Programación Front End (nivel inicial)

Ing. Leonardo Torres, Lic. Gustavo Lafuente

FACUTLAD DE INGENIERÍA
Universidad Nacional de la Pampa

Argentina programa 4.0

Javascript

Javascript (js)

Es un lenguaje de programación **interpretado**. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente **del lado del cliente**, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web.

Javascript (js)

Es embebido en un documento HTML. Se coloca el head como en el body

Puede reconocer y responder a eventos de usuario sin ninguna conexión a la red

Ej. Chequeos de validación

EN EL **HEAD** O EN EL **BODY**, el resultado funcionalmente es igual, la diferencia radica que si lo colocamos al final del body, estamos cargando nuestro js al final, cuando ya el html y css fueron cargados.

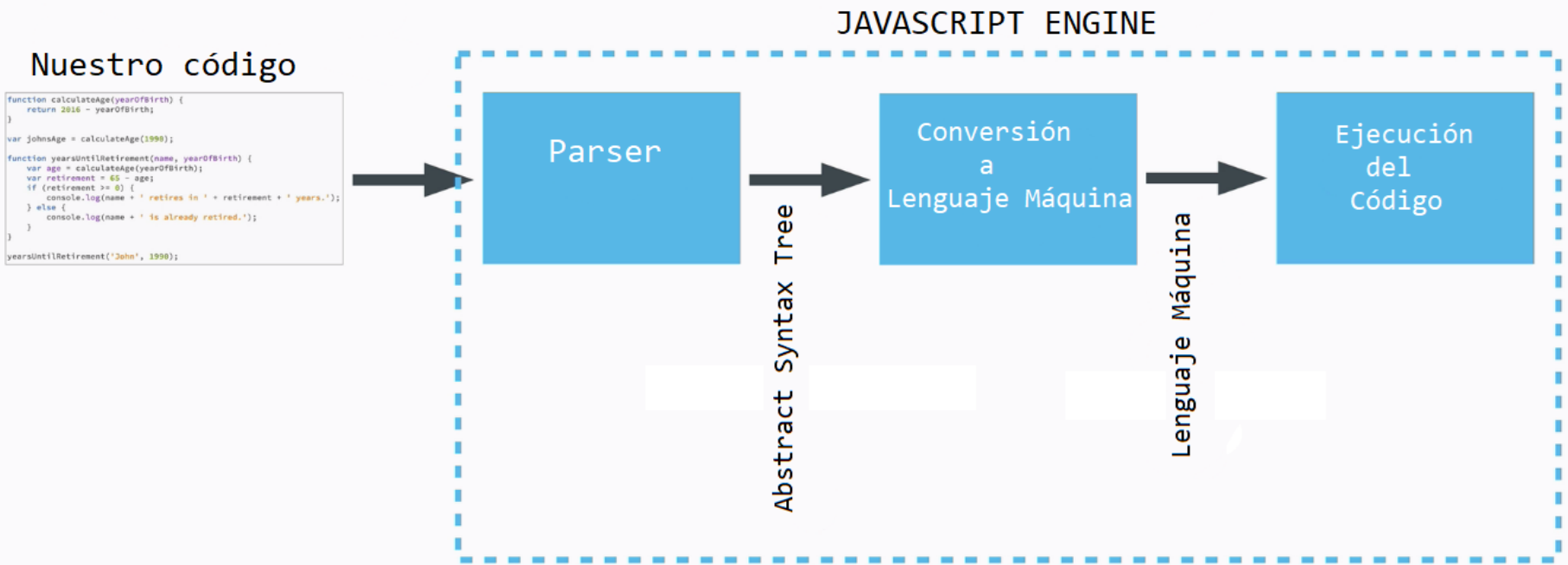
```
<HTML>
        <HEAD>
                <SCRIPT
Language="JavaScript"
        SRC="codigo.js">
                </SCRIPT>
        </HEAD>
        <BODY>



        </BODY>
</HTML>
```

```
<HTML>
        <HEAD>
        </HEAD>
        <BODY>



                <SCRIPT
Language="JavaScript"
        SRC="codigo.js">
                </SCRIPT>
        </BODY>
</HTML>
```

javascript.html

javascript.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="utf-8">
      <meta name="description" content="Your description goes here">
      <meta name="keywords" content="one, two, three">

    <title>J is for JavaScript</title>

    <script>
      alert("YO, I'm an annoying JavaScript popup")
    </script>

  </head>
  <body>
    <h1>JavaScript Was Here</h1>
  </body>
</html>
```


JAVASCRIPT ENGINE
Nuestro código
function calculateAge(yearOfBirth) {
return 2016 - yearOfBirth;
}
var johnsAge = calculateAge(1990);
function yearsUntilRetirement(name, yearOfBirth) {
var age = calculateAge(yearOfBirth);
var retirement = 65 - age;
if (retirement >= 0) {
console.log(name + ' retires in ' + retirement + ' years.');
} else {
console.log(name + ' is already retired.');
}
}
yearsUntilRetirement('John', 1990);
Parser
Abstract Syntax Tree
Conversión a Lenguaje Máquina
Lenguaje Máquina
Ejecución del Código

Del lado del Cliente

A través de los clientes Web del usuario, es compatible con la mayoría de los navegadores del mercado. Se tornó mas atractivo su uso en los dispositivos móviles.

Del lado del Servidor

En su inicios solo era ejecutado por el servidor Netscape Fiveware pero desde la liberación del código en el año 2000 han surgido otros servidores como **Node.Js**, **Jaxer, EJScript**, **RingoJS**, **AppengineJS** entre otros.

Javascript forma de uso

Se puede usar internamente

dentro del archivo html (**embebido**)

```
<html>
<TITLE>Ejemplo04.htm</TITLE>
<head>

<script language="Javascript">
  // Pedir confirmación para visitar una página
  function confirmar()
  {
    return confirm("Esta página contiene contenido para mayores de
18 años. ¿Lo cumples?")
  }
</script>

</head>
<body>
<a href="http://www.starwars.com/" onclick="return
confirmar()">Enlace a la página oficial de Star Wars</a>
</body>
</html>
```

O llamar a un archivo js con el código (**enlazado**)

```
1 | <script src="script.js" defer></script>
```

```
<HTML>
<TITLE>Ejemplo03.htm</TITLE>

<SCRIPT LANGUAGE="JavaScript">
//Recoger un dato por teclado y visualizarlo
var nom;
nom=prompt("Escribe tu nombre","NOMBRE");
alert("Mucho gusto "+ nom);

</SCRIPT>

</HTML>
```

Javascript sintaxis

Javascript es **case-sensitive** se respetan las mayúsculas y las minúsculas. No es lo mismo la variable **miVariable**, que **mivariable**.

FORMAS DE COMENTAR -------->

