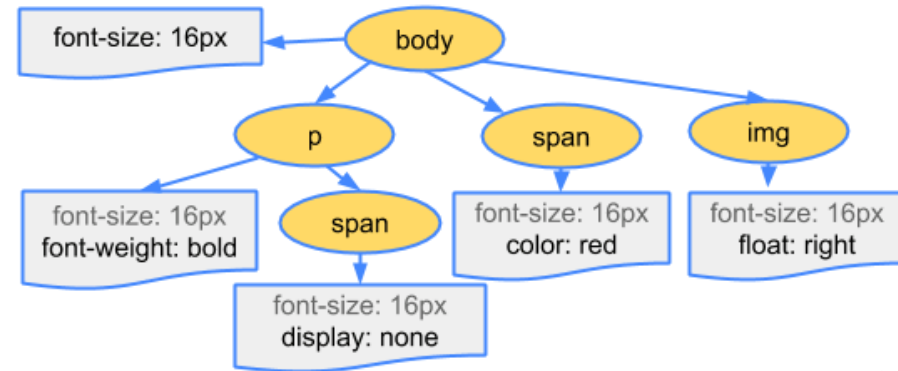
<input type="checkbox"/>	 A Browser ▾	A Rendering / Layout Eng... ▾	A Scripting Engine ▾
1	Chrome	Blink (C++)	V8 (C++)
2	Mozilla Firefox	Gecko (C++)	SpiderMonkey (C/C++)
3	IE Edge	EdgeHTML (C++)	Chakra JavaScript engine (...)
4	Opera	Blink (C++)	V8 (C++)
5	Internet Explo...	Trident (C++)	Chakra JScript engine (C++)
6	Apple Safari	WebKit (C++)	JavaScript Core (Nitro)

5. RENDER TREE: DOM+CSSOM

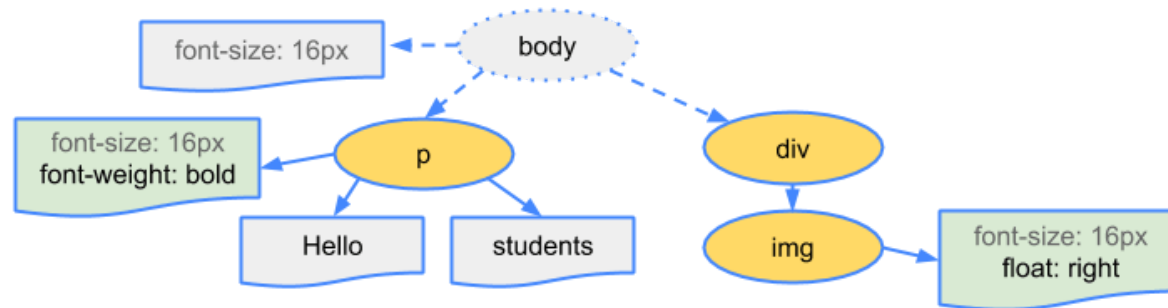
DOM



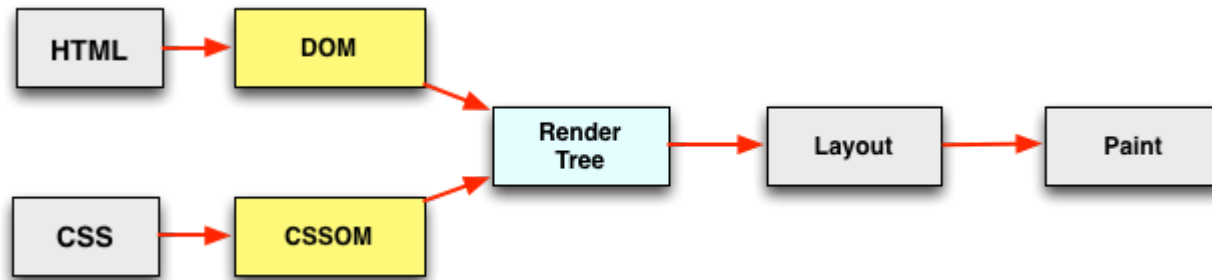
CSSOM



Render Tree



FLUJO BÁSICO DEL MOTOR DE RENDERIZACIÓN



1. Los árboles DOM y CSSOM se combinan para formar el árbol de representación.
2. Render tree: El árbol de representación solo contiene los nodos necesarios para representar la página.
3. Layout: El diseño calcula la posición y el tamaño exactos de cada objeto.
4. Paint: El último paso es la pintura, que recibe el árbol de representación final y representa los píxeles en la pantalla.

