Javascript

Programación Web I

Comisión Miércoles Noche

DIS. Alicia Rosenthal

Tec. Willian Dos Reis

Comisión Viernes Mañana

Ing. Gabriel Panik

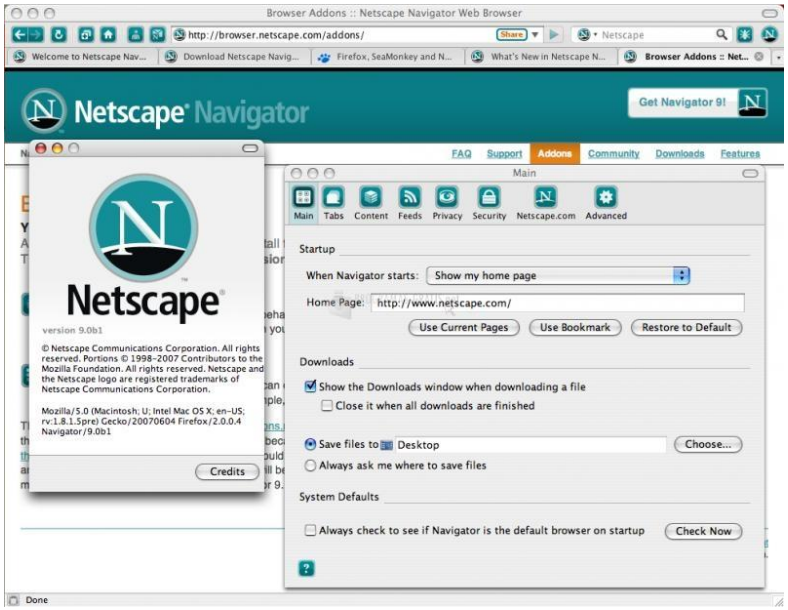
JAVASCRIPT

- Lenguaje Interpretado
- Multiplataforma
- Debilmente Tipado y Dinámico
- Orientado a Objetos y Eventos
- Imperativo
- Basado en prototipos

JAVASCRIPT

- Creado por Brendan Eich, programador de Netscape v2.0 en 1995. ->Livescript ->javascript

- Microsoft lanzó Jscript para Internet Explorer 3.0

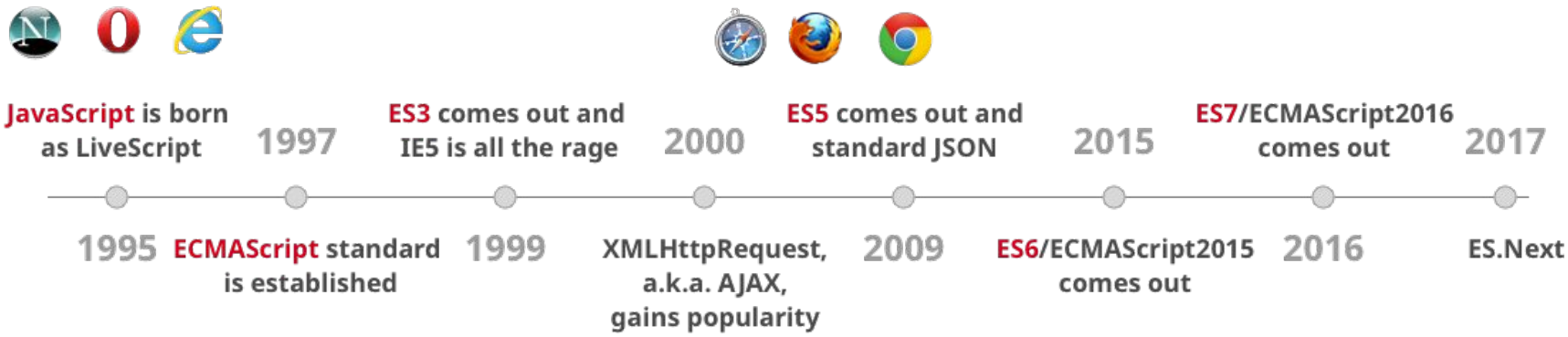


ECMAScript

ECMA

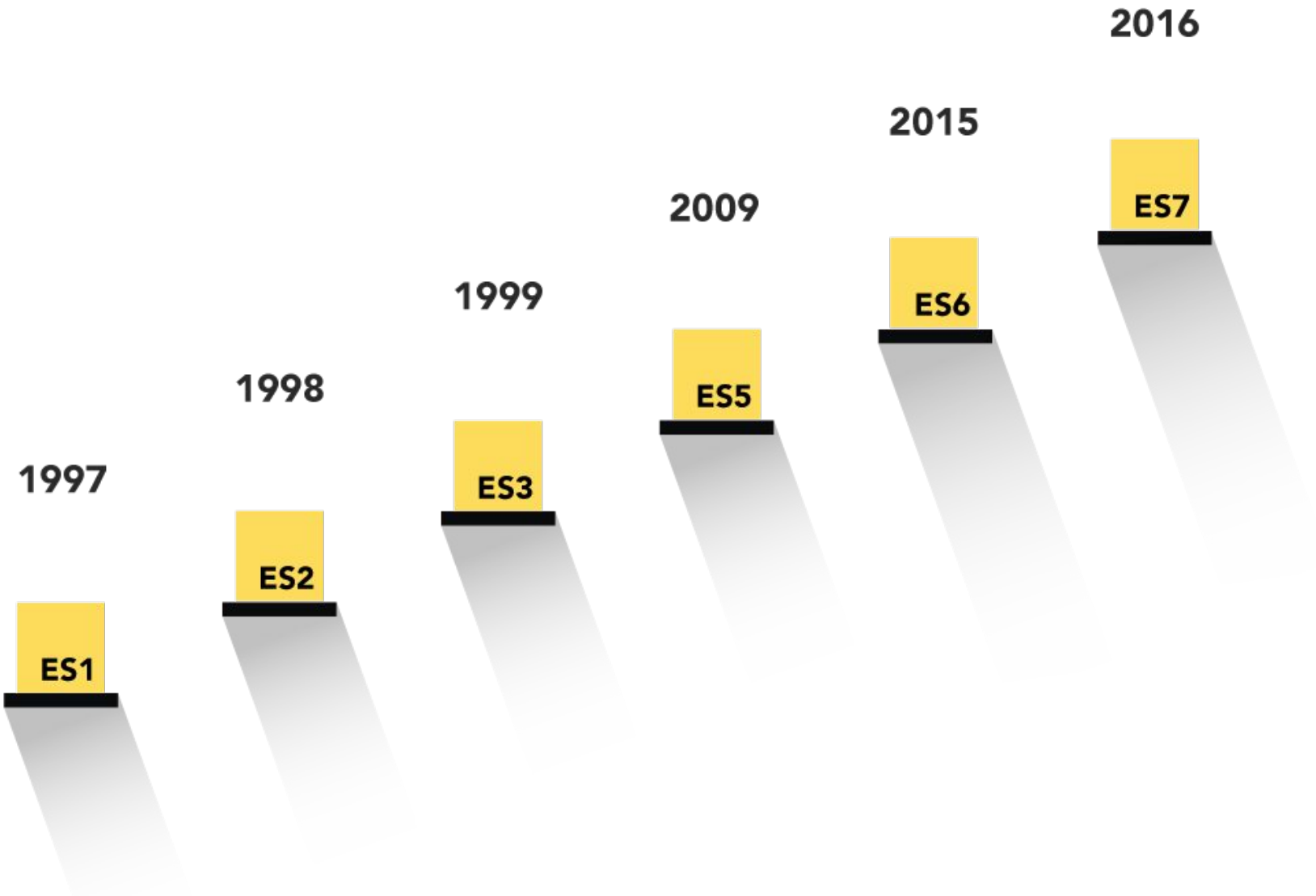
European Computer Manufacturers Association