```
<SCRIPT>
//Este es un comentario de una línea
/*Este comentario se puede extender
por varias líneas.
Las que quieras*/
</SCRIPT>
```

Javascript ventajas

Velocidad. Al ser client-side, es muy rápido y cualquier función puede ser ejecutada inmediatamente en lugar de tener que contactar con el servidor y esperar una respuesta.

Simplicidad. Es relativamente simple de aprender e implementar.

Versatilidad. Encaja perfectamente con otros lenguajes y puede ser usado en una gran variedad de aplicaciones. puede insertarse en cualquier página independientemente de la extensión del fichero.

Carga del servidor. Al ejecutarse del lado del cliente reduce la carga en el servidor de la página web.

Javascript desventajas

Seguridad. A razón de que el código se ejecuta en la computadora del usuario en algunos casos puede ser explotado con propósitos malintencionados.

Confianza en el usuario. JavaScript es, algunas veces, interpretado diferente dependiendo en el navegador que sea ejecutado. Mientras que un código server-side siempre producirá el mismo resultado, código client-side puede ser un poco impredecible. No te preocupes demasiado por esto -mientras pruebes tu código en los navegadores más populares estarás a salvo.

Javascript políticas de seguridad

Content Security Policy (CSP) Política de Seguridad del Contenido

Es una capa de seguridad adicional que ayuda a prevenir y mitigar algunos tipos de ataque, incluyendo **Cross Site Scripting** (XSS) y ataques de inyección de datos. Estos ataques son usados con diversos propósitos, desde robar información hasta desfiguración de sitios o distribución de malware .

Si el sitio web no ofrece la cabecera CSP, los navegadores igualmente usan la política estándar mismo-origen.

Para habilitar CSP, necesitas configurar tu servidor web para que devuelva la cabecera HTTP **Content-Security-Policy**.

Consola de los navegadores

Dentro de los **inspectores de código** que vimos en clases pasadas, existe una **consola** muy utilizada para javascript, aunque no es el único uso.

Muestra mensajes de información, error o alerta que se reciben al hacer las peticiones para cargar desde la red los elementos incluidos en las páginas.

Además incluye inspectores o verdaderos depuradores de código.

También permite interactuar con la página, ejecutando expresiones o comandos de JavaScript. **VEAMOS UN EJEMPLO...**

Console log

Un símbolo exportado por Console.jsm es "**console**". a continuación se muestra un ejemplo de cómo accesarlo. lo que añade un mensaje a la Consola del Navegador.

```
console.log("Hello from Firefox code"); //output messages to the console
```

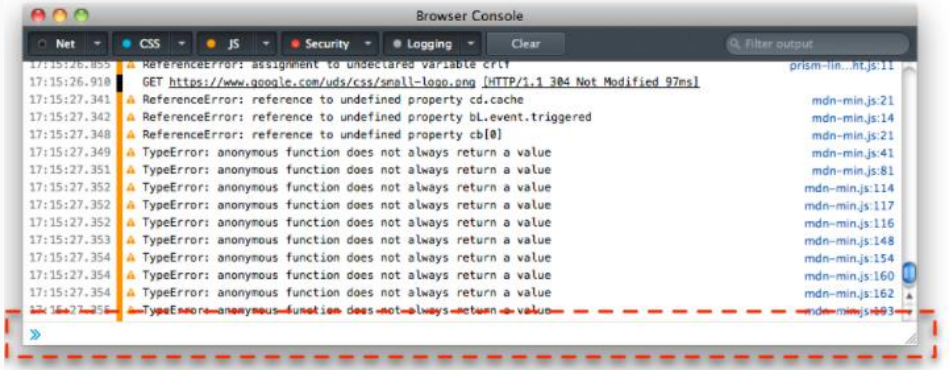
Consola de línea de comandos del Navegador

Al igual que la consola Web, el intérprete

de línea de comando le permite evaluar

expresiones JavaScript en tiempo real.

VEAMOS UN EJEMPLO...



Algo de sintaxis

- Es interpretado por un intérprete incorporado en el navegador que posibilita la ejecución de código generado dinámicamente.
- La sintaxis es muy similar a **Java o C++**, permitiendo una fácil adaptación. (como diferencia este puede o no terminar sus sentencias con ;)
- Es un lenguaje basado en **objetos** que no implementa ni el concepto de clase ni el mecanismo de herencia.
- Utiliza una jerarquía de objetos encabezada por el propio navegador (**DOM**)
- Permite la captura y tratamiento de una serie de eventos provocados tanto por el usuario como por el navegador
- Sirve para incorporar otros elementos tecnológicos como ActiveX, XML, DHTML (es la base de **AJAX**)

Comentarios	//Esta es una línea de comentario /*Este es un comentario multilinea*/	- Similares a C++
Código	alert('hola mundo');	- Serie de sentencias separadas por punto y coma (;). - Es opcional el punto y coma pero clarifica la separación las instrucciones
Impresión	document.write("Hola Mundo"+ "!!!");	- El texto puede ir con comillas simples o dobles - Incorporación de Tags embebidos en el texto, interpretados por el Navegador
Alertas	alert("Hola JavaScript");	- Permite mostrar una ventana de mensajes sobre la pantalla
Variables y Constantes	let variable1; var variable2; const variable3;	- **CONST:** Es una constante la cual NO cambiará su valor en ningún momento en el futuro. - **VAR:** Es una variable que SI puede cambiar su valor y su scope es local. - **LET:** Es una variable que también podrá cambiar su valor, pero solo vivirá(Funcionara) en el bloque donde fue declarada.
Concatenación	String1 = String1 + String2 + String3;	- concatena variables siempre y cuando no sean numéricos en su totalidad, de ser así sumará los valores.
Incrementales y Decrementales	x++; x--;	++ (Incremento) -- (decremento)

Operadores Numéricos	y=((a+b)-c*d/e)%f;	+ (suma) - (resta) * (Multiplicación) / (División) % (Módulo) Resto de la división

Operadores de Igualdad	if(a==b)
	if(a!=b)
	if(a>b)
	if(a>=b)
	if(a<b)
	if(a<=b)
		• = = es igual a
		• != no es igual
		• > es mayor que
		• >= es mayor o igual a
		• < es menor que
		• <= es menor o igual a

Operadores Lógicos	if(a==b && a==c)
	if(a==b || a==c)
	if(!a==b)
		• && AND lógico
		• | | OR lógico
		• ! NOT

Sentencias condicionales	If (condición a evaluar) {
	/*Bloque de código*/ }
	Else {
	/*Bloque de código*/ }
		• If ..... Else ....
		• If ..... Else if ..... Else ....

control de flujo	For (var i=0;i=10; i++) {
	/*Bloque de código*/ }
	
	For (variable in object) { sentencias }
	
	While (condición) { /*Bloque de código*/ }
		• **For** bucle
		• **for...in** Itera una variable específica
		• **While** Ejecuta un bloque de código mientras la evaluación de la condición sea verdadera

Javascript arreglos (array)

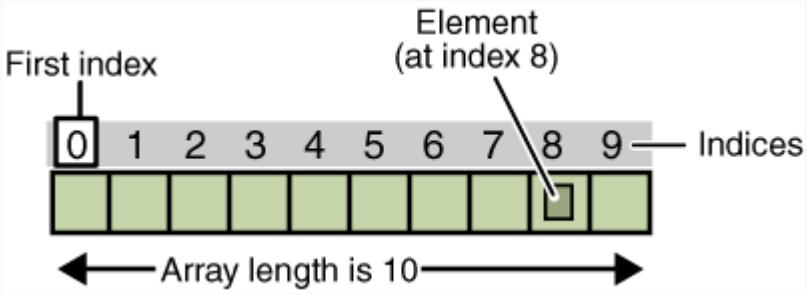
Son útiles para crear, ordenar y manipular lista usando bucles

Creación de un arreglo

- **nombreArray= new Array();** //arreglo vacío
- **nombreArray= new Array(10);** //arreglo con 10 items para almacenar

Propiedad length Da la longitud de un arreglo

- **nombreArray.length();**



Extensión de Arreglo

- Simplemente si necesitamos extender el tamaño de un arreglo.