<https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction?hl=es-419>

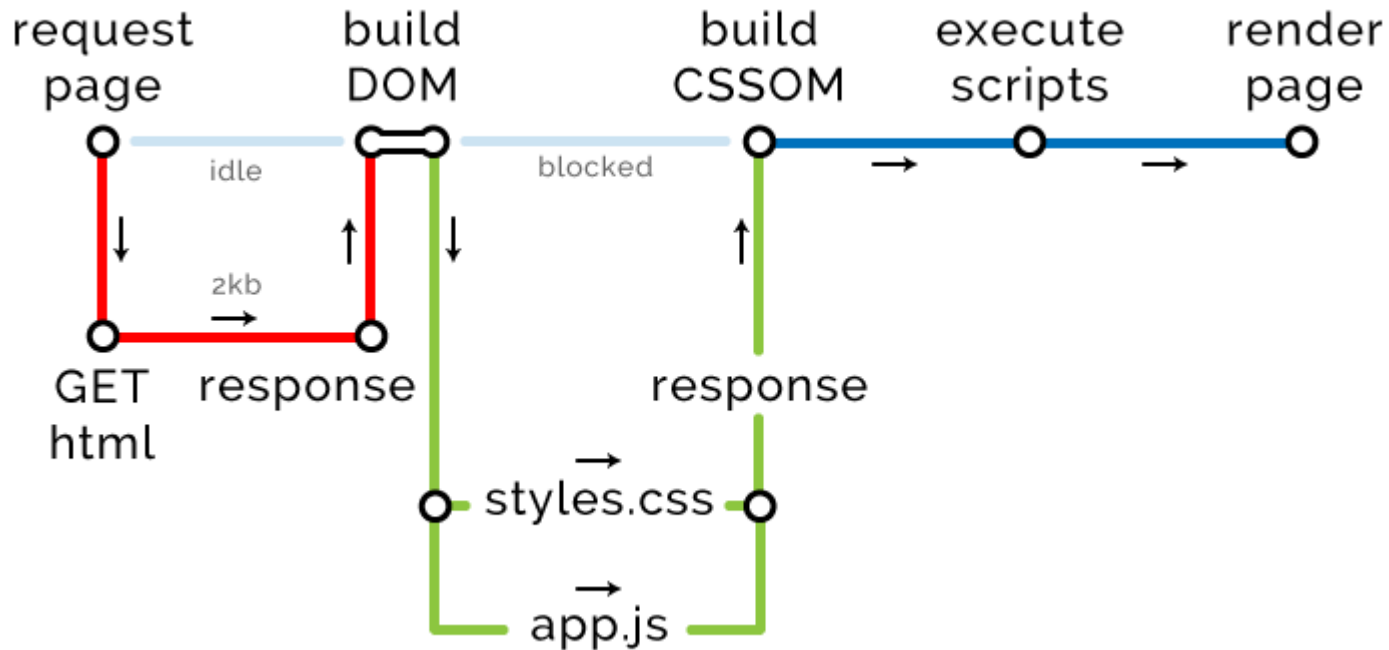
CSS RENDER

- En <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference> tenemos una guía de CSS estándar.

```
div.error-label{  
  color: #fff;  
  background-color: #DA362A;  
  -webkit-border-radius: 5px;  
  -moz-border-radius: 5px;  
  border-radius: 5px;  
}
```

- Pero los motores de renderizado tienen ciertas extensiones de CSS que no son todavía estándares:
 - [Mozilla CSS extensions](#) (prefixed -moz-)
 - [WebKit CSS extensions](#) (prefixed -webkit-)
 - [Microsoft CSS extensions](#) (prefixed -ms-)

CRP: CRITICAL-RENDERING-PATH (RUTA DE REPRESENTACIÓN CRÍTICA)



<https://hackernoon.com/optimising-the-front-end-for-the-browser-f2f51a29c572>

Podemos ver el CRP con la línea de tiempo de las herramientas de desarrolladores.

DIFERENCIAS JAVASCRIPT/NODEJS

- JavaScript se ejecuta en el navegador, mientras que el uso de Node.js nos permite ejecutar JavaScript fuera del navegador (Script Engine V8).
- JavaScript puede manipular DOM o agregar HTML, mientras que Node.js no tiene la capacidad de agregar HTML.
- JavaScript se usa principalmente para crear aplicaciones web front-end mientras que Node.js se usa en el desarrollo de back-end que es el desarrollo del lado del servidor.