https://www.w3schools.com/js/js_versions.asp



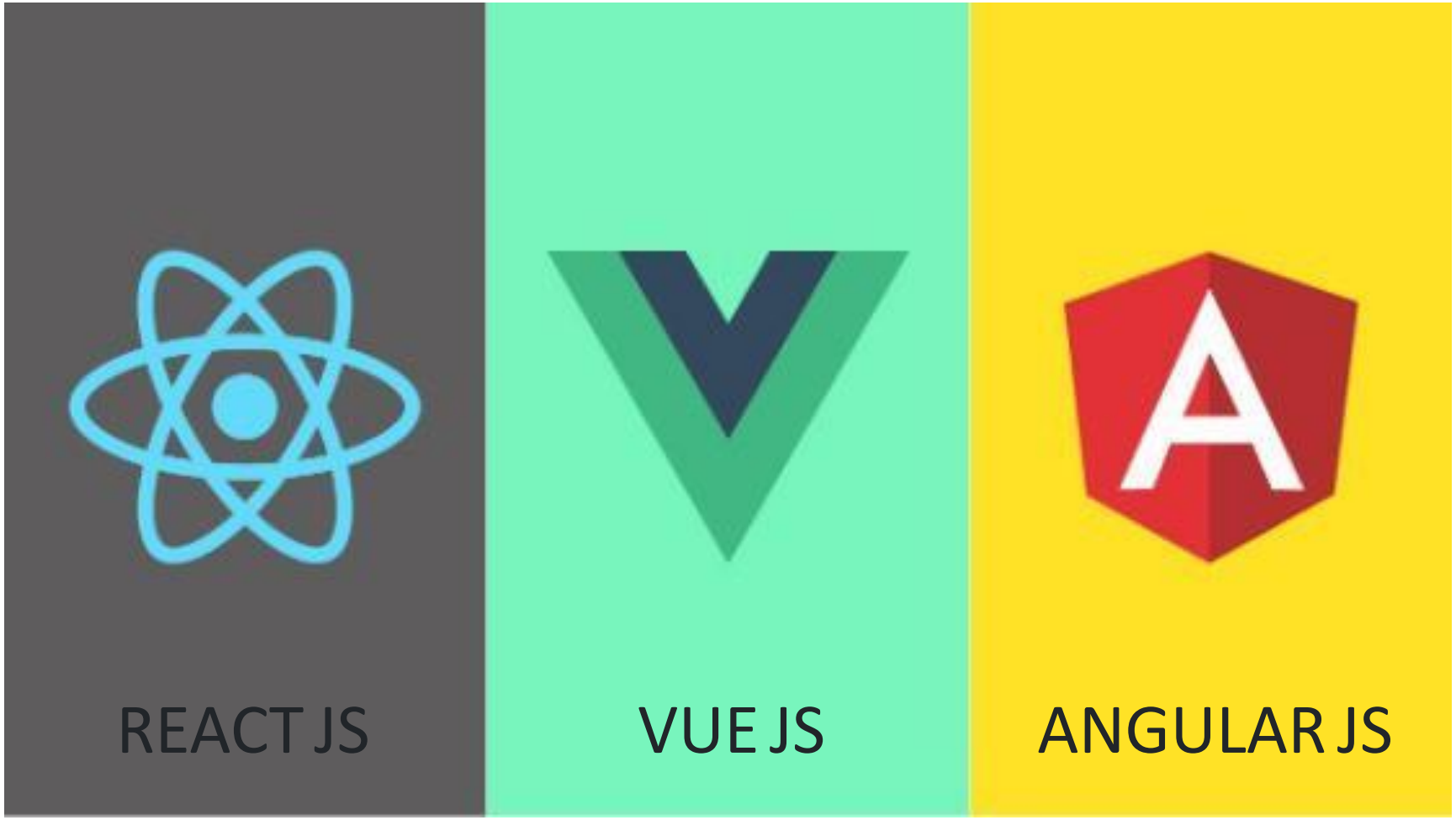
Estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa

ECMAScript

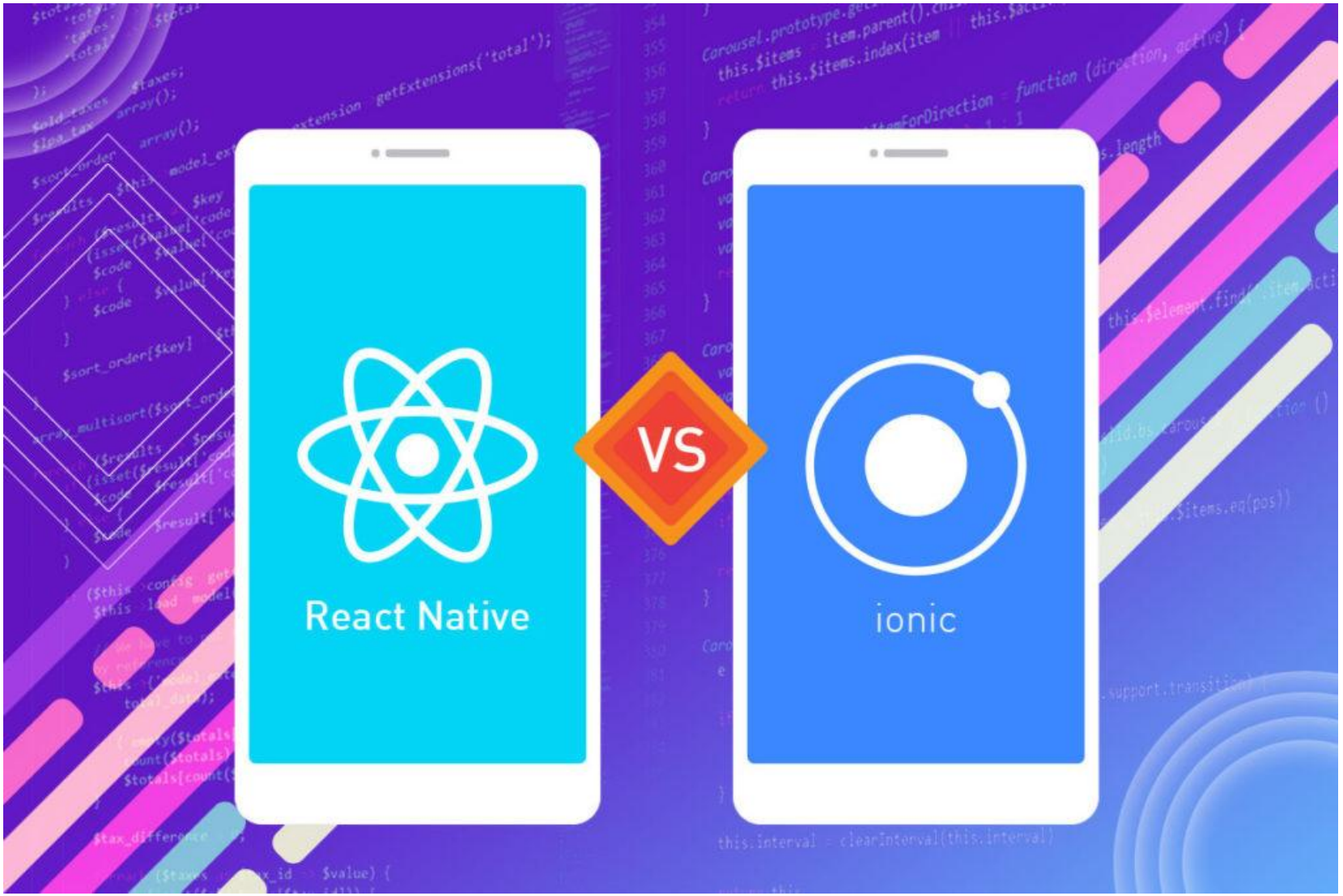


Librerías JAVASCRIPT

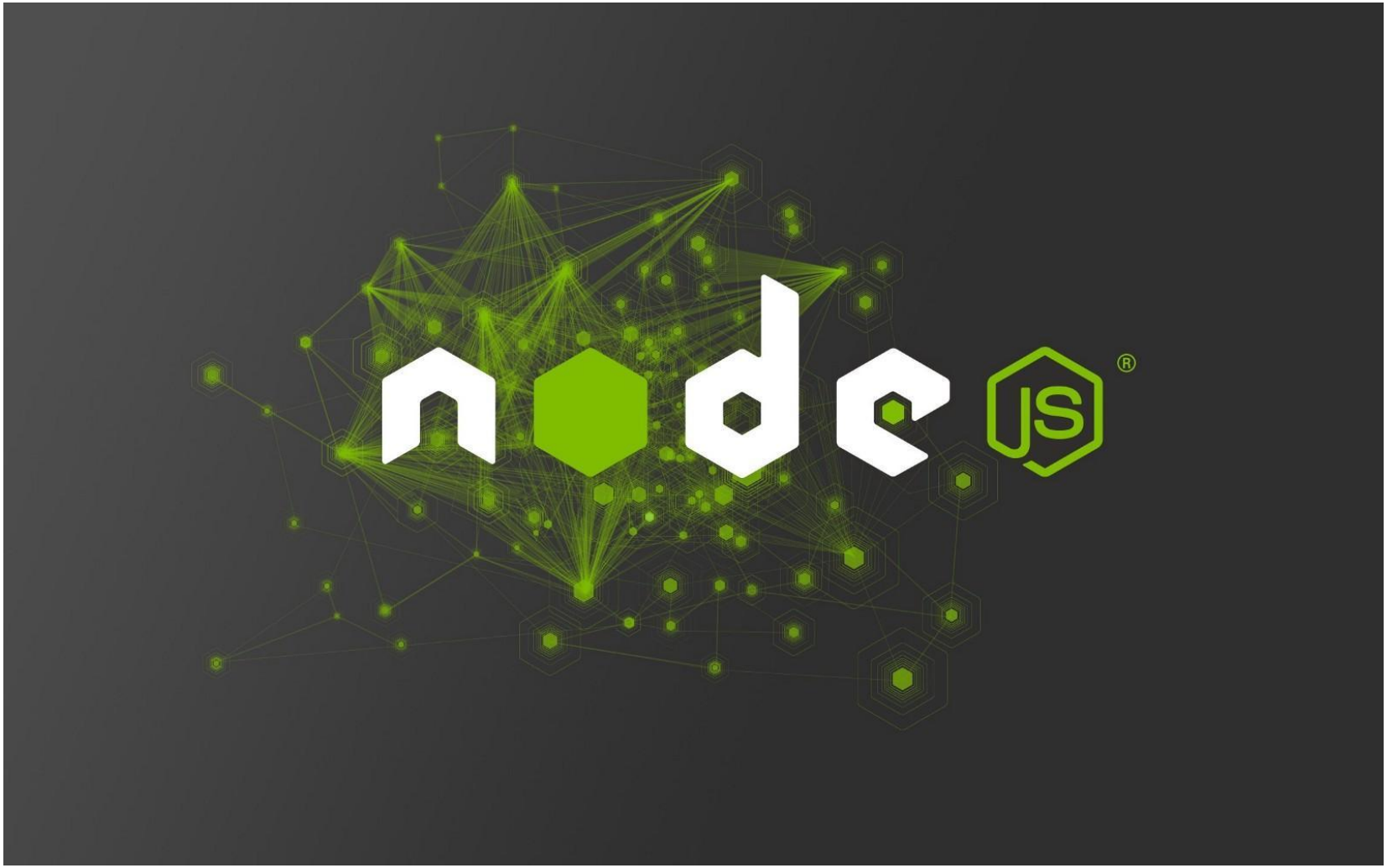
Frameworks JAVASCRIPT



Frameworks JAVASCRIPT



Frameworks JAVASCRIPT



INCLUIR JS en HTML

```
<head>
   <meta charset="UTF-8">
   <script>
      alert ("Primer Script");
   </script>
</head>
```

INCLUIR JS en Archivo Externo

1. Crear un archivo con la extensión .js dentro de una carpeta llamada js
2. Vincular el código javascript a nuestro html

```
<head>
    <meta charset="UTF-8">
    <script src="js/codigo.js">
    </script>
</head>
```

MOSTRAR DATOS

- //popup
```
alert ("");
```

- //en el documento HTML
```
document.write ("");
```

- //en la consola
```
console.log("");
```

SINTAXIS JAVASCRIPT

- **No** se tienen en cuenta los espacios en blanco y las nuevas líneas
- Se distinguen las mayúsculas y minúsculas
- **No** se define el tipo de las variables
- **No** es necesario terminar cada sentencia con el carácter de punto y coma
- Se pueden incluir comentarios
  - Una sola línea //
  - Varias líneas (entre /* y */)

Variables

- Almacenar un dato
- Se crean con la palabra reservada var

```
//declaración
var nombre;

//inicialización
nombre ="Pablo";
```

Variables

El nombre de una variable:

- **NO** puede empezar con un número
- **NO** puede usar carácteres especiales, ni espacios en blanco
- **Se puede** utilizar el signo $ y el guión bajo (_)
- Distingue Mayúsculas de Minúsculas

Variables

Convención para los nombres de Variables:

camelCase

Primera palabra toda en minúscula, primera letra de cada palabra siguiente Mayúsculas

Por ejemplo:

```
var nombreApellido;
var numeroDocumento;
```

Tipos de Datos Primitivos

TIPO	Valores
Number	Numéricos incluidos números negativos y decimales
String	Cadenas de Texto. Comillas simples o dobles.
Boolean	Valores posibles true o false
Null	Nulos o vacíos
Undefined	Sin definir, cuando una variable no fue inicializada.
Symbol	Nuevo tipo en JavaScript introducido en la versión ECMAScript Edition 6

PEDIR DATOS

Palabra reservada prompt

```
prompt("Mensaje al usuario");

var nombre;

nombre = prompt("Ingrese su nombre");
```

Constantes

- Almacenar un dato NO Variable
- Se crean con la palabra reservada const

```
//declaración
const PI = 3.14;
```

Constantes

Convención para los nombres de Constantes:

UPPERCASE

Toda la palabra en MAYÚSCULA

Por ejemplo:

```
const PI;
const IVA;
```

Operadores de Asignación

- =
- +=, -=, *=, /=

Ejemplos:

```
numeroUno = 8;
numeroDos = 3;
numeroUno += 3;
```

Operadores Aritméticos

- +
- -
- *
- /
- %
- ++ (Incremento)
- -- (decremento)

Operadores Lógicos

- Negación (!)
- AND (&&)
- OR (||)

Operadores Lógicos

Ejemplos:

```
a = 8;
b = 3;
c = 3;
document.write( (a == b) && (c > b) );
document.write( (a == b) && (b == c) );
document.write( (a == b) || (b == c) );
document.write( (b <= c) );
document.write( !(b <= c) );
```

Operadores Relacionales

- >, >= Mayor, mayor o igual
- <, <= Menor, menor o igual
- == Igual
- != Distinto

Operadores Relacionales

```
var numero1 = 3;
var numero2 = 5;
resultado = numero1 > numero2;
resultado = numero1 < numero2;
numero1 = 5; numero2 = 5;
resultado = numero1 >= numero2;
resultado = numero1 <= numero2;
resultado = numero1 == numero2;
resultado = numero1 != numero2;
numero1 = 5; numero2 = 4;
resultado = numero1 == numero2 ;
resultado = numero1 === numero2;
```

PARSEAR DATOS

- `isNaN(variable);`
  `//devuelve true si no lo es`
- `parseInt(variable);`
  `//convierte en número entero`
- `parseFloat(variable);`
  `//convierte en número flotante`
- `String(variable);`
  `//convierte en cadena de texto`

CONDICIONALES IF ELSE

```
if(condicion){

} else{

}
```

CONDICIONALES ELSE if

```
if(condicion){

} else if(condicion){

}
```

CONDICIONALES SWITCH

```
switch(nomVariable) {
case "opcion1" :
    instrucciones;
   break;
case "opcion2" :
    instrucciones;
    break;
.....................
case "opcionN" :
    instrucciones;
    break;
default :
    instrucciones;
}
```

CONDICIONALES SWITCH

```
switch(numero) {
    case 5:
    ...
    break;
    case 8:
     ...
     break;
    case 20:
    ...
    break;
    default:
    ...
    break;
}
```

BUCLE FOR

```
for(var i=0;i<=valor;i++){
    //instrucciones a repetir
}
```

BUCLE WHILE

```
while(condicion a evaluar){

}
```

CONDICIONALES ELSE if

```
do{

} while(condicion)
```

Funciones

¿Qué es una función?

Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

```
function nombre_funcion(argumento1, argumento2, …., argumentoN) {

        ...

        return valor_a_retornar;
}
```

A continuación se presentarán las funciones y propiedades básicas de Javascript

FUNCIONES útiles Números

Error común para números

- **NaN** (del inglés, *"Not a Number"*. Indica un valor numérico no definido. Por ejemplo, la división 0/0)

```
var numero1 = 0;
var numero2 = 0;
alert(numero1/numero2);                    // se muestra el valor NaN
```

Funciones útiles para números

- **isNaN()** (Permite proteger a la aplicación de posibles valores numéricos no definidos)

```
var numero1 = 0;
var numero2 = 0;
if(isNaN(numero1/numero2)) {
        alert("La división no está definida para los números indicados");
}
else {
        alert("La división es igual a => " + numero1/numero2);
}
```

FUNCIONES útiles Números

Funciones útiles para números

- **Infinity** (Hace referencia a un valor numérico infinito y positivo. También existe el valor –Infinity para los infinitos negativos)

```
var numero1 = 10;
var numero2 = 0;
alert(numero1/numero2);              // se muestra el valor Infinity
```

- **toFixed(digitos)** (Devuelve el número original con tantos decimales como los indicados por el parámetro dígitos y realiza los redondeos necesarios. Se trata de una función muy útil por ejemplo para mostrar precios)

```
var numero1 = 4564.34567;
numero1.toFixed(2);       // 4564.35
numero1.toFixed(6);       // 4564.345670
numero1.toFixed();        // 4564
```

FUNCIONES útiles Cadenas

`length`

Calcula la longitud de una cadena de texto (el número de caracteres que la forman)

```
var mensaje = "Hola Mundo";
var numeroLetras = mensaje.length;
```

FUNCIONES útiles Cadenas

`+ o concat`

(Se emplean para concatenar varias cadenas de texto)

```
var mensaje1 = "Hola";
var mensaje2 = " Mundo";

var mensaje = mensaje1 + mensaje2;

var mensaje3 = mensaje1.concat(mensaje2);
```

FUNCIONES útiles Cadenas

toUpperCase()

Transforma el texto en mayúscula

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toUpperCase();
```

toLowerCase()

Transforma el texto en minúscula

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toLowerCase();
```

FUNCIONES útiles Cadenas

charAt()

Obtiene el caracter que en la posición indicada

```
var mensaje = "Hola";
var letra = mensaje.charAt(0); // letra = H
letra = mensaje.charAt(2);     // letra = l
```

substring(inicio, final)

Toma una porción del texto

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(2) //"la Mundo"
porcion = mensaje.substring(1, 8); //"ola Mun"
```

FUNCIONES útiles Cadenas

`indexOf(caracter)`

(Calcula la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter se incluye varias veces dentro de la cadena de texto, se devuelve su primera posición empezando a buscar desde la izquierda. Si la cadena no contiene el carácter, la función devuelve el valor -1)

```
var mensaje = "Hola Carola";
var posicion = mensaje.indexOf('a');
//posicion =3
posicion = mensaje.indexOf('b'); // posicion=-1
```

FUNCIONES útiles Cadenas

`lastIndexOf(caracter)`

(calcula la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si la cadena no contiene el carácter, la función devuelve el valor -1. La posición devuelta se calcula empezando a contar desde el principio de la palabra)

```
var mensaje = "Hola Carola";
var posicion = mensaje.lastIndexOf('a');
//posicion =10
posicion = mensaje.lastIndexOf('b'); //
posicion=-1
```

Arrays

```
var variable1 = new Array();

var variable1 = new Array(10);

var variable1 = new Array(2, "hola",

true, 45.34);
```

Arrays

```
var variable1 = new Array();

variable1[0] = 2;

variable1[1] = "hola";

variable1[2] = true;

variable1[3] = 45.34;
```

Arrays

```
var dias = ["Lunes", "Martes", "Miércoles",

"Jueves", "Viernes", "Sábado", "Domingo"];



for(var i=0; i<7; i++) {

  alert(dias[i]);

}
```

Arrays

```
var dias = ["Lunes", "Martes", "Miércoles",

"Jueves", "Viernes", "Sábado", "Domingo"];



for(i in dias) {

  alert(dias[i]);

}
```

Funciones útiles para arrays

Funciones útiles para arrays

- **length** (Calcula el número de elementos de un array)

```
var vocales = ["a", "e", "i", "o", "u"];
var numeroVocales = vocales.length;            // numeroVocales = 5
```

- **concat()** (Se emplea para concatenar los elementos de varios array)

```
var array1 = [1, 2, 3];
array2 = array1.concat(4, 5, 6);       // array2 = [1, 2, 3, 4, 5, 6] array3 =
array1.concat([4, 5, 6]);   // array3 = [1, 2, 3, 4, 5, 6]
```

- **join(separador)** (Es la función contraria a split(). Une todos los elementos de un array para formar una cadena de texto. Para unir los elementos se utiliza el carácter separador indicado)

```
var array = ["hola", "mundo"];
var mensaje = array.join("");          // mensaje = "holamundo"
mensaje = array.join(" ");         // mensaje = "hola mundo"
```

Funciones útiles para arrays

Funciones útiles para arrays

- **pop() (**Elimina el último elemento del array y lo devuelve. El array original se modifica y su longitud disminuye en 1 elemento)

```
var array = [1, 2, 3];
var ultimo = array.pop();   // ahora array = [1, 2], ultimo = 3
```

- **push()** (Añade un elemento al final del array. El array original se modifica y aumenta su longitud en 1 elemento. También es posible añadir más de un elemento a la vez)

```
var array = [1, 2, 3];
array.push(4);          // ahora array = [1, 2, 3, 4]
```

- **shift()** (Elimina el primer elemento del array y lo devuelve. El array original se ve modificado y su longitud disminuida en 1 elemento)

```
var array = [1, 2, 3];
var primero = array.shift();              // ahora array = [2, 3], primero = 1
```

Funciones útiles para arrays

Funciones útiles para arrays

- **unshift()** (Añade un elemento al principio del array. El array original se modifica y aumenta su longitud en 1 elemento. También es posible añadir más de un elemento a la vez)

```
var array = [1, 2, 3];
array.unshift(0);              // ahora array = [0, 1, 2, 3]
```

- **reverse()** (Modifica un array colocando sus elementos en el orden inverso a su posición original)

```
var array = [1, 2, 3];
array.reverse();               // ahora array = [3, 2, 1]
```

Ámbito de las variables

Variables locales

El ámbito de una variable (llamado "scope" en inglés) es la zona del programa en la que se define la variable. JavaScript define dos ámbitos para las variables: global y local.

```
function creaMensaje() {
        var mensaje = “Mensaje de prueba”;
}
creaMensaje();
alert(mensaje);
```

Al ejecutar el código anterior no se muestra ningún mensaje por pantalla. La razón es que la variable ”mensaje” se ha definido dentro de la función creaMensaje() y por tanto, es una variable local que solamente está definida dentro de la función.

Cualquier instrucción que se encuentre dentro de la función puede hacer uso de esa variable, pero todas las instrucciones que se encuentren en otras funciones o fuera de cualquier función no tendrán definida la variable “mensaje”.

Ámbito de las variables

Variables globales

Está definida en cualquier punto del programa (incluso dentro de cualquier función).