```
gatos=new Array(4);
gatos[0]=Tom;
gatos[1]=Lucas;
gatos[2]=Adam;
gatos[3]=Luis;
```

```
gatos=new Array("Tom", "Lucas", "Adam", "Luis");
```

```
let myArray = [1, 2, 3, 4, 5];

let myReversedCopy = myArray.slice().reverse();

// Outputs: [1, 2, 3, 4, 5]
console.log(myArray);

// Outputs: [5, 4, 3, 2, 1]
console.log(myReversedCopy);
```

Existen metodos para manejo de arreglos como *sort(), Reverse(), pop(), toSrting(), join(), push(),*

Javascript Objetos

JS permite definir objetos personalizados, que contiene un conjunto de métodos preconstruidos

```
<SCRIPT>
    var remera=new Object
    remera.color="roja"
    remera.size=8
    remera.type="algodon"
    document.write(remera.color + "<BR>")
    document.write(remera.size + "<BR>")
    document.write(remera.type + "<BR>")
</SCRIPT>
```

```
<SCRIPT>
    var miDia=new Date();
    document.write(miDia + "<BR>")
    var x=miDia.toGMTString();
    document.write(x + "<BR>")
</SCRIPT>
```

Javascript ¿Qué es DOM?

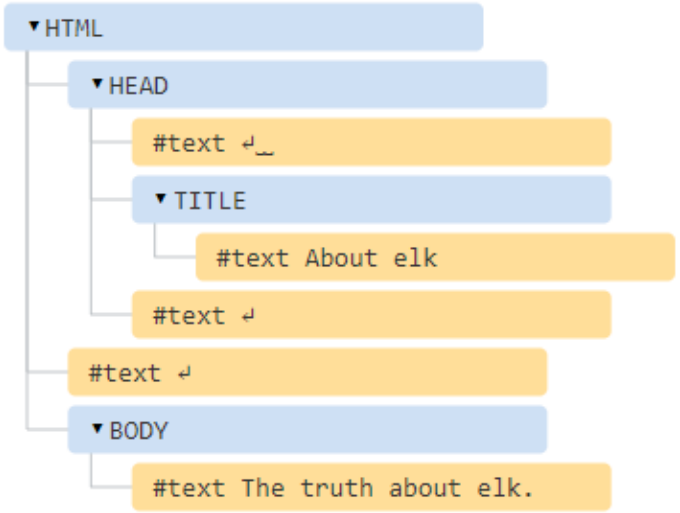
El modelo de objeto de documento (**DOM**) es una interfaz de programación para los documentos **HTML** y **XML**. Facilita una representación estructurada del documento y define de qué manera los programas pueden acceder, al fin de modificar, tanto su estructura, estilo y contenido.

El DOM es una representación completamente orientada al objeto de la página web y puede ser modificado con un lenguaje de script como JavaScript.

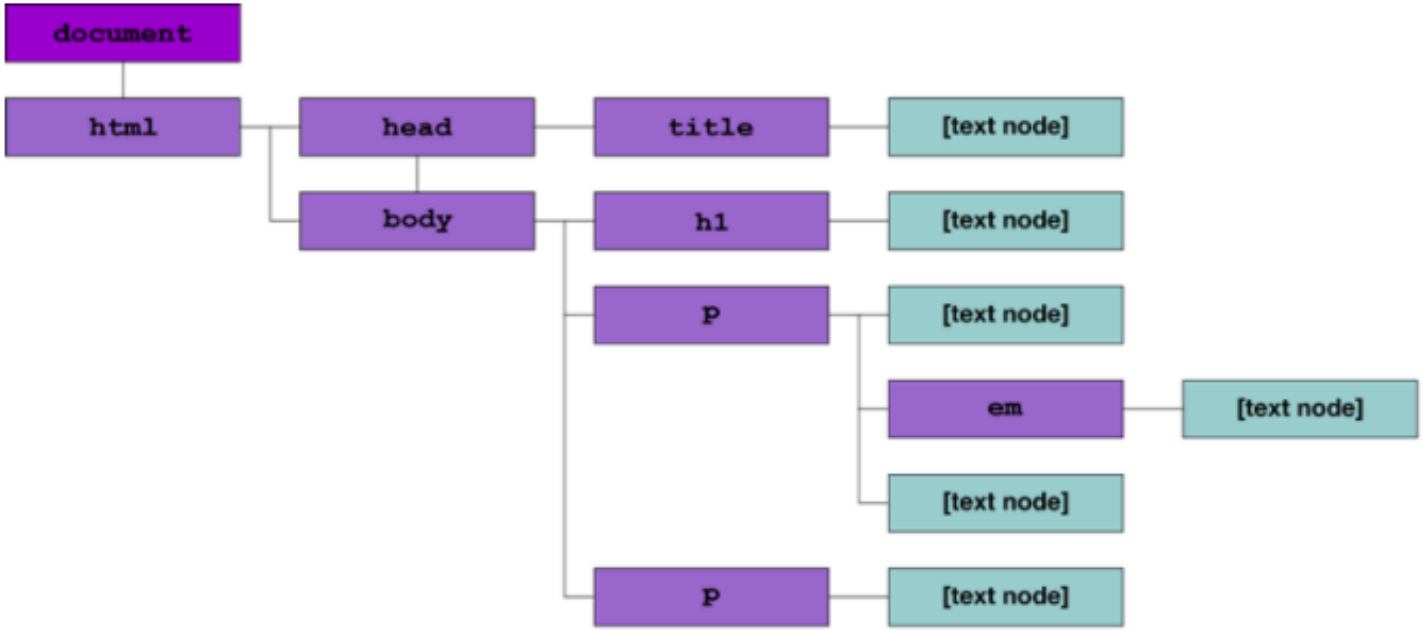
El **W3C DOM** estándar forma la base del funcionamiento del DOM en muchos navegadores modernos.

Por ejemplo, el **DOM** de W3C especifica que el método **getElementsByTagName** en el código de abajo debe devolver una lista de todos los elementos <p> del documento:

```
<!DOCTYPE HTML>
<html>
<head>
  <title>About elk</title>
</head>
<body>
  The truth about elk.
</body>
</html>
```