La salida de JavaScript en el navegador puede ser:

- HTML element: `element.innerHTML()`.
- HTML output: `document.write()`.
- En un cuadro de alerta: `window.alert()`.
- En la consola del navegador: `console.log()`.

No tiene acceso a los dispositivos estándar de salida a excepción de :
`window.print()` que permite imprimir la página en la impresora.

CUESTIONES:

¿JavaScript tiene acceso a ficheros del cliente? ¿ y del servidor?

¿Node Js tiene acceso a ficheros del cliente?¿ y del servidor?

6. JAVASCRIPT EN EL NAVEGADOR

El código JS, se puede poner:

- Como valor de un atributo que es un evento. (No usar)

```
<span onclick="return handleClick(event, this);"> ¡Pulsa aquí
```

- Dentro de la etiqueta *script* tanto en la cabecera como en el body.

```
<script> console.log('hola'); </script>
```

- Importando un fichero externo

```
<script src="mi_url" defer></script>  
<script src="mi_url2" defer></script>
```

La etiqueta script tiene 2 atributos cuando se importa un el código de un fichero externo:

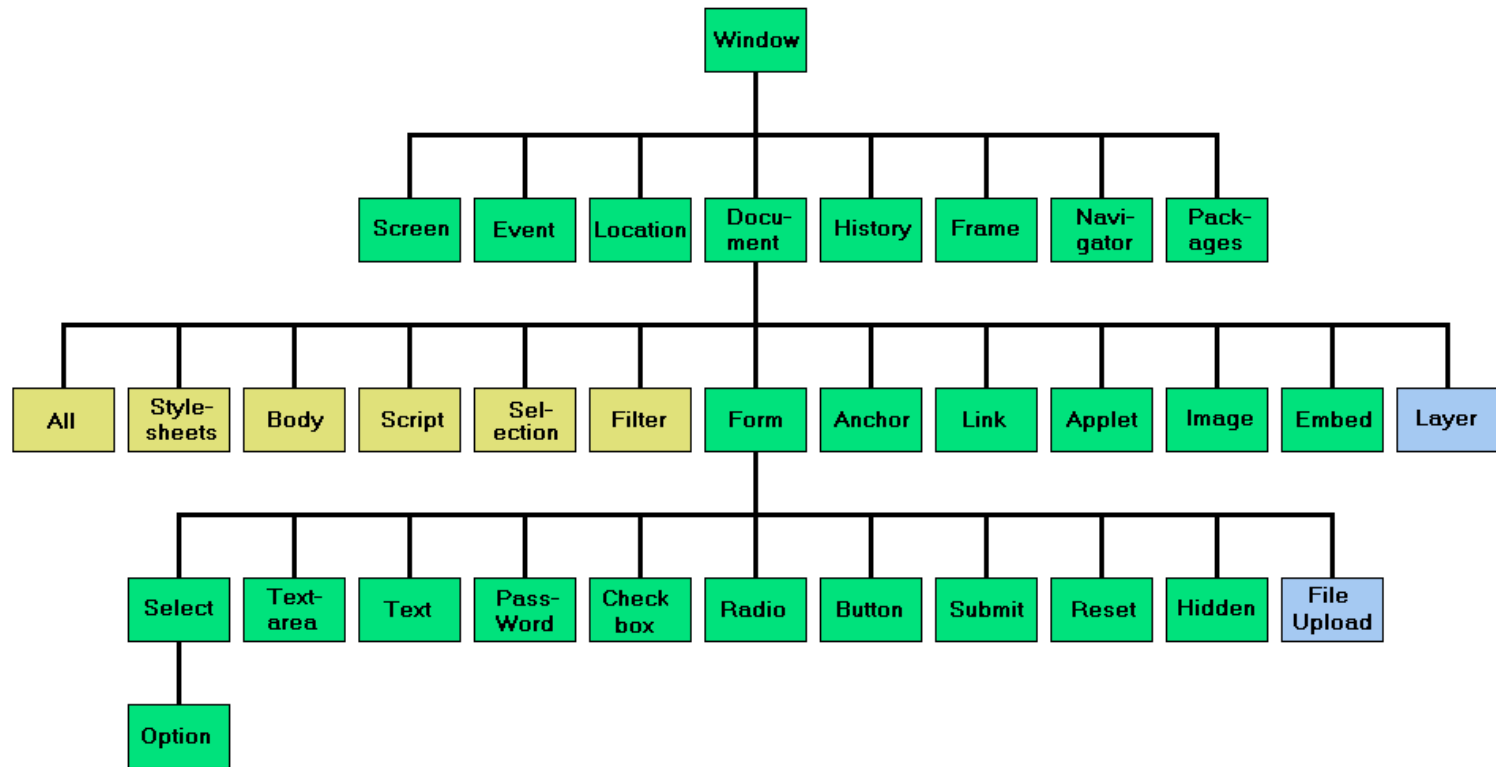
- **async**: Indica que los recursos se cargan asíncronamente y se ejecutan cuando están preparados
- **defer**: Los recursos se cargan asíncronamente y se ejecutan cuando el DOM se ha cargado ejecutándose los scripts por orden.