```
var mensaje = “Mensaje de prueba”;

function muestraMensaje() {
        alert(mensaje);
}
```

La variable ”mensaje” se ha definido fuera de cualquier función.

Este tipo de variables automáticamente se transforman en variables globales y están disponibles en cualquier punto del programa (incluso dentro de cualquier función)

Si en el interior de una función, las variables se declaran mediante var se consideran locales y las variables que no se han declarado mediante var, se transforman automáticamente en variables globales.

¿Qué sucede si una función define una variable local con el mismo nombre que una variable global que ya existe? En este caso, las variables locales prevalecen sobre las globales, pero sólo dentro de la función:

Ámbito de las variables

Variables globales

¿Qué sucede si dentro de una función se define una variable global con el mismo nombre que otra variable global que ya existe?
La variable global definida dentro de la función simplemente modifica el valor de la variable global definida anteriormente:

```
var mensaje = "gana la de fuera";

function muestraMensaje() {
        mensaje = "gana la de dentro";
        alert(mensaje);
}

alert(mensaje);         // gana de la fuera
muestraMensaje();  // gana la de adentro
alert(mensaje);// gana la de adentro
```

Javascript - DOM y Eventos

Programación Web I

Comisión Miércoles Noche

DIS. Alicia Rosenthal

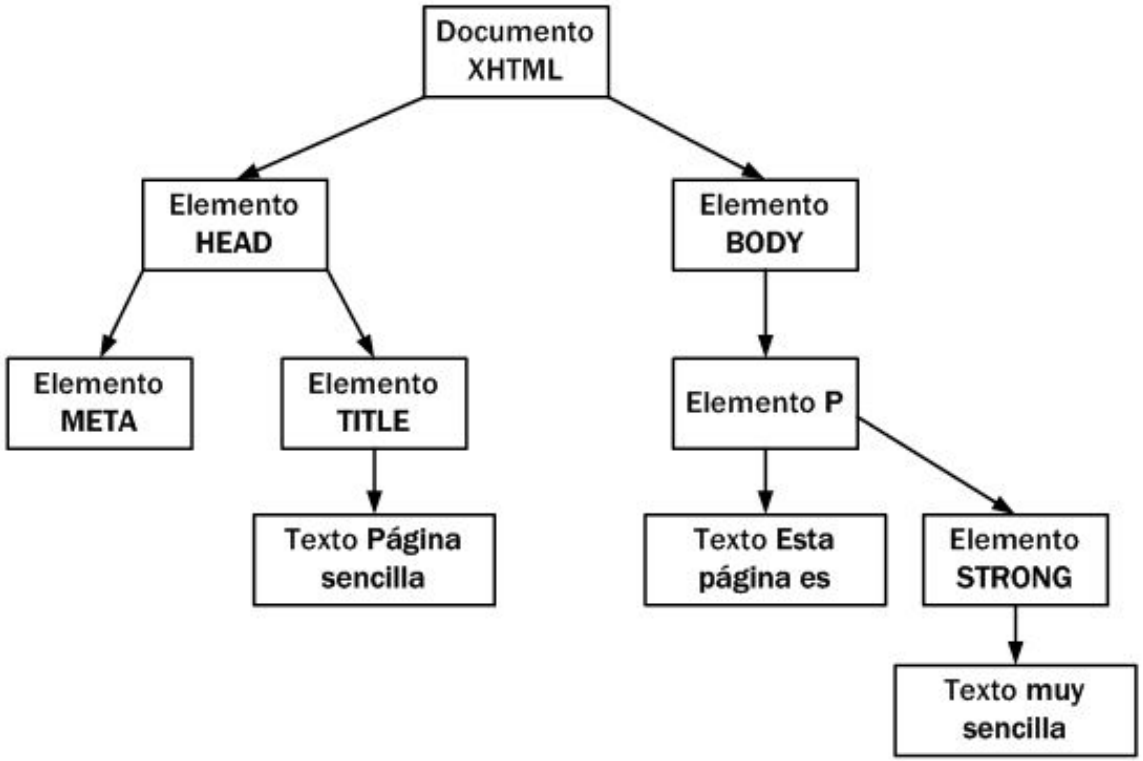
Tec. Willian Dos Reis

Comisión Viernes Mañana

Ing. Gabriel Panik

Document Object Model (DOM)

```
<!DOCTYPE html>
<html>
<head>
     <meta charset=”UTF-8”>
     <title>Página sencilla</title>
</head>
<body>
     <p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```



Tipos de Nodos

DOM define 12 tipos de nodos, aunque las páginas HTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

1. **Document**
   Nodo raíz del que derivan todos los demás nodos del árbol.

2. **Element**
   Representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.

3. **Attr**
   Se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.

4. **Text**
   Nodo que contiene el texto encerrado por una etiqueta HTML.

5. **Comment**
   Representa los comentarios incluidos en la página HTML.

Los otros tipos de nodos existentes que no se van a considerar
son DocumentType, CDataSection, DocumentFragment,Entity, EntityReference, ProcessingInstruction y Notation.

DOM - Tipo de Acceso a Nodos

Tipo de Acceso a los nodos

Una vez construido automáticamente el árbol **completo** de nodos DOM, ya es posible utilizar las funciones DOM para acceder de forma directa a cualquier nodo del árbol. Es decir, su consulta, modificación y su eliminación solamente es posible después de que la página HTML se cargue por completo.

DOM proporciona dos métodos alternativos para acceder a un nodo específico:

1. Acceso a través de sus nodos padre
2. Acceso directo.

Las funciones que proporciona DOM para acceder a un nodo a través de sus nodos padre consisten en acceder al nodo raíz de la página y después a sus nodos hijos y a los nodos hijos de esos hijos y así sucesivamente hasta el último nodo de la rama terminada por el nodo buscado. Sin embargo, cuando se quiere acceder a un nodo específico, es mucho más rápido acceder directamente a ese nodo y no llegar hasta él descendiendo a través de todos sus nodos padre.

DOM - Acceso Directo a los nodos

1) getElementsByTagName(nombreEtiqueta)
La función obtiene todos los elementos de la página HTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

```
var parrafos = document.getElementsByTagName("p");
```

El valor devuelto es un array de nodos DOM, no un array de cadenas de texto o un array de objetos normales

2) getElementsByName()
Se buscan los elementos cuyo atributo “name” sea igual al parámetro proporcionado.

DOM - Acceso Directo a los nodos

Acceso directo a los nodos

3) getElementById()

Devuelve el elemento HTML cuyo atributo id coincide con el parámetro indicado en la función. Como el atributo id debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

```
var nombre = document.getElementById("nombre");
```

Es tan importante y tan utilizada en todas las aplicaciones web, que casi todos los ejemplos y ejercicios que siguen la utilizan constantemente.

DOM - Creación de Nodos

Creación de elementos HTML simples

Por este motivo, crear y añadir a la página un nuevo elemento HTML sencillo consta de cuatro pasos diferentes:

1. Creación de un nodo de tipo Element que represente al elemento.
2. Creación de un nodo de tipo Text que represente el contenido del elemento.
3. Añadir el nodo Text como nodo hijo del nodo Element.
4. Añadir el nodo Element a la página, en forma de nodo hijo del nodo correspondiente al lugar en el que se quiere insertar el elemento.

El proceso de creación de nuevos nodos puede llegar a ser tedioso, ya que implica la utilización de tres funciones DOM:

1. createElement(etiqueta): crea un nodo de tipo Element que representa al elemento HTML cuya etiqueta se pasa como parámetro.
2. createTextNode(contenido): crea un nodo de tipo Text que almacena el contenido textual de los elementos HTML.
3. nodoPadre.appendChild(nodoHijo): añade un nodo como hijo de otro nodo. Se debe utilizar al menos dos veces con los nodos habituales: en primer lugar se añade el nodo Text como hijo del nodo Element y a continuación se añade el nodo Element como hijo de algún nodo de la página.