```html
<html>
  <head>
    <title>Esto es un Documento</title>
  </head>
  <body>
    <h1>Esto es una cabecera</h1>
    <p id="TextoExcitante">
      Esto es un párrafo! <em>Excitante</em>!
    </p>
    <p>
      Esto también es un párrafo, pero no es ni de lejos tan excitante como el último.
    </p>
  </body>
</html>
```

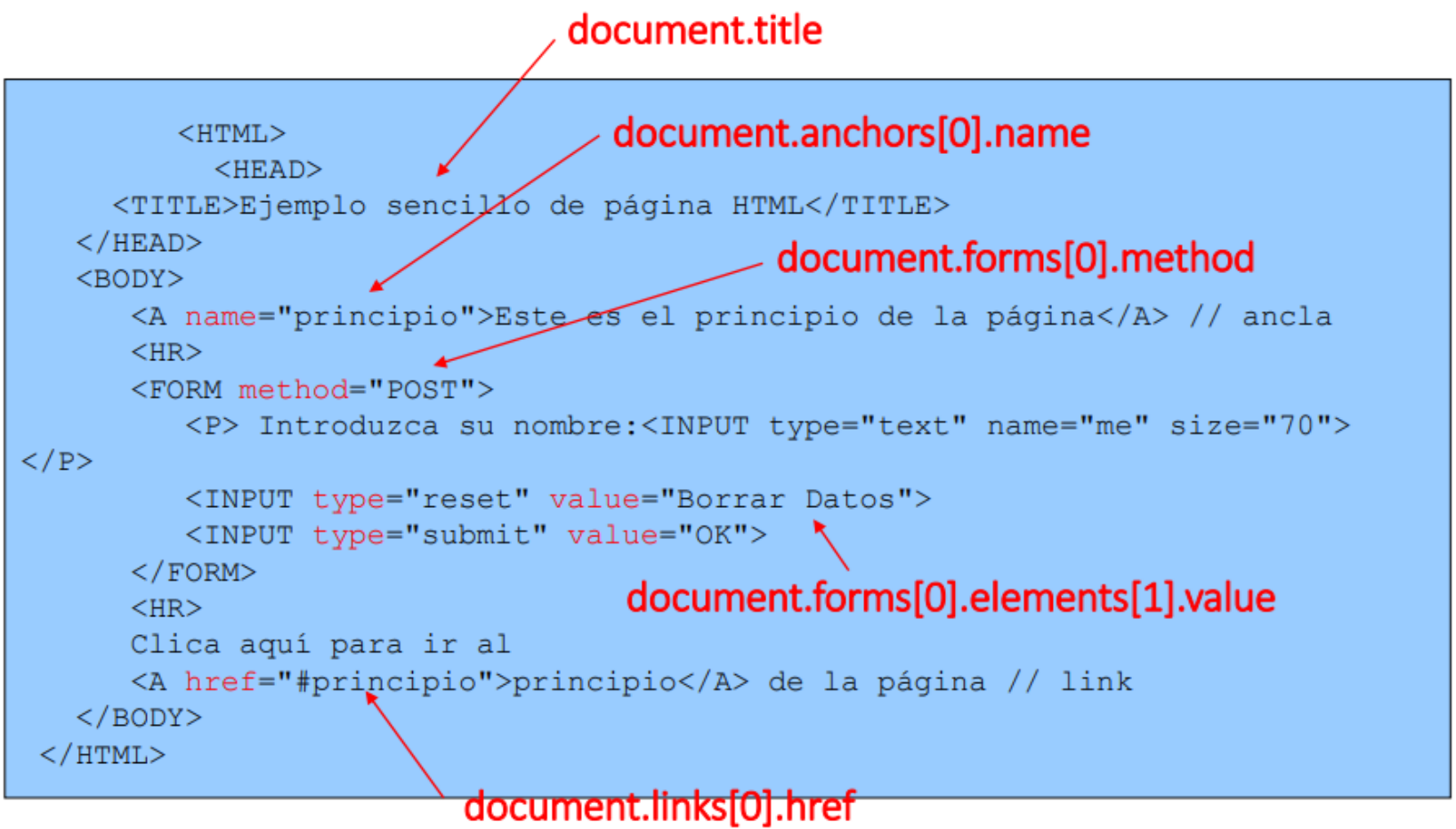


Javascript ¿Que es DOM?

veamos unos ejemplos:

```
paragraphs = document.getElementsByTagName ("p");
// paragraphs[0] es el primer elemento <p>
// paragraphs[1] es el segundo elemento <p>, etc.
alert (paragraphs [0].nodeName);
```

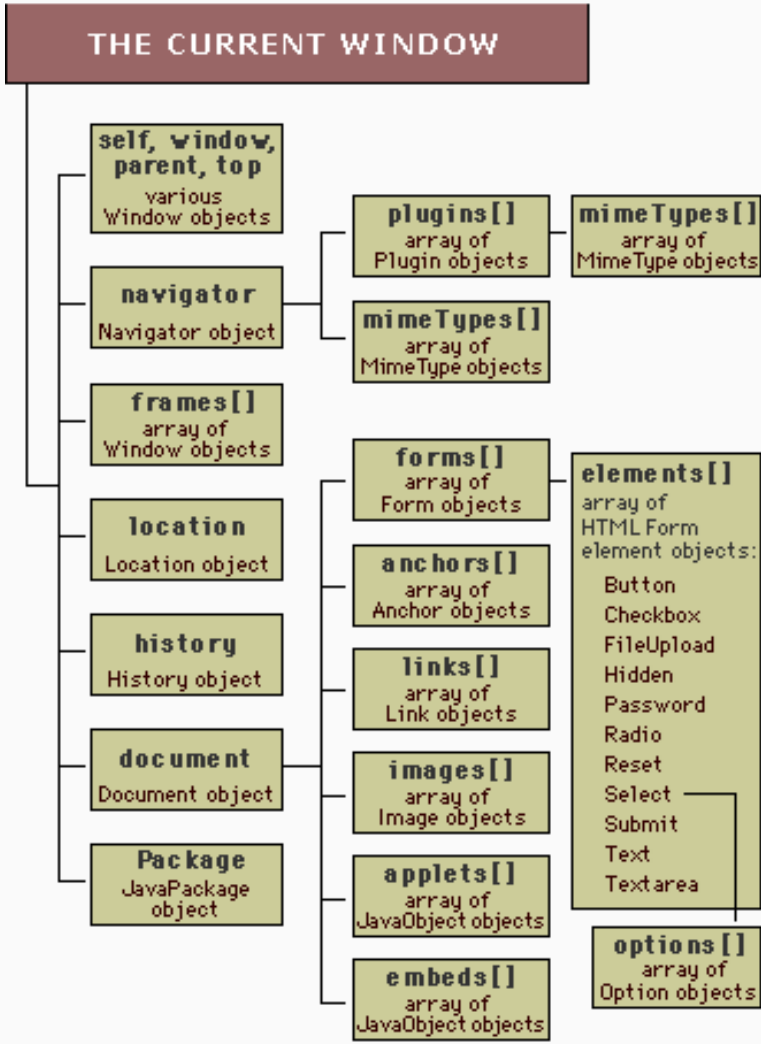
```
var table = document.getElementById("table");
var tableAttrs = table.attributes; // Node/interfaz Element
for (var i = 0; i < tableAttrs.length; i++) {
  // interfaz HTMLTableElement: atributo border
  if(tableAttrs[i].nodeName.toLowerCase() == "border")
    table.border = "1";
}
// interfaz HTMLTableElement: atributo summary
table.summary = "nota: borde aumentado";
```




