

# JAVASCRIPT

JavaScript es un lenguaje de programación creado por Netscape que nos permite crear páginas web interactivas, es decir, páginas dinámicas en las que los usuarios pueden participar y aportar. Este es un lenguaje interpretado que no necesita de un servidor para analizar todo el código escrito pues está presente en todos los navegadores y son estos los encargados de interpretar los scripts (códigos). Básicamente, JavaScript nos permite crear eventos en una página web y responder a estos. Por ejemplo, cuando presionamos un botón JavaScript identifica un evento, que es la acción que queremos realizar, ante esto nos entrega una respuesta/solución desencadenando otro evento o tarea. Entre los tipos de lenguajes de programación y sus características JavaScript se define como un lenguaje de programación orientado a objetos, imperativo, débilmente tipado e interpretado. Aunque esto puede ser muy extraño inicialmente, iremos comprendiendo a lo largo de esta unidad qué significa cada uno de estos términos.

JavaScript fue creado en sus inicios para extender un poco más la funcionalidad de un simple documento HTML, el cual se limita a presentar un contenido estático sin ningún tipo de interacción con el usuario, por ejemplo, emitir una alerta o mensaje de aviso y abrir una nueva ventana. En la actualidad JavaScript es uno de los lenguajes más populares entre los amantes de la tecnología, quienes lo prefieren por su nivel de flexibilidad y dinamismo, la especialidad del lenguaje, la interacción con el DOM y la programación de páginas web dinámicas.

- El Navegador // navigator
- El DOM // document
- La ventana del navegador // window

## Historia

JavaScript fue creado por Brenda Eich de Netscape quien inicialmente lo llamó 'Mocha', más tarde recibió el nombre de LiveScript para finalmente consolidarse como JavaScript.

Actualmente JS es una marca registrada de Oracle y fue llamada JavaScript debido a que JAVA, un lenguaje de programación muy popular de la misma compañía, estaba en una época bastante brillante, Oracle decidió entonces llamarlo JavaScript como una estrategia para venderlo mejor.

Cuando JS estaba en una buena posición en el mercado, Microsoft decidió sacar su propia versión y lo llamó JScript, el cual fue lanzado en la versión 3.0 de Internet Explorer. Aunque era una especificación muy parecida a JavaScript, este era totalmente incompatible.

En 1997 los autores propusieron a JavaScript para que fuera considerado como un estándar ante la European Computer Manufacturers Association (ECMA) y poco después un estandar ISO.

En la actualidad existen miles de tecnologías y herramientas basadas en JavaScript, las cuales facilitan tareas y ofrecen muchas soluciones prácticas al día a día de un desarrollador.

Al retomar un poco la historia y los estándares que definen a JavaScript, es necesario hablar sobre ECMAScript, lenguaje de programación publicado por ECMA International, que es el que define el comportamiento, las funcionalidades, las formas de trabajar y todo lo relacionado con el lenguaje. ECMAScript está basado en JavaScript (estándar propuesto por la compañía Netscape). Podemos ver un poco más de información en su sitio oficial:

<http://www.ecma-international.org/publications/standards/Ecma-262.htm>



## Historia

ECMAScript fue creado por Brenda Eich, un empleado de Netscape, y tuvo varios nombres como Mocha y LiveScript antes de conocerse como JavaScript. Fue anunciado oficialmente en 1996 con una especificación llamada ECMA-262.

A lo largo de la historia se han publicado seis versiones oficiales, pero aún existe un alto grado de incompatibilidad entre los navegadores y la versión ECMAScript 6 (o ECMAScript 2015 porque fue liberada en junio de ese año). Debido a esto, la versión 5 es la más utilizada en la actualidad por ser precisamente la más compatible.

El siguiente es el listado de las versiones oficialmente liberadas por ECMA International (Tomado de: <https://es.wikipedia.org/wiki/ECMAScript>)



## Espacios en blanco

Los caracteres de espacio en blanco tienen un comportamiento especial. Aunque no interfieren con el funcionamiento de los scripts es necesario tener en cuenta que los espacios en JavaScript son ignorados, sean pocos o muchos el intérprete no tendrá en cuenta los espacios.

Por lo tanto: `var a=2`

es lo mismo que tener: `var a=2`

## Diferencia entre minúsculas y mayúsculas

JavaScript es un lenguaje sensible a las minúsculas y mayúsculas, por lo tanto no es lo mismo tener una cadena de texto escrita en mayúscula que una en minúscula.

Por ejemplo: `var x=6`

No es lo mismo que: `var X=6`

El punto y coma ( ; )

La mayoría de lenguajes de programación utilizan el simbolo de punto y coma para finalizar cada instrucción o línea de código, en JavaScript esto no es obligatorio; al ser un lenguaje tan dinámico esta es una de sus características importantes, la única situación en donde es necesario el punto y coma es cuando queremos ponervarias expresiones en la misma línea:

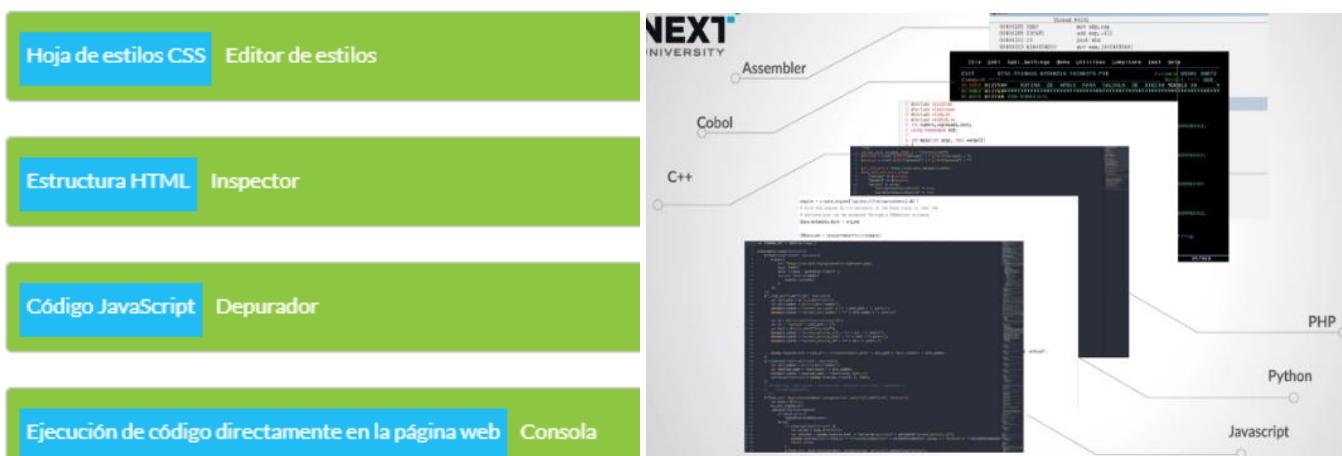
Ejemplo:

Sin punto y coma:

```
var a,b,c  
a=3  
b=4  
c=a+b
```

Instrucciones en una sola línea añadiendo el punto y coma:

```
var a,b,c;  
a=3; b=4; c=a + b;
```



Algoritmo Conjunto finito de pasos lógicos y ordenados para darle solución a un problema determinado.

Ejemplos de algoritmo Instrucciones para armar una silla, una receta de cocina y una lista de compras.

Un algoritmo debe ser

Finito, preciso, estar compuestos por una entrada, un proceso y una salida y estar definidos.

1 Debe ser **preciso**.

2 Debe estar **definido**.

3 Debe ser **finito**.

4 Debe estar compuesto por:  
**entrada, proceso y salida**.

PHP, JavaScript, C++ Son algunos de los lenguajes que sirven para ejecutar algoritmos en una computadora.

Un lenguaje **interpretado**, es un lenguaje de programación en el que la mayoría de sus implementaciones ejecutan las **instrucciones** directamente, sin la necesidad de **compilar** un programa de **instrucciones de código maquina**. Por ejemplo: **JavaScript**.

Un lenguaje **compilado**, es un lenguaje de programación en el cual su **código fuente** pasa por un proceso de **traducción** a **código máquina** antes de ser ejecutado por la máquina. Por ejemplo: **Java**.