DOM - Creación y Eliminación de Nodos

```
// Crear nodo de tipo Element
var parrafo = document.createElement("p");

// Crear nodo de tipo Text
var contenido = document.createTextNode("Hola Mundo!");

// Añadir el nodo Text como hijo del nodo Element
parrafo.appendChild(contenido);

// Añadir el nodo Element como hijo de la pagina
document.body.appendChild(parrafo);
```

DOM - Creación y Eliminación de Nodos

Creación y Eliminación de nodos

Eliminación

Solamente es necesario utilizar la función removeChild().

```
var parrafo = document.getElementById("provisional");
parrafo.parentNode.removeChild(parrafo);


<p id="provisional">...</p>
```

La función removeChild() requiere como parámetro el nodo que se va a eliminar. Además, esta función debe ser invocada desde el elemento padre de ese nodo que se quiere eliminar.

Así, para eliminar un nodo de una página HTML se invoca a la función removeChild() desde el valor parentNode del nodo que se quiere eliminar. Cuando se elimina un nodo, también se eliminan automáticamente todos los nodos hijos que tenga, por lo que no es necesario borrar manualmente cada nodo hijo.

DOM - Creación y Eliminación de Nodos

Acceso directo a los atributos HTML

Mediante DOM, es posible acceder de forma sencilla a todos los atributos HTML y todas las propiedades CSS de cualquier elemento de la página.

Los atributos HTML de los elementos de la página se transforman automáticamente en propiedades de los nodos. Para acceder a su valor, simplemente se indica el nombre del atributo HTML detrás del nombre del nodo.

```
var enlace = document.getElementById("enlace");

alert(enlace.href); // muestra http://www...com

<a id="enlace" href="http://www...com">Enlace</a>
```

Las propiedades CSS no son tan fáciles de obtener como los atributos HTML. Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo style.

```
var imagen = document.getElementById("imagen");

alert(imagen.style.margin);

<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

DOM - Acceso directo a los atributos HTML

Acceso directo a los atributos HTML

La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guión medio.

- font-weight se transforma en fontWeight
- line-height se transforma en lineHeight
- border-top-style se transforma en borderTopStyle
- list-style-image se transforma en listStyleImage

El único atributo HTML que no tiene el mismo nombre en HTML y en las propiedades DOM es el atributo “class”. Como la palabra class está reservada por JavaScript, no es posible utilizarla para acceder al atributo class del elemento HTML. En su lugar, DOM utiliza el nombre className para acceder al atributo class de HTML

EVENTOS de Teclado

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
onkeyup	dejar de teclear	TODOS
onkeypress	presionar una tecla	TODOS
onkeydown	presionar una tecla sin soltar	TODOS

ACCEDER A UN ELEMENTO

```
<!--HTML-->
<h1 id="titulo">

//javascript
document.getElementById("titulo")
```

Value

```
<!--HTML-->
<input type="text" id="nombre"
name="nombre">

//javascript
var nombre=
document.getElementById("nombre").value;
```

INNERHTML

```
//javascript
document.getElementById("mensaje")
.innerHTML="texto";
```

EVENTOS de FORMULARIOS

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
onfocus	seleccionar un elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
onblur	deseleccionar un elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
onsubmit	enviar un formulario	`<form>`
onreset	al resetear un formulario	`<form>`
onchange	deseleccionar un elemento que fue modificado	`<input>`, `<select>`, `<textarea>`

PROPIEDAD DISABLED

```
//javascript
document.getElementById("cantidad")
.disabled=true;
```

Modificar Estilos

```
//javascript
  document.getElementById("titulo")
  .style.propiedadCss= "valor";

  document.getElementById("titulo")
  .style.display= "none";
```

EVENTOS DE PÁGINA

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
onload	Al cargar la página	<body>
onunload	Al abandonar la página	<body>
onresize	al achicar o agrandar la ventana del navegador	<body>

Windows.onload

```
window.onload = function(){

}
```

Eventos de Mouse

EVENTO	DESCRIPCIÓN	ELEMENTOS QUE PUEDEN TENER ESTE EVENTO
onclick	al hacer click	TODOS
ondblclick	al hacer doble click	TODOS
onmouseover	al pasar el mouse	TODOS
onmousedown	mientras tengo presionado el mouse	TODOS
onmouseup	cuando suelto el mouse	TODOS
onmousemove	cuando muevo el mouse	TODOS

Eventos de Mouse

```
function mostrarMensaje() {
 alert("Este es un evento");
}

<button
onclick="muestraMensaje()">
Hacé click</button>
```

Expresiones Regulares

```
expresionRegular.test(valorAEvaluar);
```

```
(regex).test(cadenaAEvaluar)
```

Local Storage

```
Guardar:
localStorage.setItem
('nombreVariable',valor);

Obtener:
localStorage.getItem
('nombreVariable');
```

Local Storage

Borrar:

```
localStorage.clearItem
('nombreVariable');
```

Intervalos

Intervalo, ejecutar unas instrucciones cada x cantidad de tiempo.

```
setInterval(función,tiempo en
milisegundos);

setInterval(function(){
   funcion();
},1000);
```

Frenar un intervalo

```
clearInterval(nombreIntervalo);
```

Tiempo definido

```
setTimeout(function(){
  funcion()
},1000);
```

addEventListener

Le agrega un manejador de eventos a un elemento. Es un escuchador.

Sintaxis:

```
element.addEventListener(event, function);
```

Ejemplo:

```
var boton= document.getElementById("boton");

boton.addEventListener("click", function(){
   ...acciones
});
```

Obtener el valor de una tecla

```
document.addEventListener
('keydown',function(e){
    console.log(e);
});

e.key;
//muestra la tecla pulsada
e.keyCode;
//muestra el codigo de tecla
```

Eventos del mouse

```
document.addEventListener
('mousemove',function(e){
      var x=e.screenX;
      var y=e.screenY;
});
```

screenY muestra la posición y
screenX muestra la posición x

Eventos del mouse

```
document.addEventListener
('mousedown',function(e){
      console.log('Mouse Apretado');
});

document.addEventListener
('mouseup',function(e){
      console.log('Suelto mouse');
});
```

Eventos del mouse

```
document.addEventListener
('mousedown',function(e){
    console.log('Mouse Apretado');
});