```
<HTML>
  <HEAD>
<TITLE>Ejemplo sencillo de página HTML</TITLE>
</HEAD>
<BODY>
<A name="principio">Este es el principio de la página</A> // ancla
<HR>
<FORM method="POST">
<P> Introduzca su nombre:<INPUT type="text" name="me" size="70">
</P>
<INPUT type="reset" value="Borrar Datos">
<INPUT type="submit" value="OK">
</FORM>
<HR>
Clica aquí para ir al
<A href="#principio">principio</A> de la página // link
</BODY>
</HTML>
```

Jerarquía de objetos del Navegador

- window
  - + history
  - + location
  - + document <BODY> ... </BODY>
    - - anchor <A NAME="..."> ... </A>
    - - applet <APPLET> ... </APPLET>
    - - area <MAP> ... </MAP>
    - - form <FORM> ... </FORM>
      - + button <INPUT TYPE="button">
      - + checkbox <INPUT TYPE="checkbox">
      - + fileUpload <INPUT TYPE="file">
      - + hidden <INPUT TYPE="hidden">
      - + password <INPUT TYPE="password">
      - + radio <INPUT TYPE="radio">
      - + reset <INPUT TYPE="reset">
      - + select <SELECT> ... </SELECT>
        - - options <INPUT TYPE="option">
      - + submit <INPUT TYPE="submit">
      - + text <INPUT TYPE="text">
      - + textarea <TEXTAREA> ... </TEXTAREA>
    - - image <IMG SRC="...">
    - - link <A HREF="..."> ... </A>
    - - plugin <EMBED SRC="...">
  - + frame <FRAME>



Javascript documentación oficial

Para ver la sintaxis y semántica del lenguaje lo más conveniente es ver la documentación oficial.

Link útiles

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide

https://developer.mozilla.org/es/docs/Web/JavaScript

https://devdocs.io/javascript/

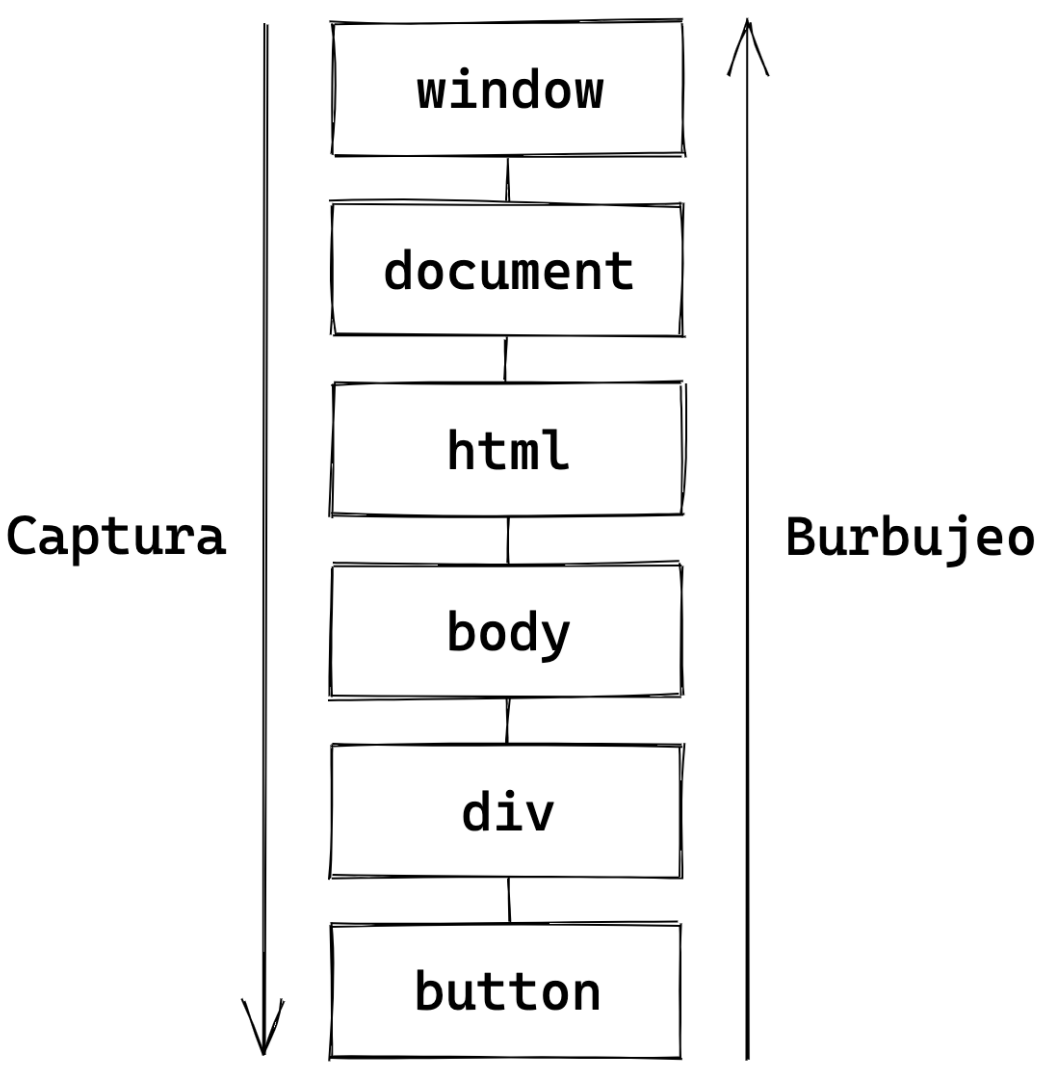
https://developer.chrome.com/docs/devtools/

Javascript - DOM

El Documento o nodo "**Document**" es el **nodo raíz del DOM**. En HTML se corresponde con la etiqueta **<html>** que engloba la página. Una vez cargada la página podemos acceder a él desde javascript.

La forma normal de acceder al documento es mediante la palabra clave **document**.

veamos algunas propiedades y metodos...


window
document
html
body
div
button
Captura
Burbujeo

Propiedades tipo array

nos devuelven un **array** con todos los elementos que tienen unas ciertas características.

Propiedad	Explicación
`document.links`	Devuelve un array con todos los **enlaces** que hay en el documento
`document.forms`	Devuelve un array con todos los **formularios** del documento
`document.images`	Devuelve un array con todas las **imágenes** del documento
`document.anchors`	Devuelve un array con todos los enlaces tipo **referencia** (`<a name="..">...</a>`) del documento

manipulación de nodos.

Métodos y propiedades nos permiten manipular los nodos, de forma que podemos buscarlos y guardarlos en variables, acceder a su contenido y atributos, modificarlos, crear nodos nuevos, insertarlos, reemplazarlos, eliminarlos, o hacerles una copia.

Los métodos o propiedades se aplican a veces directamente sobre el nodo raíz document y otras veces sobre el nodo tipo element que queremos actuar, el cual lo hemos buscado previamente.

Método	Explicación
`getElementsByTagName()`	Crea un array con todos los elementos cuya etiqueta sea igual a la que se le pasa en el parámetro. Se aplica normalmente a `document`. Ej.:`parrafos=document.getElementsByName("p").`
`getElementsByName()`	Crea un array con todos los elementos cuya atributo `name` sea igual al que se le pasa en el parámetro. Se aplica normalmente a `document`.
`getElementById()`	Devuelve el elemento cuyo atributo `id` sea igual al que se le pasa en el parámetro. Se aplica normalmente a `document`. Como el valor de id debe ser único en cada elemento, sólo puede devolver uno.
`createElement()`	Crea un nodo de tipo Element (etiqueta). Se aplica normalmente a `document`; `document.createElement()`. Como parámetro se pasa el nombre de la etiqueta.
`createTextNode()`	Crea un nodo de tipo Text (texto). Se aplica normalmente a `document`; `document.createTextNode()`. Como parámetro se pasa el texto.
`appendChild()`	Inserta un nodo dentro de otro, de manera que se aplica al nodo padre, y como parámetro pasamos el nodo hijo: `nodoPadre.appendChild(nodoHijo)`. Si el nodo padre tiene otros nodos hijos, el nuevo se coloca el último de todos.
`insertBefore()`	Inserta un nodo delante de otro de referencia, Lo insertamos en el nodo padre del de referencia, y pasamos como primer parámetro el nodo nuevo, y como segundo el de referencia: `nodoPadre.insertBefore(nodoNuevo,nodoRef);`
`replaceChild()`	Remplaza un nodo (nodoNuevo) por otro (nodoRef). Lo insertamos en el nodo padre del nodoRef, y pasamos como primer parámetro el nodo nuevo, y como segundo el de referencia: `nodoPadre.replaceChild(nodoNuevo,nodoRef);`
`removeChild()`	Elimina un nodo. Lo aplicamos en el nodo padre del nodo a eliminar: `nodoPadre.removeChild(nodo);`
`hasChildNodes()`	Aplicado a un nodo, comprueba si éste tiene nodos hijos. (devuelve true o false).
`getAtribute()`	Lee el valor de un atributo. Se aplica al nodo que contiene el atributo. Ej.:`valor=enlace.getAttribute("href").`
`setAtribute()`	crea y define el valor de un nuevo atributo. Se aplica al nodo que contendrá el atributo. Pasamos dos parámetros, el primero el nombre del atributo, y el segundo su valor: Ej.:`valor=enlace.getAttribute("target","_blank").`
`removeAtribute()`	elimina un atributo. Se aplica al nodo que contiene el atributo. Pasamos dos parámetros, el primero el nombre del atributo, y el segundo su valor: Ej.:`enlace.removeAttribute("class").`

Propiedad	Explicación
`innerHTML`	Devuelve o cambia el contenido en lenguaje HTML que contiene el nodo o etiqueta , pudiendo así crear nuevos nodos dentro de uno ya existente. ej.; `elemento.innerHTML="<h3>hola mundo</h3>"`
`parentNode`	Devuelve el elemento padre del nodo al que se le aplica. (sólo lectura).
`firstChild`	Devuelve el primer elemento hijo del nodo al que se le aplica. (sólo lectura).
`lastChild`	Devuelve el último elemento hijo del nodo al que se le aplica. (sólo lectura).
`nodeName`	Devuelve el nombre (nombre de etiqueta o atributo) del nodo al que se le aplica (en mayúsculas).
`nodeType`	Devuelve el tipo de nodo al que se le aplica.
`nodeValue`	Devuelve el valor del nodo al que se le aplica. (valor del atributo "value", o el texto en nodos de texto).
`attributes`	Devuelve un array con todos los atributos del nodo al que se le aplica (sólo lectura).
`childNodes`	Devuelve un array con todos los nodos hijo del nodo al que se le aplica (sólo lectura).
`tagName`	Devuelve el nombre en mayúsculas de la etiqueta del nodo al que se le aplica (sólo lectura).
`previousSibling`	Devuelve el elemento inmediatamente anterior al que se le aplica.
`NexSibling`	Devuelve el elemento inmediatamente posterior al que se le aplica.
`style`	mediante la propiedad `style` podemos acceder al código CSS y ver, modificar o crear nuevo código para el nodo o etiqueta al que se le aplica.
`this`	Devuelve el mismo elemento al que se le aplica o en el que está (por ejemplo en eventos)

Propiedades de informacion del documento

Propiedad	Explicación
`cookie`	Devuelve todas las parejas nombre/valor de las cookies guardadas, como una cadena de texto. ej.;`misCookies=document.cookie`.
`domain`	Devuelve el dominio del servidor que ha cargado el documento.
`lastModified`	Devuelve la fecha de la última modificación del documento.
`readyState`	Devuelve el estado de carga del documento.
`referrer`	Devuelve la URL del enlace que ha cargado el documento actual.
`title`	Permite cambiar o devuelve el título de la página actual (etiqueta `title` del `head`).
`URL`	Devuelve la URL completa de la página actual.
`clientHeight`	Devuelve la altura de la parte visible del documento (en píxeles).
`clientWidth`	Devuelve la anchura de la parte visible del documento (en píxeles).
`scrollHeight`	Devuelve la altura total del documento (en pixels), incluyendo las zonas ocultas por barras de desplazamiento.
`scrollWidth`	Devuelve la anchura total del documento (en pixels), incluyendo las zonas ocultas por barras de desplazamiento.
`scrollTop`	Devuelve la distancia entre el borde superior del documento en sí y el borde superior de la parte que vemos del documento.
`scrollLeft`	Devuelve la distancia entre el borde izquierdo del documento en sí y el borde izquierdo de la parte que vemos del documento.

Más métodos del documento

Método	Explicación
`toString()`	Convierte cualquier elemento en una cadena de texto. ej.; `cadena=elemento.toString()`
`focus()`	Pone el foco del documento en el elemento al que se le aplica
`blur()`	Quita el foco del documento del elemento al que se le aplica
`click()`	Tiene el mismo efecto que si se efectuara un click de ratón en el elemento al que se le aplica.

USOS