## Tipos de datos

```
"numero.toString()"  
"numero.toFixed()"  
"numero.toPrecision()"
```

## Math

```
"Math.min(0, 150, 30, 20, -8, -200);"  
"Math.random();"  
"Math.round(4.7);"
```

## Fechas

¿Qué puedo hacer para fijar una fecha entre el año 0 y 99?

```
1 var fecha=new Date(99,5,24,11,33,30,0);  
2 undefined  
3 fecha.setFullYear(99);  
4 -59027901998000  
5 fecha  
6 Wed Jun 24 99 11:33:30 GMT-0500 (COT)
```

El valor del mes es un número entero entre 0 y 11, donde 0 es enero y 11 corresponde a diciembre.

El día recibe un valor entre 1 y 31.

La hora recibe un valor entre 0 y 23, los minutos y segundos un valor entre 0 y 59 y milisegundos un valor entre 0 y 999.

## Enteros

17, 30, 400, 7896

```
540e+200*1.540e+200  
Infinity
```

## Decimales

1.456, 13.65, 234.7

```
> 2/0  
< Infinity
```

```
new Date();  
Wed Aug 10 2016 15:01:51 GMT-0500 (COT)
```

```
typeof "Juan" // Devuelve string  
typeof 3.14 // Devuelve number  
typeof NaN // Devuelve number  
typeof false // Devuelve boolean  
typeof [1,2,3,4] // Devuelve object  
typeof {nombre:'John', edad:34} // Devuelve object  
typeof new Date() // Devuelve object  
typeof function () {} // Devuelve function  
typeof miCarro // Devuelve undefined  
typeof null // Devuelve object
```

## Ejemplo fechas:

Tienes una aplicación que guarda eventos históricos importantes y necesitas guardar la fecha del gran incendio de Roma ocurrido en el verano del año 64 d.C. durante el reinado de Nerón. La fecha aproximada de este incendio data entre el 18 o 19 de julio del año 64 d.C.

```
var fecha = new Date(64, 7, 19);  
fecha.setFullYear(64);  
var dias_semana = ["Domingo", "Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "S\u00e1bado"];  
console.log("A\u00f1o: " + fecha.getUTCFullYear());  
console.log("Mes: " + fecha.getMonth());  
console.log("D\u00eda: " + fecha.getUTCDate());  
console.log("D\u00eda de la semana: " + dias_semana[fecha.getDay()]);
```

## Matrices

Posici\u00f3n columna			
	[0]	[1]	[2]
1 matriz[2][0]	"Carlos"	12	40
2 "Andres"	"Camil\u00f3n"	45	73
	[2]	"Andres"	67
	[3]	"Pablo"	78

## Objetos

```
var persona = {  
    nombre: '',  
    edad: 0,  
    peso: 0,  
    estatura: 0,  
    IMC: function calIMC (pes, est) {  
        var imc = pes / (est ^ 2);  
        return imc;  
    }  
}
```

## Conversiones

```
x.toFixed(digitos_decimales);  
x.toFixed(0);  
"378"  
x.toFixed(3);  
"378.458"  
x.toFixed(5);  
"378.45800"
```

"67"-78;  
-11  
"67"\*78;  
5226  
"5""\*6";  
30

NaN  
true  
0  
"20,50,40"  
"67"+78;  
"6778"  
true  
0  
"20,50,40" var numeroDesdeCadena= Number("89.543");

## Operadores Condicionales

Comparación	Operador
Igual a	<code>==</code>
Igual valor e igual tipo	<code>===</code>
Diferente	<code>!=</code>
Diferente valor y diferente tipo	<code>!==</code>
Mayor que	<code>&gt;</code>
Menor que	<code>&lt;</code>
Mayor o igual que	<code>&gt;=</code>
Menor o igual que	<code>&lt;=</code>

```
(function(){
    this.variable1 = "Hola Mundo";
})();
console.log(window.variable1);
var myObject = {
    property: "Hola Mundo",
    func: function(){
        return this.property;
    }
};
console.log(myObject.func());
```

### uso del this.

## Secuencias - ciclos



```
do {
    mensaje = mensaje + deportes[1] + "\n";
    l++;
} while (l < 6)
alert(mensaje);
```

```
for (var l = 0; l < l++) {
    mensaje = mensaje + deportes[1] + "\n";
}
alert(mensaje);
```

## Objeto event

```
function resalta(elEvento) {
    var evento = elEvento || window.event;
    switch(evento.type) {
        case 'mouseover':
            this.style.borderColor = 'black';
            break;
        case 'mouseout':
            this.style.borderColor = 'silver';
            break;
    }
}
```

- Evento keydown:
  - Mismo comportamiento en todos los navegadores:
    - Propiedad keyCode: código interno de la tecla
    - Propiedad charCode: no definido
- Evento keypress:
  - Internet Explorer:
    - Propiedad keyCode: el código del carácter de la tecla que se ha pulsado
    - Propiedad charCode: no definido
  - Resto de navegadores:
    - Propiedad keyCode: para las teclas normales, no definido. Para las teclas especiales, el código interno de la tecla.
    - Propiedad charCode: para las teclas normales, el código del carácter de la tecla que se ha pulsado. Para las teclas especiales, 0.
- Evento keyup:
  - Mismo comportamiento en todos los navegadores:
    - Propiedad keyCode: código interno de la tecla
    - Propiedad charCode: no definido

Tipo	Nombre	Descripción aprenderaprogramar.com
Propiedades de control del evento	type	Devuelve el tipo de evento producido, sin el prefijo on (p.ej. click)
	target	Devuelve el elemento del DOM que disparó el evento (inicialmente)
	currentTarget	Devuelve el elemento del DOM que está disparando el evento actualmente (no necesariamente el elemento que disparó el evento, ya que puede ser un disparo debido a burbujeo)
Otras propiedades de control del evento	eventPhase (indica en qué fase de tratamiento de evento estamos, 1 captura, 2 en objetivo, 3 burbujeo), bubbles (booleana, indica si es un evento que burbujea o no), cancelable (booleana, devuelve si el evento viene seguido de una acción predeterminada que puede ser cancelada), cancelBubble (booleana, devuelve si el evento actual se propagará hacia arriba en la jerarquía del DOM o no).	
Propiedad temporal	timeStamp	Devuelve una medida de tiempo en milisegundos desde un origen temporal determinado.
Propiedades de localización del puntero del ratón	clientX, clientY	Devuelven las coordenadas en que se encontraba el puntero del ratón cuando se disparó el evento. Las coordenadas están referidas a la esquina superior izquierda de la ventana del navegador y se expresan en pixeles.
	screenX, screenY	Devuelven las coordenadas en que se encontraba el puntero del ratón cuando se disparó el evento. Las coordenadas están referidas a la esquina superior izquierda de la pantalla y se expresan en pixeles.
	pageX, pageY	Devuelven las coordenadas en que se encontraba el puntero del ratón cuando se disparó el evento. Las coordenadas están referidas a la esquina superior izquierda del documento, que pueden ser distintas a las de la ventana si el usuario ha hecho scroll sobre el documento.
Propiedad para detectar el botón del ratón pulsado	button	Normalmente empleado para el evento mouseup (liberación de botón del ratón) para detectar cuál ha sido el botón pulsado. Contiene un valor numérico: 0 para click normal (botón izquierdo), 1 para botón central (botón en el scroll), 2 para botón auxiliar (botón derecho).
Propiedades relacionadas con el teclado	Para determinar qué tecla ha sido pulsada	Lo estudiaremos por separado en la siguiente entrega del curso
Propiedades relacionadas con drag and drop	Algunas no estandarizadas	dataTransfer, dropEffect, effectAllowed, files, types
Otras propiedades varias	Otras (algunas no estandarizadas)	x, y, layerx, layery, offsetX, offsetY, wheelDelta, detail, relatedNode, relatedTarget, view, attrChange, attrName, newValue, prevValue, data, lastEventId, origin, source
Método	stopPropagation()	Detiene la propagación del evento
Método	preventDefault()	Cancela (si es posible) la acción de defecto que debería ocurrir después del evento (equivale a return false para cancelar la acción).
Otros métodos	initEvent(a, b, c)	se usa para definir eventos. No lo estudiaremos.