Webs APIs en el Navegador

- Canvas API: Proporciona un medio para dibujar gráficos a través de JavaScript y el elemento HTML `<canvas>`. Animación, gráficos de juegos, visualización de datos, manipulación de fotografías y procesamiento de video en tiempo real.
- Console API: permite a los desarrolladores realizar tareas de depuración, como registrar mensajes o los valores de las variables en puntos establecidos en su código. `console.log(myString);`
- HTML DOM API: Es un DOM Level 3 extendido con las especificaciones de HTML. Por ejemplo :el elemento `<canvas>` se representa por un `HTMLCanvasElement`.
- HTMLVideoElement Interfaz: Permite manipular objetos de videos. También, este interfaz hereda propiedades y métodos de `HTMLMediaElement` y `HTMLElement`.
- SVGElement: Interfaces de los elementos del SVG DOM.
- CSSOM (El Modelo de objetos CSS) conjunto de APIs que permite manipular CSS desde JavaScript.

7.API HTML DOM

La API HTML DOM: Definen la funcionalidad de cada uno de los elementos del HTML. Estas interfaces ofrecen acceso a la ventana del navegador y al documento que contiene el HTML, así como al estado del navegador, los complementos disponibles.



http://www.cs.ucc.ie/~gavin/javascript/05_img01.gif

INTERFAZ WINDOW

La interfaz Window representa una ventana que contiene un documento DOM HTML.

- El objeto window es el de mayor nivel en la jerarquía de objetos de JavaScript en el navegador.
- Es un objeto global en JavaScript. Contiene todas las variables y funciones globales.
- Representa la "ventana del navegador" y proporciona los métodos para controlarlo.

Por tanto:

- *window* : OBJETO GLOBAL que contiene todas las variables.
- *window.document* o *document* : objeto que tiene el documento html cargado en el navegador, o sea el DOM del documento HTML.

DOCUMENT OBJECT

- Es el Nodo raíz del HTML DOM
- Todos los nodos heredan los atributos del objeto `eventTarget`
- Atributos:
 - `document.documentURI`: URI del documento
 - `document.contentType`: tipo de contenido (html, imagen, etc.)
 - `document.styleSheets`: node con las hojas de estilos
 - `document.images` : lista de imágenes
 - `document.anchors`: lista de hiperenlaces
 - `document.body`: nodo del elemento body

EJEMPLO CREAR NUEVA VENTANA EN NAVEGADOR CON UN DOCUMENTO

```
var ventana=window.open();
var father, child;
father = ventana.document.querySelector('head');
child = document.createElement('title');
child.textContent='HTMLDOM';
father.appendChild(child);
father = ventana.document.querySelector('body');
child = document.createElement('h1');
father.appendChild(child);
child.textContent='Bienvenido </br>';
child = document.createElement('h2');
child.innerHTML='Esto es el H2<br/>';
father.appendChild(child);
```


CUESTIONES

- ¿Cómo modifico un documento cargado en el navegador?
- ¿Por qué es diferente el manejo del DOM en PHP y HTML?
- Diferencia entre innerHTML, textContent e innerText.
- ¿Se carga primero el body o se ejecuta primero el script?

EJEMPLO INNERHTML VS TEXTCONTENT

```
child.textContent='Bienvenido </br>';  
'Bienvenido </br>'  
child.innerHTML  
'Bienvenido &lt;/br>';  
child.innerHTML='Esto es el H2<br/>';  
'Esto es el H2<br/>'  
child.textContent  
'Esto es el H2'
```

NODO ELEMENT Y DOCUMENT

- Propiedades de un elemento x:
 - `x.events`: lista de Eventos de x
 - `x.methods`: lista de métodos de x
 - `x.elements`: colección de nodos de tipo elementos de x
 - `x.name`: lista de hijos de x con el nombre 'name'
 - `x.innerHTML`: asigna o devuelve el contenido html del nodo elemento(Crítico).
 - `x.style`: permite dar estilo a las etiquetas.
- Eventos:
 - `x.blur()`: Quita el foco en de un elemento.
 - `x.click()`: Realiza un click en un elemento.
 - `x.focus()`: Pone el foco en un elemento.
 - `x.toString()`: Convierte el contenido en una cadena.