```
document.getElementById("firstName");
```

```
document.createTextNode("Este es un nodo de texto");
```

```
document.getElementsByTagName("span");
```

```
document.getElementsByClassName("container");
```

```
document.createElement("div");
```

```
document.createAttribute("href");
```

```
// Elemento base
<div class="container">
  <p>Bienvenido a este sitio</p>
</div>

// Obtenermos el elemento al que queremos cambiar el contenido
let container = document.querySelector(".container");

// Reemplazamos el contenido
container.outerHTML = "<span>Bienvenidos<span>";

// Resultado
<span>Bienvenidos<span>
```

```
// Nuestro input
<input class="form-control"
       id="persona-nombre"
       placeholder="Nombre completo" />

// Obtenemos el input
const input = document.querySelector("#persona-nombre");

// Obtenemos los atributos
input.className // salida: "form-control"
input.id // salida: "persona-nombre"
input.placeholder // salida: "Nombre completo"
```

USOS

```
input.removeAttribute("name");

// input final
<input class="form-control"
       id="persona-nombre"
       placeholder="Nombre completo" />
```

```
// Nuestro HTML
<div class="parent">
  <p class="p-child">Titulo</p>
  <span class="span-child">Subtitulo</span>
</div>

// Obtenemos el nodo a eliminar
const nodoAEliminar = document.querySelector(".p-child");

// Eliminamos el nodo
nodoAEliminar.removeChild();
```

```
// Nuestro HTML
<div class="parent">
  <p class="p-child">Titulo</p>
  <span class="span-child">Subtitulo</span>
</div>

// Obtenemos el nodo padre
const parent = document.querySelector(".parent");

// Obtenemos el nodo a eliminar
const nodoAEliminar = document.querySelector(".p-child");

// Eliminamos el nodo
parent.removeChild(nodoAEliminar);
```

```
// Elemento base
<div class="container">
  <p>Bienvenido a este sitio</p>
</div>

// Obtenermos el elemento al que queremos cambiar el contenido
let container = document.querySelector(".container");

// Reemplazamos el contenido
container.innerHTML = "<span>Bienvenidos<span>";