//e.which muestra el botón del
mouse que fue apretado
```

Eventos del mouse

```
document.addEventListener
('mousewheel',function(e){
  console.log(e.deltaY);
});
```

deltaY muestra valores positivos cuando la ruedita del mouse gira hacia arriba y negativos hacia abajo.

Programación Web I

Comisión Miércoles Noche

DIS. Alicia Rosenthal

Tec. Willian Dos Reis

Comisión Viernes Mañana

Ing. Gabriel Panik

Agenda

1. Introducción
2. ¿Qué es jQuery?
3. Conceptos Básicos
4. Conceptos Fundamentales

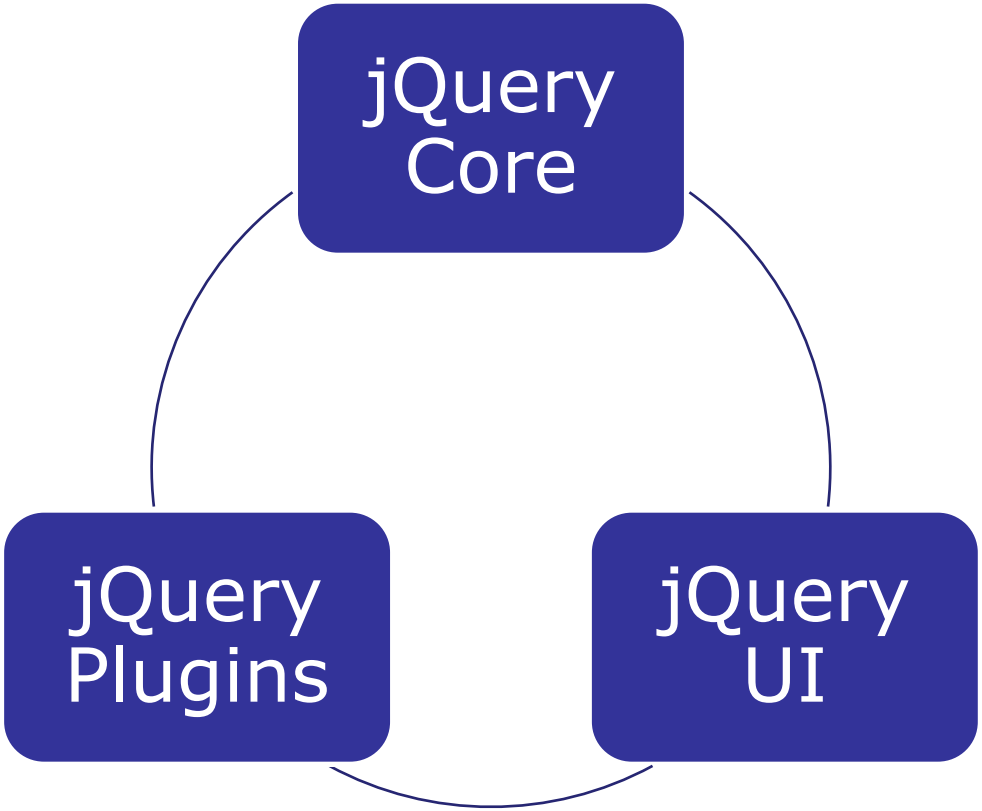
¿Qué es jQuery?

Es un biblioteca de javascript rápida y concisa que permite simplificar el desarrollo de la parte del cliente. Es open source y fue diseñado para cambiar la manera de escribir código Javascript.

jQuery brinda las siguientes funcionalidades:

- Navegación DOM (usando expresiones CSS o Xpath)
  - Acceder a elementos de una Página web
  - Modificar la apariencia de una Página web
  - Alterar el contenido de una Página web
  - Agregar animaciones a una Página web
- Capturar Eventos
- Mejorar la usabilidad de una Página web
- Soporta extensiones
- Interacción con AJAX
- Compatible con CSS3
- Compatible con la mayoría de los navegadores

¿Qué es jQuery?



Conceptos Básicos

¿Cómo empiezo a usar jQuery Core?

Es necesario descargar la biblioteca jquery.js (http://jquery.com/)

Existen 2 niveles de compresión:
- Desarrollo (mínima compresión, permite un mejor debug)
- Producción (máxima compresión, para el servidor destino)

Incluirla en la página web

```
<script type="text/javascript" src="jquery-1.10.2.js"></script>
```

Conceptos Fundamentales

Windows Onload

Anteriormente en las páginas web (quizás algunas aún lo usan) era necesario ejecutar código Javascript inmediatamente después de la carga del contenido de la página.

En Javascript "puro" se utiliza la siguiente instrucción:

```
window.onload = function() {……..}
```

En jQuery tenemos la siguiente instrucción:

```
$(window).load(function() {……..});
```

Ambas funciones **no** se ejecutan hasta que se cargue TODO el contenido…

Es un costo de tiempo que muchas páginas no desean asumir…

Conceptos Fundamentales

Document Ready

En las aplicaciones web es necesario esperar que el árbol DOM sea construido por completo antes de comenzar a realizar modificaciones o seleccionar algún elemento.

jQuery posee la siguiente instrucción:

```
$(document).ready( function() { ... } );

o

$(function() { … });
```

La función se ejecuta cuando el DOM se encuentra disponible, aunque el contenido no se haya cargado aún

Conceptos Fundamentales

Acceso al DOM

En Javascript "puro" se utilizan las siguientes instrucciones:

```
getElementById()              // Acceso a un elemento por el ID
getElementsByTagName()   // Acceso a elementos por Tag
•getElementsByName()        // Acceso a elementos por el Nombre
getElementsByClassName() // Acceso a elementos por su clase
          (función disponible en Internet)
```

jQuery incluye la función **$()** como una alternativa más rápida y completa para seleccionar elementos de la página.
Para ello, se utilizan **selectores**. Alguno de ellos son:

```
$("#txtCliente");   // Seleccionar por el id "txtCliente"
$("div");               // Seleccionar todos los div (tag)
$(".resaltado");     // Seleccionar mediante una clase CSS
```

También puede utilizarse la función jQuery() en reemplazo de $()

Conceptos Fundamentales

Selectores

jQuery utiliza lo mejor de las distintas versiones de CSS (1, 2 y 3) y XPath para seleccionar de forma sencilla cualquier elemento de la Página web.

Los selectores devuelven un objeto jQuery (NO un nodo DOM).

Si se quieren utilizar caracteres como (!"#$%&'()*+,./:;<=>?@[\]^`{|}~) como parte del nombre de un elemento HTML, se deben anteponer dos contrabarras "\\" antes del caracter utilizado.

Ej: id="principal.titulo"