SELECCIÓN DE NODOS CON MEDIAQUERIES

Podemos utilizar la notación de los mediaqueries de las reglas CSS para seleccionar nodos document o element.

- `x.querySelector(selector)` // Recuperar el primer nodo que cumple la condición
- `x.querySelectorAll(selector)` // Recuperar todos los nodos que cumplen la condición

```
x.querySelector("div");  
x.querySelector("#form1");  
x.querySelectorAll("div");
```

EJEMPLO CARGA IMÁGENES

```
Url2="http://www.bigfoto.com/themes/nature/animals/rhino.jpg"  
document.querySelector("#animal").src=Url2;
```

THIS OBJECT

- Cuando estamos dentro del atributo de una etiqueta **html**, **this** se refiere a la **etiqueta** que contiene el atributo.
- Cuando estamos en un Objeto, **this.xxx** se llama dentro de un método propiedad, refiere al **objeto** que lo contiene.
- En otro caso, **this** se refiere a la variable global, **window** del documento HTML activo cargado en la ventana. Esto permite en objetos como `alert` o `document`, no referenciar al padre `window`.

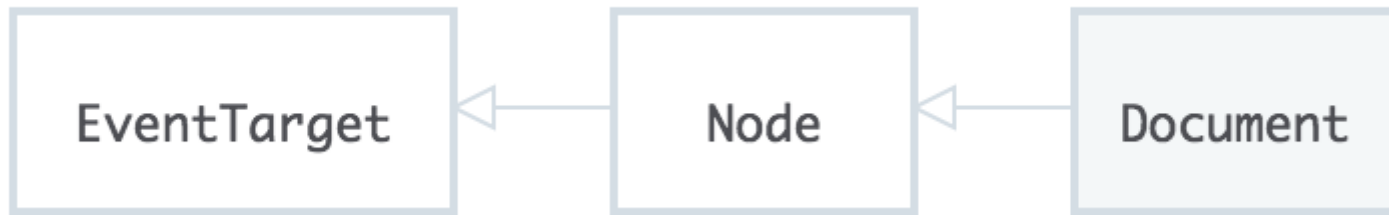
<https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Operadores/this>

```
window.alert('error');  
alert("sin window!");  
document.getElementById("identificado");
```

http://piruletas0.scienceontheweb.net/PHP/T4/HTML_DOM_W.html

8. GESTIÓN DE EVENTOS: EVENT TARGET INTERFACE

Clase del para gestionar eventos.



- EventTarget permite detectar/recibir los eventos que se producen en el DOM o HTML DOM
- Window, Document y Element (SVGELEMENT) soportan definir detectores de eventos, "event handlers".

EJEMPLO

```
<script type="text/javascript">
var Autor="";    var node0=null;

function borrar(val) { node0=val;
    console.info("nodo a borrar:"+node0.nodeName);
    console.info("val:"+val.nodeName);
    var node=val.parentNode;
    node.removeChild(node0);
    alert("¿Algo borrado ?");}

function recuperar(Id) {
    var node = document.getElementById(Id);
    node.appendChild(node0);
    alert("nodo recuperado:");}
</script>

<div>
<p id="borrar">Mueve ratón <span
onMouseOver="borrar(this) ;">AQUÍ</span>:</p>
<p><span onClick="recuperar('borrar') ;">PULSA
AQUÍ</span> para recuperar el original</p>
</div>
```

Cuestiones:

- ¿Qué hace el código anterior?
- ¿Qué vale `document.getElementById("id")`?
 - si el script se pone en el HEAD
 - si el script se pone al principio del BODY
 - si el script se pone al final del BODY
- ¿Qué referencia `this`?

http://piruletas0.scienceontheweb.net/PHP/T4/HTML_DOM.html

EJERCICIO: CREA UN CARRUSEL DE 5 FOTOS.

REGISTRO EVENTOS

```
x.addEventListener ("Evento", funcionEjecutar, boolean)
```

- El **evento** es el nombre del evento “click”, “load”, etc.
- La **funcionEjecutar** se ejecuta cuando se produce el evento
- El **Boolean es opcional** especifica si el evento debe ser capturado (true) o no (false, valor por defecto).