## Escuchar eventos:

Añadir un evento que escuche cuando el usuario presiona una tecla en el campo de texto del nombre y muestre el valor de la tecla presionada en consola.

```

1 function asociarEvento(){
2     var elementoWeb=document.querySelector("input");
3     elementoWeb.onkeypress=funcionEvento;
4 }
5 function funcionEvento(event){
6     var tecla=event.which || event.keyCode;
7     console.log("Presionaste la tecla: " + String.fromCharCode(tecla));
8 }
9 asociarEvento();

```

JavaScript

document.getElementById('musica').onfocus=eventoOnFocus  
document.getElementById('animales').onblur=eventoOnBlur

```

elemento.onclick = mifuncion;
elemento.onmouseover = mifuncion;
elemento.onkeypress = mifuncion;
elemento.onkeydown = mifuncion;
elemento.onfocus = mifuncion;
elemento.onblur = mifuncion;

```

## JSON

Durante mucho tiempo XML fue la única opción que existía para compartir datos entre servidores y sistemas. XML era la solución a todos los problemas de intercambio de datos.

Uno de los mayores problemas de XML es que es muy poco legible para el usuario, ya que una estructura XML puede almacenar cualquier tipo de dato incluyendo imágenes, audio, video y mucho más, formando un código demasiado extenso. Esto puede ser peligroso a nivel de seguridad ya que se podría incluir código malicioso dentro de la estructura XML. Otra desventaja de los archivos XML es que son difíciles de tratar, además de que la sintaxis entre marcas, algo parecido al HTML, hace que sean muy pesados cuando tienen grandes cantidades de información.

JSON es una estructura que está limitada a almacenar datos clásicos y comunes, como números y textos, permitiendo que sea un formato ligero ideal para el transporte de datos. En la actualidad es el formato estándar usado para la comunicación de datos, superando a XML.

Ejemplo de la misma estructura en Formato XML y en JSON:

```

1  <Empleados>
2  <Empleado>
3    <Nombre>Claudia</Nombre>
4    <Edad>24</Edad>
5    <Profesion>Desarrolladora Web</Profesion>
6  </Empleado>
7  <Empleado>
8    <Nombre>Manuel</Nombre>
9    <Edad>44</Edad>
10   <Profesion>Desarrolladora iOS</Profesion>
11 </Empleado>
12 </Empleados>

```

```

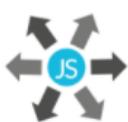
1  "Empleados" : [
2    "Empleado" : {
3      "Nombre" : "Claudia",
4      "Edad" : 24,
5      "Profesion" : "Desarrolladora Web",
6    },
7    "Empleado" : {
8      "Nombre" : "Manuel",
9      "Edad" : 40,
10     "Profesion" : "Desarrolladora iOS",
11   },
12 ]

```

## Ventajas de JSON



- JSON soporta dos tipos de estructuras: un set de pares llave-valor contenidos por un objeto y la otra es un array de valores.



- JSON es una estructura totalmente independiente, no necesita de tecnologías alternas.



- JSON es totalmente flexible y para extenderlo solo hace falta agregar una nueva propiedad con los valores deseados.
- JSON Tiene una alta velocidad de procesamiento y un menor tamaño con respecto a XML.

En conclusión, JSON es usado con frecuencia en ambientes web en los que fluye mucha información entre el cliente y el servidor; y también donde los tiempos de respuesta son vitales para el rendimiento.

## Geolocalizacion

```

if( navigator.geolocation ){
  //Obtenemos los datos de la ubicación del usuario
  navigator.geolocation.getCurrentPosition( function(position){
    var positionInfo = position;
  })
} else {
  alert('El navegador no soporta la característica de Geolocalización de HTML5')
}

```

<https://developers.google.com/maps/documentation/javascript/get-api-key>

La key se obtiene con el correo propio y permite crear una interfaz

## Local storage

Continuamos explorando las nuevas funcionalidades de HTML5 y sin duda otra de las mejores herramientas que se implementaron en esta versión Web Storage o Almacenamiento local. Esta herramienta guarda datos del usuario o de la aplicación directamente en la máquina del usuario y no en un servidor de bases de datos. Así, su funcionalidad facilita la vida de los desarrolladores a la hora de manipular datos que tienen que persistir constantemente. Para hacer uso de esta característica solo es necesario Javascript, por lo que es otra ventaja grande frente a otros métodos de almacenamiento. Además, con esta funcionalidad podemos guardar hasta aproximadamente 5MB de datos en el disco duro del usuario. Existe otra forma de guardar datos localmente que se conoce como cookies pero estas tienen graves problemas de seguridad y solo soportan unos 4KB. Como en todas las nuevas características el factor Compatibilidad es esencial y debe ser tenido en cuenta a la hora de implementar estas funciones. La página de caniuse nos puede ayudar con eso.

```

// Almacenar Datos
var datos = {...}
localStorage.setItem('NombreItem', JSON.stringify(datos))
sessionStorage.setItem('NombreItem', JSON.stringify(datos))
localStorage.NombreItem = JSON.stringify(datos)
sessionStorage.NombreItem = JSON.stringify(datos)

// Obtener Datos
var datos = JSON.parse(sessionStorage.getItem('NombreItem'))
var datos = JSON.parse(localStorage.getItem('NombreItem'))
// O También :
var datos = JSON.parse(localStorage.NombreItem)
var datos = JSON.parse(sessionStorage.NombreItem)

// Borrar Datos
localStorage.removeItem('NombreItem')
sessionStorage.removeItem('NombreItem')
// Borrar con la palabra delete
delete localStorage.NombreItem
delete sessionStorage.NombreItem

```

## Web Workers

Permite crear varios procesos simultáneamente

**Web Workers** son otra de las API's que le dieron tanto éxito a la versión 5 de HTML, rompiendo el esquema tradicional de **JavaScript** de correr todas sus tareas y operaciones en un único hilo de ejecución. Algo a tener en cuenta es que un worker **es un proceso totalmente aislado de nuestra aplicación** y no se ejecuta bajo el mismo contexto, ni tenemos acceso al DOM, por lo tanto, cualquier variable o elemento que queramos usar dentro del worker lo debemos pasar como un parámetro.

**NOTA:** Cuando creamos un worker y lo enlazamos a nuestro archivo principal, podemos crear una comunicación entre las partes por medio de la cual podemos enviar y recibir información de forma constante.

`new Worker()`

Para crearlo necesitas un archivo principal donde declaras el worker. Y para que inicie necesitas crear la instancia y pasarle la ruta y el nombre del script que ejecutará.

`postMessage()`

Es necesario para transmitir información entre el worker y el proceso principal.

`terminate()`

Es la función con la cual puedes finalizar el proceso del worker.

`var worker = new Worker('tareas.js')`

// Iniciar la Ejecución  
`worker.postMessage()`

// Escuchar los eventos  
`worker.addEventListener('message', function(e) {  
 console.log("El worker envio:" + e.data)  
}, false)`

// Enviar mensajes  
`worker.postMessage('Hola Worker!')`

¡Excelente! Un Web Worker está completamente aislado del objeto document, por ende, se arroja un error que indica que la variable document no está definida.

# FRAMEWORKS Y LIBRERÍAS JAVASCRIPT

**Aspectos:**

- 1 Documentación
- 2 Tiempo en el mercado
- 3 Mantenimiento
- 4 Que se adapte a requerimientos
- 5 Aplicaciones ya hechas
- 6 Popularidad
- 7 Desarrollador

Rápido  
 Mejor

## ¿Por qué usar un framework?

Los frameworks pueden compararse a casas prefabricadas. Vienen con muchas cosas ya hechas, tienen una estructura ya definida a la cual hay que adaptarse. Existen principios básicos por los cuales se desarrollaron los frameworks:



**Hacer menos código.**



**No reinventar la rueda:** existen tareas genéricas que ya otros han hecho y funcionan bien.



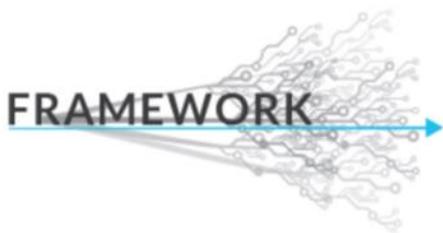
**Eficiencia:** dejando las tareas genéricas a un lado es posible concentrarse en la lógica de negocio.