// Resultado
<div class="container">
  <span>Bienvenidos<span>
</div>
```

```
input.setAttribute("name", "fullName");

// Ahora el input quedará de la siguiente manera

<input class="form-control"
       id="persona-nombre"
       placeholder="Nombre completo"
       name="fullName" />
```

Importancia de anidar correctamente

Si los elementos son anidados de manera inadecuada pueden generarse problemas:

```
<p>Estos elementos han sido <strong>incorrectamente
</p>anidados </strong>
```

El árbol que resulta de esto se encuentra incorrectamente anidado del todo, por tanto generará errores inesperados en los navegadores.

Eventos y listeners

Los **Eventos DOM HTML** permiten a JavaScript para registrar diferentes controladores de eventos de elementos en el DOM.

Los eventos se utilizan normalmente en combinación con las funciones y la función no se ejecutarán antes de que ocurra el evento.

Un elemento que posee un evento asignado, accionara este en el momento que se produzca la acción, esto lo hace gracias al **listener (EventListener),** que tiene la capacidad de "**escuchar**" los sucesos o acciones que se producen en cada elemento del DOM.

Evento	Descripción	Elementos para los que está definido
onblur	Deseleccionar el elemento	<button>, <input>, <label>,<select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	<button>, <input>, <label>,<select>, <textarea>, <body>
onkeydown	Pulsar una tecla y no soltarla	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	Página cargada completamente	<body>
onmousedown	Pulsar un botón del ratón y no soltarlo	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón *"sale"* del elemento	Todos los elementos
onmouseover	El ratón *"entra"* en el elemento	Todos los elementos
onmouseup	Soltar el botón del ratón	Todos los elementos
onreset	Inicializar el formulario	<form>
onresize	Modificar el tamaño de la ventana	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	
onunload	Se abandona la página, por ejemplo al cerrar el navegador	<body>

Plug-in o Complemento

Es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la interfaz de programación de aplicaciones.

Permiten:

- Que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones.
- Reducir el tamaño de la aplicación.
- Separar el código fuente de la aplicación.

Plug-in o Complemento

Plugin javascript

- egg.js
- elevator.js
- fullPage
- interactive-maps-generator
- native-flash-radio
- wowbook-a-flipbook-jquery-plugin
- resizable-multicolor-countdown
- media-boxes-portfolio-responsive-grid

Plugin CSS

- https://wordpress.org/plugins/advanced-css-editor/
- https://es.wordpress.org/plugins/simple-custom-css/
- https://wordpress.org/plugins/theme-junkie-custom-css/

Jquery

Es una biblioteca multiplataforma de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Es software libre y de código abierto, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código.

La sintaxis de JQuery está diseñada para facilitar la navegación por un documento.

https://jquery.com/

Jquery

Para agregarla

la podemos descargar (lo mas recomendable)

<script src="jquery-3.5.1.min.js"></script>

o usarla desde CDN (Content Delivery Network)

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

CDN (Content Delivery Network)

Una red de distribución de contenidos es una red superpuesta de computadoras que contienen copias de datos, colocados en varios puntos de una red con el fin de maximizar el ancho de banda para el acceso a los datos de clientes por la red.

https://cloud.google.com/

librerias

https://developers.google.com/speed/libraries

jQuery Sintaxis

$(selector).action()

- Un signo $ para definir / acceder a jQuery
- Un (selector) para "consultar (o buscar)" elementos HTML
- Una acción jQuery () que se realizará en los elementos

jQuery Sintaxis ejemplo

- **$(this).hide()** – Oculta el elemento actual.
- **$("p").hide()** – Oculta todos los elementos.
- **$(".test").hide()** – Oculta todos los elementos con class="test".
- **$("#test").hide()** – oculta el elemento con id="test".

```
$(document).ready(function(){
  //colocar aquí los metodos jQuery
});
```

```
$(document).ready(function(){
        $("button").click(function(){
                $("p").hide();
        });
});
```

```
$("#hide").click(function(){
  $("p").hide();
});
```

```
$("#show").click(function(){
  $("p").show();
});
```

2. Ejemplos de síntaxis

JAVASCRIPT	jQUERY
document.getElementById(“id)	$(“#id”)
e = document.getElementById(“id); e.stye.fontSize=”3rem”;	$(“#id”).css(“font-size”,”2rem”);

e =document.getElementById(“id);
e.
addEventListener('click',function(event) {
….
});
	$(“#id”).click(function(event) {
	…
	}

jQuery documentación oficial

- https://api.jquery.com/category/selectors/jquery-selector-extensions/
- https://plugins.jquery.com/
- https://www.jqueryscript.net/
- taller de jquery https://desarrolloweb.com/manuales/taller-jquery.html

Prototype

Es un **framework escrito en JavaScript** que se orienta al desarrollo sencillo y dinámico de aplicaciones web. Es una herramienta que implementa las técnicas AJAX y su potencial es aprovechado al máximo cuando se desarrolla con **Ruby On Rails**.

http://prototypejs.org/

Vanilla JS

Vanilla JS actualmente es el framework más usado en Internet, siendo usado por **Google, Amazon, Twitter** y muchas otras webs, incluida Xitrus.

A la hora de acceder al DOM por IDs es el doble de rápido que **Dojo** y 100 veces más rápido que **Prototype JS** cuando accedemos a los elementos por el nombre de la etiqueta.

http://vanilla-js.com/

Dibujando con html5 canvas

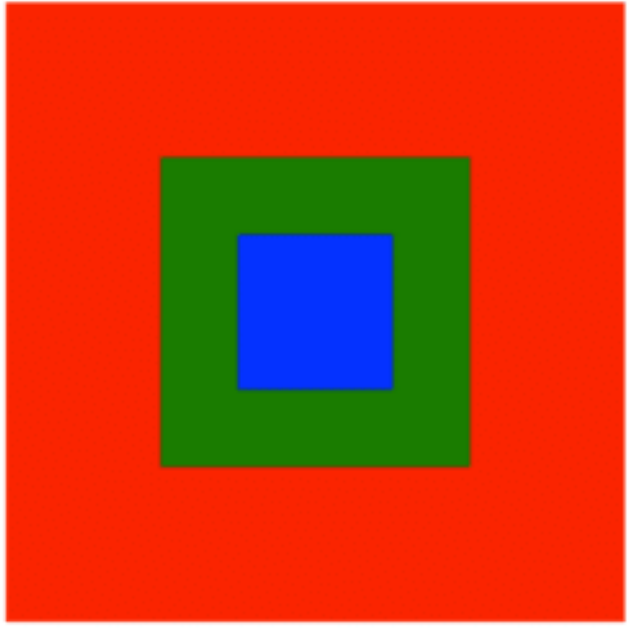
<canvas> es un elemento HTML el cual puede ser usado para dibujar gráficos usando **scripts** (normalmente JavaScript). Este puede, por ejemplo, ser usado para dibujar gráficos, realizar composición de fotos o simples (y no tan simples) animaciones. Las imágenes a la derecha muestran algunos ejemplos de implementaciones <canvas> las cuales se verán en un futuro en este tutorial.

<canvas> fue introducido primero por Apple para el Mac OS X Dashboard y después implementado en Safari y Google Chrome.

El tamaño por defecto del canvas es **300px * 150px [ancho (width) * alto (height)]**. Pero se puede personalizar el tamaño usando las propiedades height y width de CSS. Con el fin de dibujar gráficos en el lienzo <canvas> se utiliza un objeto de contexto de JavaScript que crea gráficos sobre la marcha.

ejemplos basicos, https://www.w3schools.com/html/html5_canvas.asp