EJEMPLO DOM VALIDACIÓN DATOS FORMULARIOS

<http://piruletas0.scienceontheweb.net/PHP/T4/FormEventHandlerError.html>

```
function validar() {
    let r = true;
    let inputs = document.forms[0];
    inputs = inputs.getElementsByTagName('input');
    for(let i = 0; i < inputs.length; i++) {
        if (inputs[i].getAttribute('type')=='text' &&
            inputs[i].value.length < 4){
            let a = document.querySelector("#errores");
            a.textContent = "4 caracteres en " + inputs[i].getAttribute('name');
            inputs[i].focus();
            return false;
        }
    }
    return r;
}

function detectores(){
    let form = document.querySelector("#form_reg");
    form.addEventListener("submit", function(event){
        if (validar()==false){event.preventDefault();});
    });
}

detectores();
```

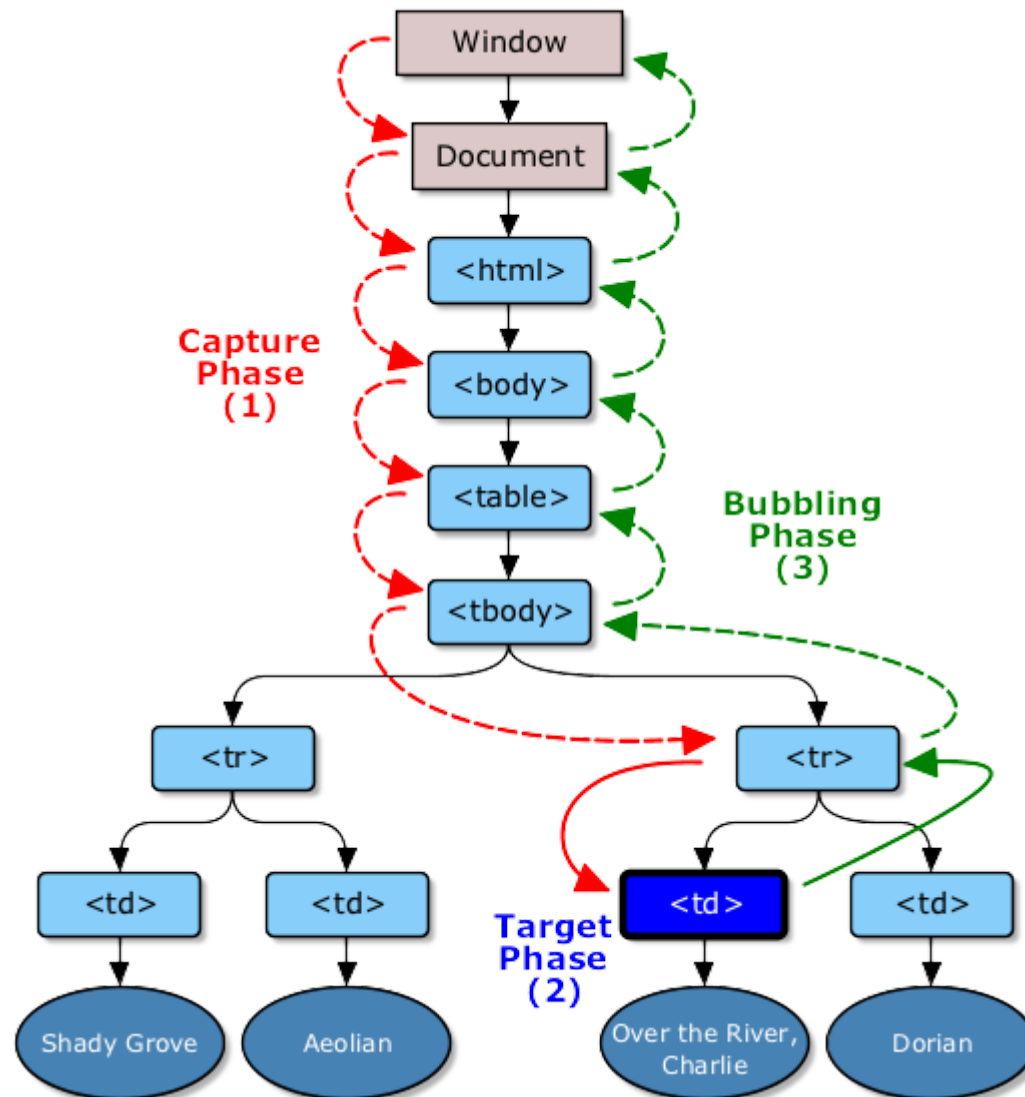
¿Funciona correctamente? Solución:

<http://piruletas0.scienceontheweb.net/PHP/T4/FormEventHandler.html>

DOMContentLoaded o load ¿Cual es mejor?