**Arquitectura:** deben ayudar a organizar el código de tal forma que otros puedan mantenerlo después.

## ¿Cuándo usar un framework?

Depende de lo que se vaya a hacer así se determina si usar un framework o no. Algunos consejos son:



**Complejidad:** cuando lo que se va a desarrollar es varias veces más complejo que un “Hola mundo”.



Cuando exista un framework que ya haga gran parte de lo que necesitas.



Si se sabe que la aplicación va a crecer en el futuro.



Si se va a ahorrar tiempo y esfuerzo. Cuando se necesita entregar con prontitud un proyecto.



**Falta de recursos (financieros o humanos):** Puede que al equipo le falte profesionales que hagan mejor el trabajo de lo que lo puede hacer el framework.

## Ventajas

Algunas ventajas de usar frameworks son:



**Agilizar código:** Los frameworks te permiten agilizar el desarrollo despreocupándote de tareas que no están relacionadas con la lógica del negocio.



**Experiencia:** Si eres atento y juicioso ganarás experiencia a la hora de resolver problemas, pues aprenderás cómo los desarrolladores del framework resolvieron los suyos.



**Optimización:** Los frameworks fomentan el uso de buenas prácticas en programación lo que resulta en códigos más óptimos.

---



**Legado:** Si el framework tiene buena documentación, los futuros desarrolladores que mantendrán el código tendrán una buena base para hacer mantenimiento. Aunque, por supuesto, queda por tu parte hacer un código limpio y legible.



**Estructura:** Los frameworks brindan estructura al proyecto.



**Soporte:** Si se utiliza un buen framework, este tendrá una buena comunidad de desarrolladores que ayudarían a dar soporte al mismo.

---

## Desventajas

Algunas desventajas de usar frameworks son:

200kb

**Sobrecarga:** Cargar 200kb en un equipo core I7 de 8Gb de memoria con internet de 5Mb puede parecer rápido, pero cuando se trata de dispositivos móviles es otra cuestión.

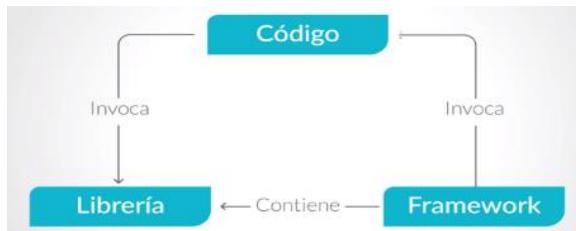


**Curva de aprendizaje:** Todo framework requiere una curva de aprendizaje y debe tenerse en cuenta a la hora de estimar el tiempo de proyecto. Cuando estudias un framework aprendes el framework, no el lenguaje.



**Te casas con él:** Los frameworks determinan cómo se deben hacer las cosas. Entre más completo sea el framework menos libertad tiene el desarrollador.

## Librería



## Porque usar una librería

Hay miles de librerías de javascript que puedes usar en tus proyectos. Estas librerías pueden ayudarte en aspectos como animación, **AJAX**, manipulación del **DOM**, manejo de eventos del DOM, creación de gráficos, entre otros. Si debes decidir entre si usar una librería o escribir el código por tí mismo, a continuación encontrarás razones de por qué usar librerías:

**Código testeado:** Por lo general el código de las librerías incluye tests que garantizan el buen funcionamiento de estas. Además los desarrolladores ya se habrán enfrentado a errores de código, descubierto por ellos o por la comunidad, lo cual hace que el código sea algo estable.



**Te enfocas en la lógica del negocio:** Al usar librerías en las que ya se han considerado los casos aislados, las incompatibilidades y particularidades de los navegadores, ahorras tiempo que puedes invertir en desarrollar propiamente tu aplicación.



**Reusar código:** Si encuentras una librería que soluciona lo que estás buscando, lo hace de una forma óptima y con buenas prácticas, es una buena opción utilizarla. Si la solución no se adapta a lo que necesitas puedes considerar hacer una contribución al desarrollo de la librería añadiendo tu funcionalidad, si tu

contribución es aceptada, ¡excelente!, puedes usar la librería. Recuerda que una buena práctica es nunca modificar librerías de terceros localmente. Esto puede traer problemas de mantenimiento a la hora de actualizar la librería a una versión nueva.



Como desarrollador muchas veces te enfrentarás a la decisión de qué librería usar para un reto en particular. En ocasiones existen múltiples librerías diseñadas para dar solución al mismo problema, ¿cuál debes escoger? A continuación unos consejos:

- El desempeño debe ser bueno, de manera que no impacte de forma negativa el tiempo de respuesta de tu aplicación web.
- Que exista una comunidad de desarrollo activa.
- Buena documentación.
- Código Limpio.
- Que haya fácil integración con el proyecto.
- En lo posible, que las librerías en cuestión no dependan de otras librerías para su funcionamiento.

Recuerda añadir solo las librerías estrictamente necesarias, a menor cantidad de librerías será más sencillo tener control del desempeño de tu código y hacer debug para encontrar cuellos de botella.

## Frameworks mas usados



### Angular.js

- Alta complejidad.
- Modular.
- Se hace cargo de todo.
- Versión actual: 1.5.7 - 2
- Sitio web: <https://angularjs.org/>

### Ember.js

- Funciona.
- Ahorro en tiempo y presupuesto..
- No se necesita flexibilidad.
- Versión actual: 2.1.0
- Sitio web: <http://emberjs.com/>

EmberJS Framework que se encarga de la productividad

Mocha Framework para hacer test de aplicaciones

Aurelia Framework para móviles y aplicaciones de escritorio

AngularJS Framework soportado por Google para proyectos robustos y complejos



**Simplicidad:**

- Al manipular el DOM
- Manejar eventos
- Hacer animaciones

**lodash.js**

Array	Number
Chain	Object
Collection	String
Function	Utility
Lang	Methods
Math	Properties



datepickers  
sliders  
drag & drop

<b>Mustache</b>	Librería para trabajar plantillas dinámicas
<b>Chai</b>	Muy usada actualmente para realizar testing
<b>Highcharts</b>	Óptima para hacer gráficos de reportes
<b>jQueryUI</b>	Proporciona una potente interfaz de usuario

## Canvas

```
<canvas id="micanvas" width="200" height="200">
  <p>Tu explorador no soporta canvas.</p>
</canvas>
```



Konva es un framework para canvas que permite añadir interactividad, hacer animaciones con buen desempeño, transiciones, capas, filtrado y más para aplicaciones de escritorio y móviles. A continuación haremos una comparación de los elementos antes vistos ahora utilizando Konva:

## Convencional

## Konva

### Lineas

```
context.beginPath();
context.moveTo(0,0);
context.lineTo(400,200);
context.stroke();
```

```
var line=new Konva.line({
  points:[0,0,400,200],
  stroke: "black",
  strokeWidth: 1,
});
```

### Rectángulos

```
context.rect(100, 50, 50, 100);
context.stroke();
```

```
var rect=new Konva.Rect({
  x: 100,
  y: 50,
  height: 100,
  stroke: "black",
  strokeWidth: 5,
  fill: "blue"
});
```

### Arcos

```
context.arc(100, 100, 50, Math.PI, Math.PI*1.5, false);
context.stroke();
```

```
var arc=new Konva.Arc({
  x: 100,
  y: 50,
  innerRadius: 50,
  outerRadius: 50,
  angle: 90,
  stroke: "black",
  strokeWidth: 1,
  rotation: 180
});
```

### Círculos

```
context.arc(120, 120, 70, 0, 2*Math.PI, false);
context.fillStyle="#FF8800";
context.fill();
```

```
var circle=new Konva.Circle({
  x: 120,
  y: 120,
  radius: 70,
  fill: "#FF8800"
});
```

### Textos

```
context.font="bold 30pt Arial, sans";
context.fillText("Hola mundo", 20, 50);
```

```
var text=new Konva.Text({
  x: 120,
  y: 120,
  text: "Hola mundo",
  fontSize: 30,
  fontFamily: "Arial",
});
```

### Imágenes

```
var imageObj=new Image();
imageObj.onload=function(){
context.drawImage(imageObj, 50, 50, 200, 150);
};
imageObj.src="images/fruits.png"
```

```
var imageObj=new Image();
imageObj.onload=function (){
var fruits=new Konva.Image({
x: 50,
y: 50,
image: imageObj,
width: 200,
height: 150
});
layer.add(fruits);
stage.add(layer);
};
imageObj.src="images/fruits.png"
```

## Eventos

Para añadir manipuladores de eventos a figuras en Konva usamos el método `on()`. Para equipos de escritorio, Konva soporta los

eventos `mouseover`, `mouseout`, `mouseenter`, `mouseleave`, `mousemove`, `mousedown`, `mouseup`, `mousewheel`

`click`, `dblclick`, `dragstart`, `dragmove`, y `dragend` En dispositivos móviles, Konva

soporta `touchstart`, `touchmove`, `touchend`, `tap`, `dbltap`, `dragstart`, `dragmove`, y `dragend`

### Arrastrar y soltar figura

Para arrastrar y soltar figuras con Konva, podemos utilizar el método `draggable()` ó configurar en `true` la propiedad `draggable`. Esto permite arrastrar y soltar figuras tanto en aplicaciones de escritorio como en móviles.