```
<html>
<body>
  <canvas id="myCanvas" width="200" height="200" />
  <script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "red";
    ctx.fillRect(0, 0, 200, 200);
    ctx.fillStyle = "green";
    ctx.fillRect(50, 50, 100, 100);
    ctx.fillStyle = "blue";
    ctx.fillRect(75, 75, 50, 50);
  </script>
</body>
</html>
```

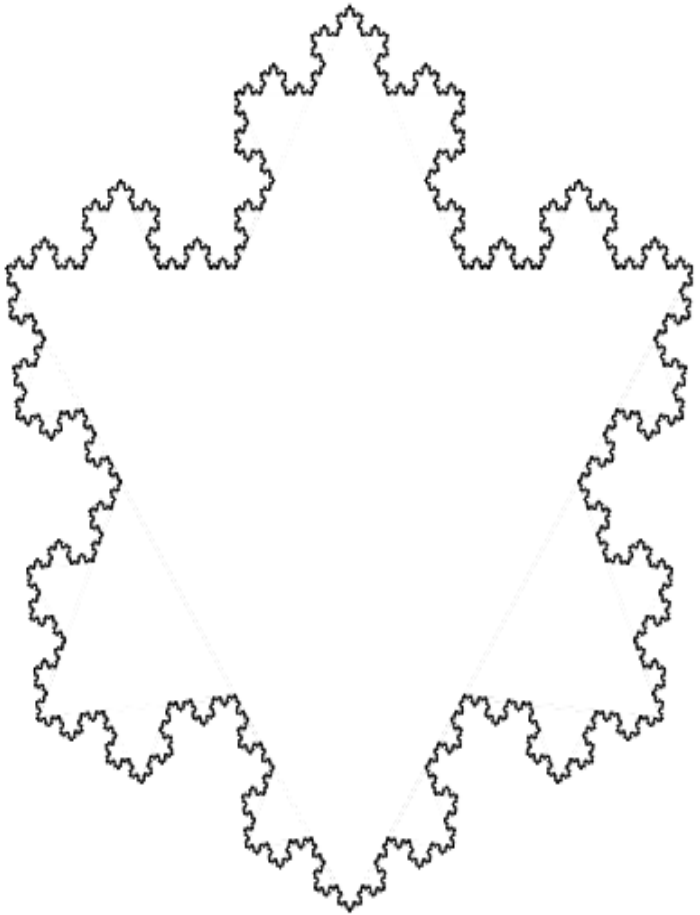


JS

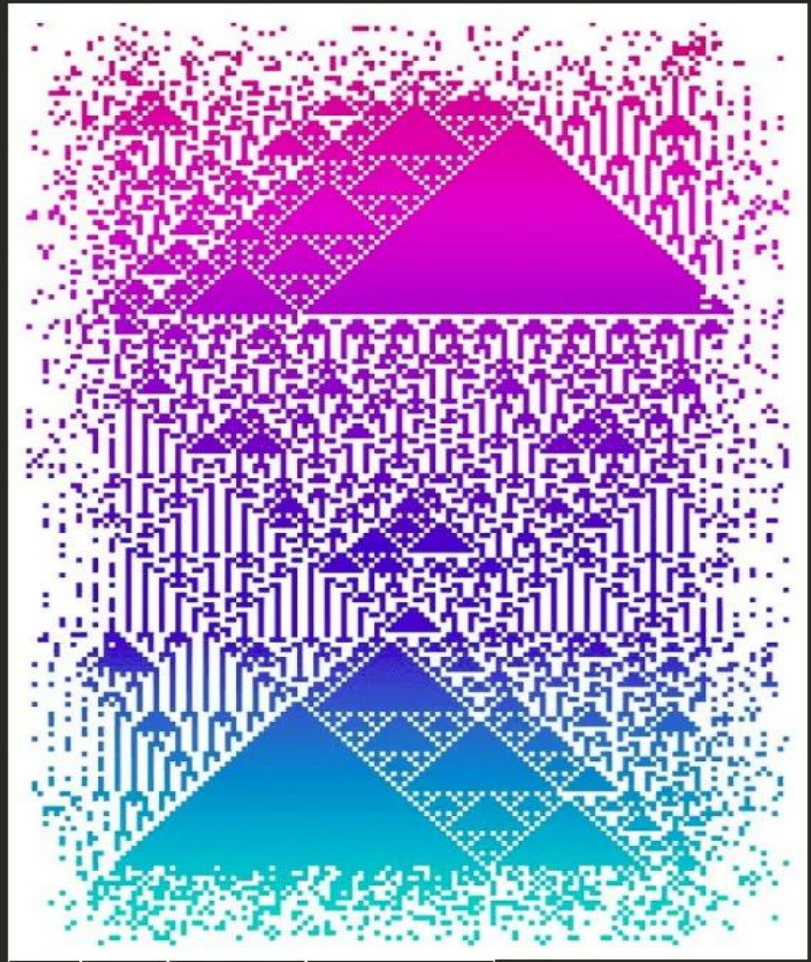
Run

PREVIEW

```
var width = 400,
    height = 400;
var canvas = $canvasPlayground.getCanvas(width, height);
var context = canvas.getContext("2d");

fractal([20, 120], [400, 120], 5);
fractal([210, 390], [20, 120], 5);
fractal([400, 120], [210, 390], 5);

function fractal(A, B, depth){
    if (depth < 0){
        return null;
    }

    var C = divide(add(multiply(A, 2), B), 3);
    var D = divide(add(multiply(B, 2), A), 3);
    var F = divide(add(A, B), 2);

    var V1 = divide(minus(F, A), length(F, A));
```



```
/*

Use the HTML canvas to code your t-shirt's image
https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial

Global variables:

  canvas:   HTML Canvas element, 2400x3200 pixels
  ctx:      Canvas context
  WIDTH:    2400 (image width in pixels)
  HEIGHT:   3200 (image height in pixels)

Tips:
  - Click 'Run' to render your image to the canvas
  - Click 'Save' to save your code to your browser's local storage
  - Runtime errors will appear in the debug console

*/
const SIZE = 900

// Draw the design to the canvas
const drawImage = (x, y, size) => {
  const opacity = Math.sqrt(size / SIZE)
  ctx.fillStyle = `rgba(0, 0, 255, ${opacity})`
  drawCircle(x, y, size)
  if (size <= 20) return
  drawImage(x + size / 3, y + size / 2, 3 * size / 4)
}

// Draw a circle to the canvas
const drawCircle = (x, y, size) => {
  // ctx.clearRect(x - size / 2, y - size / 2, size, size)
  ctx.beginPath();
  ctx.arc(x, y, size / 2, 0, Math.PI * 2)
  ctx.fill();
}

// Run the drawing
drawImage(SIZE / 2  + WIDTH / 10, SIZE / 2 + HEIGHT / 10, SIZE)
```



Run | Save | Download | Make T-Shirt

Programación Front End 2023

FACUTLAD DE INGENIERÍA
Universidad Nacional de la Pampa

Argentina programa 4.0