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

9. GESTIÓN DE EVENTOS: CAPTURE/BUBBLING



<http://www.thatjsdude.com/images/eventBubble.png>

CAPTURE/BUBBLING/TARGET

- **Capture** : Al realizar el cliente un evento, el navegador sabe que se ha producido un evento, y busca desde la raíz del DOM para ver donde se ha producido (*event object*)
- **Target** : Una vez sabe en qué elemento se ha producido (*event target object*) el explorador comprueba si tiene algún controlador en ese elemento, si existe lo ejecuta.
- **Bubbling** : Después de disparar el lanzador, busca hacia la raíz, si en los niveles padre del elemento hay algún otro controlador que se active con ese evento, en su caso lo lanzará. Esta etapa de su movimiento hacia arriba se llama la propagación de eventos

Ejemplo <http://piruletas0.scienceontheweb.net/PHP/T4/ejemCapture.html>

Cuestión:

Diferencia: `event.target` y `currentTarget`

10. JSON: DATOS ESTRUCTURADOS PARA INTERCAMBIO DE DATOS

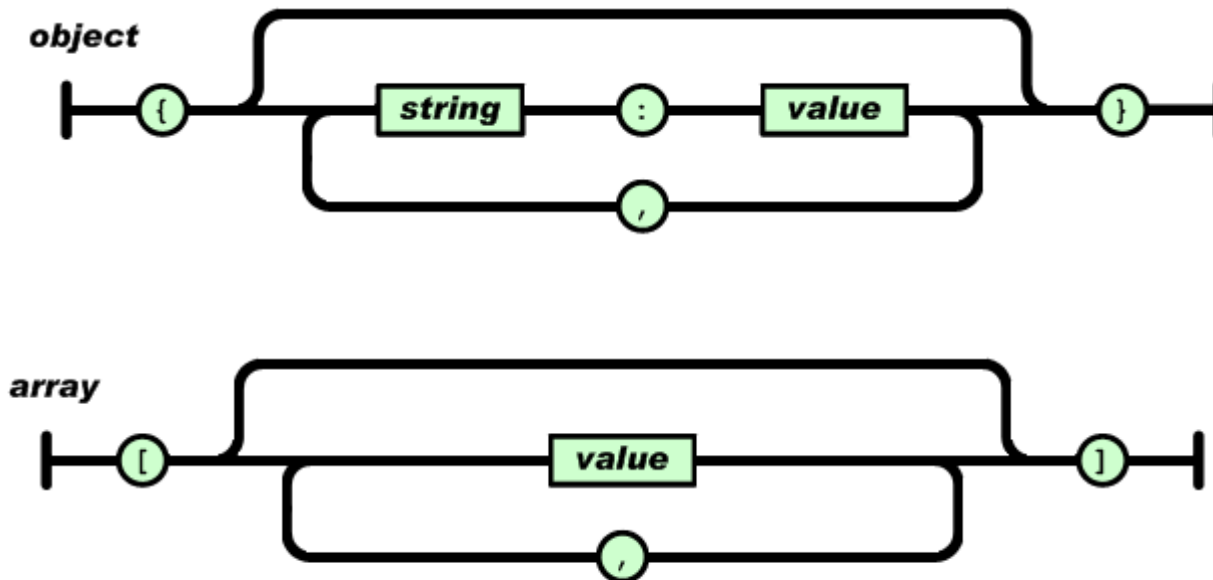
JSON, acrónimo de **JavaScript Object Notation**, es un formato ligero para el intercambio de datos.

Propiedades:

- Ligero: Formato ligero para el intercambio de datos.
- Simple: Formato sencillo de leer por humanos y de parsear/analizar por máquinas.
- Funcional: Los objetos JSON están tipificados.
- Abierto: Formato de texto independiente del lenguaje.
- Extensible: Formado por unas estructuras que pueden anidarse.

ESTRUCTURA DATOS JSON

- JSON es una sintaxis para serializar objetos, arreglos, números, cadenas, booleanos y nulos.
- La estructura de datos de JSON está basada en pares *nombre:valor*
- Las 2 estructuras principales son el objeto JSON y la lista JSON.