## Grupos

Agrupar figuras es muy útil cuando queremos hacerles alguna transformación por ejemplo mover, rotar o escalarlas al mismo tiempo. De hecho se pueden formar agrupaciones más complejas añadiendo grupos a los ya existentes. Para agrupar múltiples figuras con Konva podemos crear un objeto `Konva.Group()` y luego añadir figuras con el método `add()`

## Animaciones

Podemos usar el constructor `Konva.Animation` para crear animaciones. Este constructor acepta dos parámetros: una función de actualización y opcionalmente una capa o arreglo de capas que serán actualizadas en cada fotograma de la animación. La función de actualización sólo debe contener lógica que actualice propiedades de los nodos como `position`, `rotation`, `scale`, `width`, `height`, `radius`, `colors`, etc

Para iniciar la animación utilizamos el método `start()` y para detenerla el método `stop()`.

Por ejemplo para hacer rotar una figura podemos hacerlo con:

```
var anim=new Konva.Animation(function(frame){  
blueRect.rotate(frame.timeDiff*90/1000);  
}, layer);  
anim.start();
```

Donde la propiedad `timeDiff` del objeto `frame` es el número de milisegundos que han pasado desde el último fotograma.

## Mejorando el desempeño

Una de las formas que tiene Konva de mejorar el desempeño es el almacenamiento en Caché. Esta técnica te permite dibujar un elemento en un búfer de canvas y luego dibujarlo en el canvas. Para lograrlo se utiliza el método `shape.cache()`

Para ampliar más la información y ver la lista completa de opciones que Konva ofrece con sus respectivos ejemplos te invitamos a revisar la documentación oficial de Konva: <http://konvajs.github.io/docs/>

## PROTOTYPE

**Prototype** es un framework JavaScript creado por Sam Stephenson con el fin de optimizar y simplificar el desarrollo de páginas web. Esta librería se encarga, entre otras cosas, de facilitar la selección de elementos del DOM, manipularlos, e integrarlos para realizar páginas web altamente interactivas y dinámicas. El enfoque principal de Prototype es la codificación orientada a objetos, es decir, manipular y crear elementos del DOM que sean afines al manejo de clases, atributos y métodos. Este framework incluye funcionalidades compatibles con Ajax y Ruby on Rails. Actualmente se encuentra en la versión 1.7.3, lanzada el 22 de Septiembre de 2015.

Al ser un framework JavaScript, Prototype es una alternativa a jQuery, por tal motivo es importante decidir qué tipo de librería se va a usar en un proyecto web, de acuerdo con las características y funcionalidades que puede aportar. Prototype es un framework recomendado para proyectos que integran Ruby on Rails, o para desarrolladores familiarizados con este, ya que su sintaxis y nomenclatura son muy similares a las de Prototype. A su vez, es muy útil en proyectos en los que hay un manejo predominante de clases y objetos.

La sintaxis de Prototype es muy sencilla, ya que encapsula muchas funcionalidades y características JavaScript en sencillas funciones y objetos. En general maneja una sintaxis muy similar a la usada en jQuery involucrando tres partes principales: selector, punto y función.

### selector.función

El selector indica un elemento del DOM sobre el cual se realizarán las acciones, el punto es el carácter que indica la invocación de una función a un elemento determinado, y la función indica las acciones que se asignan al ítem seleccionado. Sin embargo, dado que Prototype es un framework basado en la programación orientada a objetos, involucra la sintaxis propia de la definición de objetos en JavaScript. Esto es, el uso de las llaves para encerrar la definición de un objeto, así como la presencia de un método initialize que se ejecuta al instanciar un objeto de la clase definida.

```
Var Person = Class.create();
Person.prototype = {
  initialize: function(name) {
    this.name = name;
  },
  say: function(message) {
    return this.name + ': ' + message;
  }
};
```

Adicionalmente en Prototype se usan objetos propios de la librería como Class, Element y Object que se usan en lugar del selector en la sintaxis descrita anteriormente.

### Descarga e Instalacion

<http://prototypejs.org/download/>

### Selectores y css

Se usa el mismo de jquery pero sin #

```
$(‘id’) = getElementById(‘id’)
$$(‘reglaCSS’) $$('.main')[0].next().childElements()[1].descendants()[1]
```

## Funciones de selección

```
$F('idElemento')
```

```
$F('nombre')  
"peter "
```

```
.previous - .next
```

```
$(‘telefono’).previous(‘label’)
```

```
<label for="telefono"> ○
```

```
$$('h2')[0].next()
```

```
<div class="item-form"> ○
```

```
$$('.item-form')[0].childElements()  
Array [ <label> ○ , <input#nombre> ○ ]
```

```
.childElements()
```

```
$(‘formulario’).descendants()    $(‘email’).ancestors()
```

## Búsqueda sobre una selección

```
$(‘idSelección’).select(‘reglaCSS’)
```

```
• $$("div.item-form")[0].select(‘input’)
```

```
var objetoSelec = new Selector(‘reglaCSS’)
```

```
var s = new Selector(‘label’)
```

```
undefined
```

```
s
```

```
Object { expression: "label" }
```

```
(‘contenedor’).next().select(‘.col’)
```

Se selecciona el elemento hermano después de un elemento con el id "contenedor", y en él, se buscan todos los elementos de la clase "col".

```
$$('.contenedor')[0].previous().select(‘#col’)
```

Se selecciona el elemento hermano previo al primer elemento con la clase "contenedor", y en él, se busca el elemento con el id "col".

```
var a = new Selector(‘.contenedor’);  
a.findElements($(‘col’))
```

Se selecciona el elemento de la clase "contenedor" y en él busca un elemento con el id "col".

## Objetos en Prototype

- argumentNames
- bind
- bindAsEventListener
- curry
- defer
- delay
- methodize
- wrap

**argumentNames()**: permite ver los nombres de parámetros que tiene una función.

**bind()**: permite usar el this en algún contexto con el objeto correspondiente

**defer()**: aplaza la ejecución de una función

**delay()**: recibe un tiempo estimado para ejecutar una función

### Métodos de validación de tipos de datos

Métodos:

- isArray
- isDate
- isElement
- isFunction
- isNumber
- isString
- isUndefined

### Métodos String

camelize

**capitalize**

empty

**include**

**Camelize()**: quita los guiones de un texto y pone mayúscula la letra siguiente

**Capitalize()**: pone la primera letra en mayúscula

**Empty**: valida si la cadena esta vacia

**Include()**: busca una cadena secundaria dentro del texto

String.prototype.anchor() Permite añadir un elemento DOM de tipo <a> con el texto String y la cadena de caracteres que se pase como parámetro al método que en el innerHTML.

String.prototype.bold() Permite añadir un elemento DOM de tipo <b> con el texto del String, al que se asocia el método, en el innerHTML.

String.prototype.link() Permite crear un link con la cadena de caracteres a la que se le aplique el método prototype.link y asignarle el url que se pase como parámetro de esta función.  
Ejemplo: cadena\_caracteres.prototype.link(url);

String.prototype.italics() Permite crear un elemento DOM de tipo <i> con el texto en el innerHTML que se pase como atributo.

String.prototype.sub() Permite crear un elemento DOM con el texto del objeto String al que se asocia el método.