```
$("#principal\\.titulo")
```

Conceptos Fundamentales

Referencia API

Core	Selectores
Atributos	Transversing
Manipulación	CSS
Eventos	Efectos
Ajax	Utilidades

Conceptos Fundamentales

Selectores Básicos

jQuery('*')
Selector universal, selecciona TODOS los elementos.

jQuery('#id')
Selecciona un elemento mediante el atributo "id"

jQuery('elemento')
Selecciona los elementos del tipo especificado

jQuery('.clase')
Selecciona los elementos que poseen la clase especificada. Utiliza la función nativa ("getElementsByClassName").

jQuery('selector1, selector2, selectorN')
Selecciona los elementos que cumplan con todos los selectors especificados.

Conceptos Fundamentales

Selectores de Atributos

jQuery('[atributo="valor"]')
Selecciona los elementos que poseen un atributo con el valor especificado.

jQuery('[atributo!="valor"]')
Selecciona los elementos que poseen un atributo con un valor distinto al especificado.

jQuery('[atributo^="valor"]')
Selecciona los elementos que poseen un atributo con un valor que comience con el valor especificado.

jQuery('[atributo|="valor"]')
Selecciona los elementos que poseen un atributo con el valor especificado, o que comienza con el string seguido de "-".

Conceptos Fundamentales

Selectores de Atributos

jQuery('[atributo*="valor"]')
Selecciona los elementos que poseen un atributo cuyo valor contiene el valor especificado.

jQuery('[atributo~="valor"]')
Selecciona los elementos que poseen un atributo cuyo valor contiene el valor especificado, delimitado por espacios.

jQuery('[atributo$="valor"]')
Selecciona los elementos que poseen un atributo cuyo valor termina con el valor especificado (case sensitive).

jQuery('[filtroAtributo1][filtroAtributo2][filtroAtributoN]')
Selecciona los elementos que cumplan con todos los filtros de atributos especificados.

Conceptos Fundamentales

Selectores de Formulario

jQuery(':elemento')
Selecciona los elementos del tipo especificado (:input, :button, :checkbox, :submit, :text, :image, :hidden, :file, :password, etc.)

Selectores de Filtros de Formulario

jQuery(':estado')
Selecciona los elementos que poseen el estado especificado (:disabled, :enabled, :checked y :selected)

Selectores de Filtros de Visibilidad

jQuery(':visibilidad')
Selecciona los elementos que poseen la visibilidad especificada (:hidden y :visible)

Conceptos Fundamentales

Selectores de Filtros Básicos

jQuery('elemento:first') (devuelve el primer elemento de la lista)
jQuery('elemento:last') (devuelve el último elemento de la lista)
jQuery('elemento:even') (devuelve índices pares, incluyendo el cero)
jQuery('elemento:odd') (devuelve índices impares)
jQuery(':header') (devuelve los elementos de tipo header -h1, h2, etc.-)
jQuery(':animated') (devuelve los elementos que tienen animación)
jQuery(':not(selector)') (devuelve los elementos que cumplen la condición)

Selectores de Filtros Básicos (con Índice)

Selecciona elementos de una lista devuelta por un selector previo.

jQuery('elemento:eq(índice)') (devuelve el índice indicado)
jQuery('elemento:lt(índice)') (devuelve los índices menores al número indicado)
jQuery('elemento:gt(índice)') (devuelve los índices mayores al número indicado)
jQuery('elemento.eq(índice)') (devuelve el índice indicado, incluyendo negativos)

Conceptos Fundamentales

Selectores de Contenido

jQuery('elemento:contains(texto)') (devuelve los elementos que cumplan con el texto indicado)
jQuery('elemento:empty') (devuelve los elementos que no tienen hijos)
jQuery('elemento:has(selector)') (devuelve los elementos que al menos cumplan una vez con lo especificado)
jQuery('elemento:parent') (devuelve elementos padres que tienen hijos, incluyendo texto)

Selectores de Filtros Hijos

jQuery('elemento:first-child') (devuelve el primer elemento de cada elemento padre)
jQuery('elemento:last-child') (devuelve el último elemento de cada elemento padre)
jQuery('elemento:nth-child(index/even/odd/equation)') (devuelve lo especificado de cada elemento padre)
jQuery('elemento:only-child') (devuelve el elemento siempre que tenga u único hijo)

Conceptos Fundamentales

Selectores de Jerarquía

Child

jQuery(“selector padre > selector hijo”) (devuelve los elementos que cumplan con el filtro especificado)

jQuery(“selector ancestro > selector descendiente”) (devuelve los elementos especificados como “descendientes” de los especificados como “ancestros”)

jQuery('selector previo + selector siguiente') (devuelve los elementos especificados como “siguientes” a los especificados como “previos”)

jQuery('selector previo ~ selector siblings') (devuelve los elementos hijos que cumplen con todos los elementos del selector previo)

Conceptos Fundamentales

Atributos

.prop("propiedad"[,"valor"]) (devuelve/setea el valor de la propiedad seleccionada)
Devuelve *undefined* si la propiedad no fue seteada.

.prop("nombre del atributo"[,"valor"]) (devuelve/setea el valor del atributo seleccionado)
Devuelve *undefined* si el atributo no fue seteado.

Ejemplos:
$(elem).prop("checked") devuelve true

$(div).attr("checked") devuelve "checked" (versión 1.6 o superior)

Aclaración:

$(div).attr("checked") devuelve true (versiones inferiores a 1.6)

.addClass("nombre de la clase") (agrega una clase a un elemento)

Conceptos Fundamentales

Atributos

.removeProp(“propiedad” , “valor”) (elimina propiedades de los elementos seleccionados).

.removeAttr(“propiedad” , “valor”) (elimina atributos de los elementos seleccionados).

.removeClass(clase1, clase2, .., claseN) (elimina las clases especificadas).

.hasClass(nombre de la clase) (elimina las clases especificadas).

.html() (muestra el código HTML del elemento seleccionado).

.val(valor) (devuelve/setea el valor (o valores) del primer elemento seleccionado).

Conceptos Fundamentales

Transversing

.add(elemento/selector/html) (agrega elementos o html o selectores)
Devuelve *undefined* si la propiedad no fue seteada.

.children(selector) (devuelve los hijos de cada elemento seleccionado)

.each(selector) (itera sobre el objeto jQuery devuelto por el selector)

.eq(indice) (selecciona el primer elemento de una lista)

.first() (selecciona el primer elemento de una lista)

.last() (selecciona el último elemento de una lista)

Otras funciones en: http://api.jquery.com/category/traversing/

Conceptos Fundamentales

Manipulación

.append() (inserta el contenido especificado en el parametro al final de cada elemento obtenido por el selector)

.appendTo() (inserta el contenido especificado en el parametro al final de cada elemento obtenido por el selector)

.clone() (clona los elementos obtenidos en el selector)

.css(propiedad [, valor]) (devuelve / setea el valor de una propiedad CSS)

.detach() (elimina del DOM los elementos devueltos por el selector)

.empty() (elimina del DOM todos los hijos de los elementos devueltos por el selector)

.insertAfter() (inserta cada elemento devuelto por el selector luego del nodo del arbol del DOM seleccionado)

.insertBefore() (inserta cada elemento devuelto por el selector antes del nodo del arbol del DOM seleccionado)

Conceptos Fundamentales

Manipulación

.text() (devuelve el contenido de los elementos devueltos por el selector, incluyendo sus hijos)

.wrap() (inserta el contenido HTML especificado alrededor de cada elemento obtenido por el selector)

.wrapAll() (inserta el contenido HTML especificado alrededor de todo el grupo de elementos obtenidos por el selector)

Otras funciones en: http://api.jquery.com/category/manipulation/

Eventos

¿Qué son? ¿Cómo se aplican?

- Son eventos tal como vimos con JavaScript, responden a cierta acción realizada sobre uno o más elementos seleccionados por Jquery.

- Para aplicarlo se debe realizar lo siguiente,

```
$( “selector" ).evento(funcion_anonima);
```

- Por Ejemplo,

```
$( "#target" ).click(function() {
      alert( “Se realize click sobre alguno de los elementos." );
 });
```

Eventos - Listado

.blur()

- El evento se activa al desenfocar el/los elementos seleccionados.

.change()

- El evento se activa al cambiar un elemento

.click()

- El evento se activa al hacer click con el mouse y soltar

.contextmenu()

- El evento se activa al mostrar el menú contextual.

.dblclick()

- El evento se activa al hacer doble click con el mouse

.dblclick()

- El evento se activa al hacer doble click con el mouse

.focus() – Ver también .focusin() y .focusout()

- El evento se activa al hacer foco sobre un elemento seleccionado

Eventos - Listado

.hover()

- El evento se activa cuando el puntero del mouse entra o abandona un elemento

jQuery.ready() - IMPORTANTE

- Notifica que el documento está listo y cargado

.keydown(), .keypress() y .keyup()

- Eventos equivalentes a los de Javascript

.mousedown(), .mousemove(), .mouseout(), .mouseover() y .mouseup()

- Eventos equivalentes a los de Javascript

.mouseenter()

- El evento se activa cuando el puntero del mouse entra en un elemento

.mouseleave()

- El evento se activa cuando el puntero del mouse abandona un elemento

Eventos - Listado

.off()

- Elimina un manejador de eventos (event handler) de los elementos seleccionados

.on()

- Agrega un manejador de eventos (event handler) a los elementos seleccionados

.one()

- Agrega un event handler a los elementos seleccionados. Este handler es ejecutado como máximo una vez por elemento por tipo de evento

.ready()

- Especifica una función a ejecutar una vez que el DOM esté cargado por completo

.resize(), .scroll(), .select() y .submit()

- Eventos equivalentes a los de Javascript

.trigger()

- Activa los eventos vinculados a los elementos seleccionados

Eventos - JQuery Mobile

¿Cómo se aplican?

- Para aplicarlo se debe realizar lo siguiente,

```
jQuery( "selector" ).on("evento", function( event ) { ... } )
```

- Por Ejemplo,

```
$( "#target" ).on(“swipe", function() {
      alert( “Se realizó un swipe." );
 });
```

Eventos - JQuery Mobile

mobileinit

- El evento se activa cuando JQuery Mobile finaliza de cargar

scrollstart

- El evento se activa cuando comienza el scroll

scrollstop

- El evento se activa cuando se detiene el scroll

swipe

- El evento se activa cuando se arrastra “el dedo” por al menos 30px en un período de 1 segundo

swipeleft / swiperight

- El evento se activa al hacer el swipe hacia la izquierda / derecha

tap

- El evento se activa luego de hacer un toque en la pantalla

taphold

- El evento se activa luego de hacer un toque prolongado en la pantalla