ACCESO ELEMENTOS OBJETO JSON

```
JSONdata={ "menu": {  
  "id": "file",  
  "value": "File",  
  "popup": {  
    "menuitem": [  
      { "value": "New", "onclick": "CreateNewDoc()" },  
      { "value": "Open", "onclick": "OpenDoc()" },  
      { "value": "Close", "onclick": "CloseDoc()" }  
    ]  
  }  
}
```

```
var menu = JSONdata.menu;  
fichero = JSONdata.menu.value;  
var submenu = JSONdata.menu.menuitem;  
var submenuVal = JSONdata.menu.popup.menuitem[ 0 ][ "value" ]
```

JSON EN HTTP

En HTTP se transmite texto por ello :

- Envío
 - Definir un objeto JSON
 - Convertir JSON a una cadena
- Recepción:

- Recibir cadena convertirla

```
- a objeto JSON en **JavaScript**.  
- a un diccionario o array en **PHP**.
```

- ¿y en NodeJS?

MÉTODOS DEL OBJETO JSON

JAVASCRIPT

- JSON .parse (): Analiza una cadena de texto con formato JSON retornando un objeto JSON. Se puede añadir una función para la transformación como parámetro.
 - JSON .stringify (): Devuelve un string a partir de un JSON. Se puede añadir una función para la transformación como parámetro.
- JSON
PHP

PHP

- **json_encode** (cadena): A partir de una cadena con sintaxis JSON la función que devuelve arreglo o objeto.
- **json_decode** (dato): Función que devuelve un objeto o arreglo en formato cadena con sintaxis JSON.

EJERCICIO: CONSTRUIR EL JSON

`{"bar":"new property","baz":3}`

```
var foo = {};  
foo.bar = "new property";  
foo.baz = 3;  
console.log (JSON.stringify(foo));  
//{"bar":"new property","baz":3}
```


EJEMPLO PHP JSON

<http://piruletas0.scienceontheweb.net/PHP/T4/JsonPhp.php>

```
<?php
$json = '[{"id_fruta": "1", "nombre_fruta": "Manzana", "cantidad": 100}, {"id_fruta": "2", "nombre_fruta": "Platano", "cantidad": 167}, {"id_fruta": "3", "nombre_fruta": "Pera", "cantidad": 820}]';
$arrayF = json_decode($json);
//OBTENER UN DATO DIRECTAMENTE DEL ARRAY.
print_r($arrayF[0]->nombre_fruta);
//RECORRER Y RECUPERAR VALORES JSON CON FOREACH.
foreach($arrayF as $obj){
    $id_fruta = $obj->id_fruta;
    $nombre_fruta = $obj->nombre_fruta;
    $cantidad = $obj->cantidad;
    echo "<br>\n".$id_fruta." ".$nombre_fruta." ".$cantidad;
}
?>
```

```
Manzana
1 Manzana 100
2 Plátano 167
3 Pera 820
```

EJEMPLO ENVÍO UNA TABLA AL CLIENTE COMO UN JSON

<http://piruletas0.scienceontheweb.net/PHP/T4/SendJSON.php>

```
header('Content-type: application/json');  
$result = $pdo->prepare("SELECT * FROM A_GrupoCliente");  
$result->execute();  
$datos = $result->fetchAll(PDO::FETCH_ASSOC);  
echo json_encode($datos);
```

10. CORS

- Por seguridad JS no permite que una aplicación que reside en un servidor pueda extraer datos de otro.
- El uso compartido de recursos de origen cruzado (CORS- introducido en HTML5) permite a las aplicaciones web de un dominio realizar solicitudes de dominio cruzado.
- Hay que habilitar estas llamadas tanto en el servidor como en el cliente.

<http://www.w3.org/TR/cors/>

<http://enable-cors.org>

CORS EN EL SERVIDOR

Se puede que habilitar CORS de dos formas:

- Con un encabezado (header) para permitir solicitudes de example.com:

```
<?php header('Access-Control-Allow-Origin: *'); ?>
```

- o en el fichero .htaccess

```
Header set Access-Control-Allow-Origin "*"
```

CORS EN EL CLIENTE

En la etiqueta script añadir atributo **crossorigin**:. Indica si la petición hecha a un servidor externo debe presentar credenciales CORS o no.

- **crossorigin=anonymous** : las peticiones CORS para el elemento tendrán la etiqueta "omit credentials" establecida.
- **crossorigin=use-credentials** : las peticiones CORS para el elemento no tendrán la etiqueta "omit credentials" establecida.

EJEMPLO Google maps con CORS y sin CORS

<http://piruletas0.scienceontheweb.net/PHP/T4/webDOMIframe.html>

BIBLIOGRAFÍA

- <https://dom.spec.whatwg.org/>
- <https://www.html5rocks.com/es/tutorials/internals/howbrowserswork/>
- <https://developer.mozilla.org/es/docs/Web/JavaScript>