### Eventos prototype: 2 formas

#### Clase Event

- Al mencionar "**definir la clase**", mostrar:  
Event
- Al mencionar "**método observe**" añadir:  
Event.observe()
- Al mencionar "**elemento del DOM**" añadir:  
Event.observe('idElemento')
- Al mencionar "**evento que se está asignando**" añadir:  
Event.observe('idElemento', 'click')
- Al mencionar "**la función que se ejecutará**" añadir:  
Event.observe('idElemento', 'click', funcionHandler )

```
document.observe("dom:loaded",function(){
  Event.observe('boton-cancelar','click', function(){
    alert('Estas seguro?')
  })
})
```

## Objetos del DOM

- Al mencionar “**selector de id**” mostrar título:  
      \$('idElemento')
- Al mencionar “**método observe**” añadir:  
      \$('idElemento').observe()
- Al mencionar “**tipo de evento**” añadir:  
      \$('idElemento').observe('click')
- Al mencionar “**handler**” añadir:  
      \$('idElemento').observe('click',funcionHandler)

```
$('titulo').observe('mouseover', function(){
    this.addClassName('activo');
})

$('titulo').observe('mouseleave', function(){
    this.removeClassName('activo');
})
```

### Objeto this y event

```
document.observe('dom:loaded', function(){
    $('navbar').observe('click', function(event){
        this.setStyle({background: 'gray'});
        event.findElement().setStyle({color: 'yellow'});
    })
})
```

El color de fondo de la lista será gris, y el de la letra del enlace “Nuestras sedes” amarillo.

### Remover eventos

**.stopObserving()**

```
$('#tablero').stopObserving('click');
```

# DESARROLLADOR JAVASCRIPT

## Objetos y funciones

### Objetos



Existen tres formas para crear un objeto:

- ✓ Lista de pares de valores propiedad/valor :  

```
var producto = {codigo:"0001",
    nombre:"Gorra", precio:500}
```
- ✓ Usando la palabra reservada new:  
  

```
var producto = new Object();
producto.codigo = "005";
producto.nombre = "camiseta";
producto.precio = 7000;
```
- ✓ Usando una función constructor:  

```
function producto(codigo, nombre, precio) {
    this.codigo = codigo;
    this.nombre = nombre;
    this.precio = precio;
}

var producto1 = new producto("001", "Gorra", 50);
```

```

//Primera forma para crear un objeto
var producto1 = {
  codigo: "0001",
  nombre: "Gorra",
  precio: 5000
};

//Segunda forma para crear un objeto
var producto2 = new Object();
producto2.codigo = "005";
producto2.nombre = "Camiseta";
producto2.precio = 7000;

//Tercera forma para crear un objeto
function producto3(codigo, nombre, precio) {
  this.codigo = codigo;
  this.nombre = nombre;
  this.precio = precio;
}

var producto3 = new producto3("008", "Zapatos", 45000);

//Accediendo a las propiedades de los objetos para mostrarlas.
document.getElementById("c1").innerHTML = producto1.codigo;
document.getElementById("n1").innerHTML = producto1.nombre;
document.getElementById("p1").innerHTML = producto1.precio;

// Llamado a una función desde el código que recibe dos parámetros y retorna un valor
function division(p, q) {
  return p / q;
}

var resultado1 = division(10, 2);
document.getElementById("resultado1").innerHTML = resultado1;

// Llamado de una función así misma
(function() {
  document.getElementById("resultado2").innerHTML = "El resultado se invoca a sí misma";
})();

//Asignación de una función a una variable.
var z = function(x, y) {
  return x - y
};

var w = z(15, 6);
document.getElementById("resultado3").innerHTML = w;

```

## Uso de las propiedades y métodos de un objeto

```

function Persona (nombre, edad, ciudad, email) { // definición de la clase Persona.

  //SETIAMOS LOS ATRIBUTOS DE LA CLASE PERSONA CON LOS PARAMETROS RECIBIDOS

  this.nombre = nombre;
  this.edad = edad;
  this.ciudad = ciudad;
  this.email = email;

  //Método saludar() propio de la clase Persona

  this.saludar = function() {
    document.getElementById("saludo1").innerHTML = "Hola mi nombre es " + this.nombre + " y mi edad es " + this.edad;
  };
}

var persona = new Persona("Carlos Morales", "24", "Cartagena", "carlos@hotmail.com");
//Creacion de un objeto de la clase Persona

function miFuncion1(argument) {
  persona.saludar(); // invocación del método saludar() a través del objeto persona
}

function miFuncion2(argument) {
  //Acceso a los atributos ciudad e email del objeto persona
  document.getElementById("saludo2").innerHTML = "Soy de " + persona.ciudad + " y mi E-Mail es " + persona.email;
}

```

## Tipos de funciones

- Closures
- Callbacks
- Anónimas
- Autoejecutables

### Closures:

Funciones internas que manejan variables independientes

```

Callbacks
1 function calificado (mensaje) {
2   alert(mensaje);
3 };
4 (function (nombre, apellido, nota1, nota2, nota3, calificado) {
5   var nombreCompleto = nombre + " " + apellido;
6   var sumaNotas = nota1 + nota2 + nota3;
7   var calificacionTotal = sumaNotas / 3;
8   var resultado = nombreCompleto + " obtuvo " + calificacionTotal;
9   calificado(resultado);
10 })(“Jason”, “Roberts”, 5.8, 4.2, 3.5)

Función anónima
1 setTimeout (function () {
2   var nombreCompleto = ‘Jason Roberts’;
3   var sumaNotas = 5.8 + 4.2 + 3.5;
4   var calificacionTotal = sumaNotas / 3;
5   alert(nombreCompleto + ‘ obtuvo ’ + calificacionTotal);
6 }, 1000)

Closures
1 function operaciones () {
2   var multiplicacion = 3569 * 5648;
3   var suma = 3569 + 5644;
4   function resultados() {
5     alert (“la suma ” + suma + “ multiplicación ” + multiplicacion);
6   }
7   resultados();
8 }
9 operaciones();


```

**Scope:** Ámbitos y alcance de las **variables**. global o local

String y sus métodos

String.prototype	Object.prototype	Number.prototype	Date.prototype
<a href="#">replace()</a>	<a href="#">search()</a>	<a href="#">slice()</a>	<a href="#">split()</a>
<a href="#">substr()</a>	<a href="#">substring()</a>	<a href="#">toLocaleLowerCase()</a>	<a href="#">toLocaleUpperCase()</a>
<a href="#">toLowerCase()</a>	<a href="#">toString()</a>	<a href="#">trim()</a>	<a href="#">valueOf()</a>

```

45 // Metodos String
46
47 var string1 = “Esta es”;
48 var string2 = “ una cadena de”;
49 var string3 = “ texto concatenada!”;
50 var stringContact = string1.concat(string2, string3); // El metodo concat() permite concatenar strings.
51 document.getElementById(“s5”).innerHTML = stringConcat;
52
53 var mensaje1=“Programando en .NET”;
54 document.getElementById(“s6”).innerHTML = mensaje1;
55
56 var mensaje2 = mensaje1.replace(“.NET,” “JavaScript”); // El metodo replace() permite reemplazar una parte de una cadena
de texto.
57 document.getElementById(“s7”).innerHTML = mensaje2;
58
59 var mensaje3 = “Luchar, reir y vivir; Esa es la clave”;
60
61 document.getElementById(“s8”).innerHTML = mensaje3;
62
63 document.getElementById(“s9”).innerHTML = mensaje3.substring(8,20); //El metodo substring() permite extraer una subcadena
de acuerdo a un rango de indices.
64
65 var mensaje4 = “EL IGNORANTE AFIRMA; EL SABIO DUDA Y REFLEXIONA”;
66
67 document.getElementById(“s10”).innerHTML = mensaje4;
68 document.getElementById(“s11”).innerHTML = mensaje4.toLowerCase();
69
70 var mensaje5 = “La verdadera vida tiene lugar en nuestro interior”;
71
72 document.getElementById(“s12”).innerHTML = mensaje5;
73 document.getElementById(“s13”).innerHTML = mensaje5.toUpperCase();

```

```

//Conversion de Number a String

var string1 = String(15);
var string2 = String(3 + 4);

var number1 = 20;

var string3 = number1.toString();
var string4 = (30).toString();

//Conversion de Boolean a String

var string5 = String(true);
var string6 = String(false);

var string7 = true.toString();
var string8 = false.toString();

// Conversion de Date a String

var string9 = String(Date());
var string10 = Date().toString();

// Conversion de String a Number

var number2 = Number("6.5");
var number3 = Number("");

var number4 = parseInt("2014");
var number5 = parseFloat("3.1416")

```

## Numéricos

```

44 //Propiedades de number.
45
46 var numero3=23434;
47 document.getElementById("n3").innerHTML = numero3.constructor; //La propiedad constructor retorna la funcion nativa que crea un numero.
48
49 document.getElementById("n4").innerHTML = Number.MAX_VALUE; // La propiedad MAX_VALUE retorna el numero maximo que se puede usar en JavaScript.
50
51 document.getElementById("n5").innerHTML = Number.MIN_VALUE; //La propiedad MIN_VALUE retorna el valor minimo que se puede usar en JavaScript.
52
53 document.getElementById("n6").innerHTML = Number.NEGATIVE_INFINITY; // La propiedad NEGATIVE_INFINITY retorna un negativo infinito.
54
55 document.getElementById("n7").innerHTML = Number.NaN; //La propiedad NaN retorna un numero ilegal.
56
57 document.getElementById("n8").innerHTML = Number.POSITIVE_INFINITY; // La propiedad POSITIVE_INFINITY retorna un positivo infinito.
58
59 Number.prototype.metodoNumerico = function() { //La propiedad prototype permite crear un nuevo metodo para manipular numeros.
60
61     return this*4/2;
62 }
63
64
65 var numero4 = 25;
66 document.getElementById("n9").innerHTML = numero4.metodoNumerico();

//Metodos de number.

var numero5 = 26.57;
document.getElementById("n10").innerHTML = numero5.toExponential(); //Convierte un numero a expresion exponencial.

var numero6 = 14.7854243;
document.getElementById("n11").innerHTML = numero6.toFixed(3); //Recorta el numero de digitos de un numero y lo convierte en string.

var numero7 = 23.5567;
document.getElementById("n12").innerHTML = numero7.toPrecision(2); // Da formato a un numero para una longitud de numeros n.

var numero8 = 3.1416;
document.getElementById("n13").innerHTML = numero8.toString(); // Convierte un numero en una cadena

var numero10 = 5000;
document.getElementById("n14").innerHTML = numero10.valueOf(); //Devuelve el valor simple de un numero.

</script>

```

```
//Metodos

// parseInt convierte un string en un numero entero
var c=parseInt("2.25"); //2

var d=parseInt("1 Next University"); //1
var e=parseInt("Next University 10"); //NaN

// parseFloat convierte un string en numero con punto decimal
var f=parseFloat("3.1416"); //3.1416
var g=parseFloat("2.2332 Hola a Todos!"); // 2.2332
var h=parseFloat("I Love JS 32.434"); //NaN

var uri = "http://nextuniversity.com/es_extra_info/index.html";
var uriEnc = encodeURIComponent(uri); // encodeURIComponent codifica una URL
var uriDec = decodeURIComponent(uriEnc); //decodeURIComponent decodifica una url
```

Valida si la entrada es numérica o no

```
function validar(num1, num2) {
    if (isNaN(num1) || isNaN(num2)) {
        return false;
    } else {
        return true;
    }
}
```

Operadores matemáticos y trigonometricos Math:

```
//Objeto Math

var a=Math.PI; //Costante PI
var b=Math.cos(90); //Metodo cos() permite calcular el coseno de un numero.
var c=Math.sin(180); // Metodo sin() permite calcular el seno de un numero.
var d=Math.max(1,20); // Metodo max() permite calcular el numero maximo entre varios numeros.
var e=Math.min(13,3); // Metodo min() permite calcular el numero minimo entre varios numeros.
var f=Math.pow(2,4); // Metodo pow () permite calcular la potencia de un numero.
var g=Math.sqrt(144); // Metodo sqrt() permite calcular la raiz cuadrada de un numero.
```

[https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos\\_globales/Math](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Math)

Operadores generales

```
//operadores de comparacion
var op1 = a < b;
var op2 = a > b;
var op3 = a == b;
var op4 = a != b;
var op5 = "perro" == "perro";
var op6 = "camello" == "camello";
var op7 = "casa" != "caza";

document.getElementById("op1").innerHTML = op1;
document.getElementById("op2").innerHTML = op2;
document.getElementById("op3").innerHTML = op3;
document.getElementById("op4").innerHTML = op4;
document.getElementById("op5").innerHTML = op5;
document.getElementById("op6").innerHTML = op6;
document.getElementById("op7").innerHTML = op7;

//Operadores aritmeticos

var op8 = a + b;
var op9 = a - b;
var op10 = a * b;
var op11 = a / b;
var op12 = a % b;
var op13 = a++;
var op14 = b--;
```

Date

## Object Date más utilizados:

getgetFullYear()	var date = new Date(); date.getFullYear();	getMonth()	var date = new Date(); date.getMonth();
setFullYear(año)	var date = new Date(); date.setFullYear(2015);	setMonth(mes)	var date = new Date(); date.setMonth(5);
			
getDate()	var date = new Date(); date.getDate();	getHours()	var date = new Date(); date.getHours();
setDate(dia)	var date = new Date(); date.setDate(30);	setHours(hora)	var date = new Date(); date.setHours(12);

```
//Objeto Date

var date1=new Date();
var date2=new Date("December 26, 2014 12:30:20");
var date3=new Date(2015, 8, 5);

document.getElementById("d1").innerHTML=date1;
document.getElementById("d2").innerHTML=date2;
document.getElementById("d3").innerHTML=date3;

//Objeto Date Metodos.

document.getElementById("d4").innerHTML=date1.getFullYear();
date1.setFullYear(2018);
document.getElementById("d5").innerHTML=date1.getFullYear();

document.getElementById("d6").innerHTML=date2.getMonth();
date2.setMonth(5);
document.getElementById("d7").innerHTML=date2.getMonth();

document.getElementById("d8").innerHTML=date3.getDate();
date3.setDate(19);
document.getElementById("d9").innerHTML=date3.getDate();

document.getElementById("d10").innerHTML=date1.getHours();
date1.setHours(4);
document.getElementById("d11").innerHTML=date1.getHours();
```

## Arrays

# Operaciones con Arrays

### Modificar valores:

```
var futbol = ["Balón", "Jugadores", "Cancha"];
futbol[1] = "Futbolistas"; → ["Balón", "Futbolistas", "Cancha"]
```

### Determinar la cantidad de datos de un Array:

```
var futbol = ["Balón", "Jugadores", "Cancha"];
futbol.length → resultado será 3
```

### Invertir orden de un Array:

```
var futbol = ["Balón", "Jugadores", "Cancha"];
futbol.reverse() → ["Cancha", "Jugadores", "Balón"];
```

### Concatenar Array:

```
var futbol = ["Balón", "Jugadores", "Cancha"];
var tenis = ["Raqueta", "Pelota", "Cancha", "Jugadores"];
document.write(futbol.concat(tenis));
→ ["Balón", "Jugadores", "Cancha", "Raqueta", "Pelota", "Cancha", "Jugadores"]
```

### Almacenamiento de datos de distintos tipos:

```
MyMatriz[0]= myFuncion;
MyMatriz[1]=futbol;
```

### Almacenar valor en la última posición:

```
var futbol = ["Balón", "Jugadores", "Cancha"];
futbol[futbol.length] = "Juez"; → ["Balón", "Jugadores", "Cancha", "Jue
```

### Ordenar un Array:

```
var futbol = ["Balón", "Jugadores", "Cancha"];
futbol.sort(); → ["Balón", "Cancha", "Jugadores"]
```

```

//Arrays definicion

var futbol = ["Balon", "Jugadores", "Cancha"];
var animales = new Array("Oso", "Aguila", "Ballena");

document.getElementById("a1").innerHTML=futbol;
document.getElementById("a2").innerHTML=animales;

//Arrays operaciones

futbol[1] = "Arbitros"; // Reemplazar un valor de un array
document.getElementById("a3").innerHTML=futbol;

document.getElementById("a4").innerHTML=futbol.length; // tamano de un array
futbol[futbol.length] = "Hinchada"; //almacenar un valor en la posicion de un array
document.getElementById("a5").innerHTML=futbol;

futbol.reverse(); //Invertir el orden de los elementos.

document.getElementById("a6").innerHTML=futbol;

animales.sort(); // Ordenar los elementos.
document.getElementById("a7").innerHTML=animales;

var arrayConca = animales.concat(futbol); // Concatenar dos arrays.
document.getElementById("a8").innerHTML=arrayConca;

```

### Uso del debugger: poner puntos de interrupcion

```

<script>

var numero1=30;
var numero2=80;
var numero3=80;

var resultado=(numero1+numero2)*numero3;
debugger;

console.log("Resultado: "+resultado);

</script>

```

### Uso del ciclo for

```

function mostrar_equipos() {
    var texto = "Equipos: <br>";

    var equipos = ["Liverpool", "Barcelona", "Juventus", "Real Madrid", "Roma"];
    for (var i = 0; i < equipos.length; i++) {
        texto += equipos[i] + " <br> ";
    }
    document.getElementById("salida").innerHTML=texto;
}

function hacerTabla() {
    var tabla = "";
    var num = parseFloat(document.getElementById("num").value);

    if (isNaN(num) == false) {
        tabla += 'TABLA DEL ' + num + '<br>';
        for (var i = 0; i <= 10; i++) {
            tabla += num + ' x ' + i + ' = ' + num * i + '<br>';
        }
        document.getElementById("tabla").innerHTML = tabla;
    } else {
        alert("Formato Incorrecto de numero!!!");
    }
}

```

## Uso del ciclo for in

```
<button onclick="mostrar_producto()">Mostrar Producto</button>

</body>

<script>
function mostrar_producto() {
    var texto="Producto:<br>";
    var producto = {
        codigo: "001",
        nombre: "lapiz",
        precio: 2000
    };
    var x;
    for (x in producto) {
        texto+=producto[x]+"<br>";
    }
    document.getElementById("salida").innerHTML = texto;
}
```



## Uso del ciclo while

```
<button onclick="conteo()">Iniciar Conteo</button>

</body>

<script>
function conteo() {
    var contador=1;
    var texto="";
    while(contador<=50) {
        texto+= contador+"<br>";
        contador++;
    }
    document.getElementById("salida").innerHTML=texto;
}
</script>
```

## Uso del ciclo do while

```
<button onclick="conteo()">Iniciar Conteo</button>

</body>

<script>
function conteo() {
    var contador = 50;
    var texto="";
    do {
        texto += contador + "<br>";
        contador++;
    } while (contador <= 100)
    document.getElementById("salida").innerHTML = texto;
}
</script>
```

## Manejo de errores

```
function miFuncion1() {  
    try {  
        var numero1=50;  
        var numero = numero1 + numero2; // La variable num  
    } catch (error) {  
        alert("Existe un Error" + " - " + error.message);  
    }  
}  
  
function miFuncion2() {  
    try {  
        var persona=null;  
        var nombre= persona.nombre // Se esta tratando de  
    } catch (error) {  
        alert("Existe un Error" + " - " + error.message);  
    }  
}
```

## Errores controlados y personalizados con throw

```
function miFuncion() {  
    try {  
        var mensaje="Esto es un mensaje desde Java Script";  
        if (isNaN(parseInt(mensaje))){  
            throw "Parametro pasado al metodo parseInt no es un numero!";  
        }  
    } catch (error) {  
        alert("Existe un Error" + " - " + error);  
    }  
}  
  
function miFuncion2() {  
    try {  
        var res=45/0;  
        if(res==Infinity) {  
            throw "Division entre cero no es valida!";  
        }  
    } catch (error) {  
        alert("Existe un Error" + " - " + error);  
    }  
}
```



## EVENTOS JS

### Al salir de un control

- ✓ **Evento onblur:** <button>, <input>, <label>, <select>, <textarea>, <body>.

### Al cambiar el valor de un elemento y perder el foco

- ✓ **Evento onchange:** <input>, <select>, <textarea>.

### Al dar clic

- ✓ **Evento onclick:** Todos los elementos.

### Al dar doble click

- ✓ **Evento ondblclick:** Todos los elementos.

### Al entrar al control(foco)

- ✓ **Evento onfocus:** <button>, <input>, <label>, <select>, <textarea>, <body>.

### Al presionar una tecla sin soltarla

- ✓ **Evento onkeydown:** <body> y todos los elementos de un formulario.

### Al presionar teclas

- ✓ **Evento onkeypress:** <body> y todos los elementos de un formulario.

### Al soltar una tecla presionada

- ✓ **Evento onkeyup:** <body> y todos los elementos de un formulario.

### Al cargar el body y usar el btn del mouse

- ✓ **Evento onload:** Solo el <body>.

- ✓ **Evento onmousedown:** Todos los <sup>elementos</sup>.

### Al pasar el mouse sobre un elemento

- ✓ **Evento onmouseover:** Todos los elementos.

### Cuando se borran los datos de un form

- ✓ **Evento onreset:** Solo el <form>.

### Al dar clic en el btn submit del form

- ✓ **Evento onsubmit:** Solo el <form>.

### Cuando se selecciona un texto

- ✓ **Evento onselect:** Solo el <input> y el <textarea>.

### Al salir de la pagina

- ✓ **Evento onunload:** Solo el <body>.

Agregar evento a un elemento: `elemento.addEventListener("click", onFaceClick);`

## CONSEJOS desarrollo

- ✓ Lentitud en la ejecución y vulnerables a grandes riesgos.

Sustituir los new Array por [ ].

Ejemplo: var a= {value1, value2,.., valueN};



Sustituir los new String() por "".

Ejemplo var b="value";

Sustituir los new function por ( ) { }.

Ejemplo: var c=function (){};

## DOM – BOM

```
document.getElementById("id"); //Trae por id un elemento
```

```
document.getElementsByTagName("p"); //Trae por tag uno o varios elementos
```

```
document.getElementsByName(); //Trae por name uno o varios elementos
```

```
document.getElementsByClassName(); //Trae por nombre de clase uno o varios elementos
```

```
document.querySelector("a") -->
```

```
<a href="#">Link</a>      document.querySelector(".card-content img");
```

```
function cambiarAtributo() {
    var cajatxt = document.getElementById("nombre");
    cajatxt.setAttribute("size", "60");
}

function cambiarSpans() {
    var spans = document.getElementsByClassName("miClase");

    for (var i = 0; i < spans.length; i++) {
        spans[i].style.fontFamily = "Verdana";
    }
}

function agregarElemento() {
    var encabezado4 = document.createElement("h1");
    var texto = document.createTextNode("Mi texto 4"); //createTextNode permite crear un nodo de tipo texto.
    encabezado4.appendChild(texto);
    document.getElementById("miDiv").appendChild(encabezado4);
}

function objetoBody() {
    var div = document.body.children[0]; //obtenemos el objeto document.body para obtener el primer elemento hijo de la etiqueta body.

    var texto = document.createTextNode("Mi nuevo texto!!!!"); //Creamos un nodo de tipo texto.
    div.appendChild(texto)
}

function removerElemento() {
    var miDiv=document.getElementById("miDiv");
    miDiv.removeChild(miDiv.childNodes[0]); //Se invoca el metodo removeChild pasandole como argumento el nodo hijo que se va a remover.
}

document.write("<h1 style='color:red;font-weight: bold;'>Hola a todos!</h1>");

document.getElementById("miDiv").style.color = "green";

function cambiarDivs() {
    var divs = document.getElementsByTagName("div");

    for (var i = 0; i < divs.length; i++) {
        divs[i].style.color = "orange";
    }
}
```

## Manejador de Eventos DOM: agrega eventos a un elemento

```
<button id="boton1">Mostrar Alert</button>

<script>
var elemento = document.getElementById("boton1");
elemento.addEventListener("click", mostrarMensaje1);
elemento.addEventListener("click", mostrarMensaje2);

function mostrarMensaje1(argument) {
    alert("EventListener1!!!!");
}

function mostrarMensaje2(argument) {
    alert("EventListener2!!!!");
}
```

## Navegación nodos DOM

Accede al hijo de un nodo (al primero) y le cambia el texto a otro

```
<h3 id="texto1">Navegando entre Nodos</h3>
<p id="texto2">Hola Mundo</p>
<script>
var cambiartexto= document.getElementById("texto1").childNodes[0].nodeValue;
document.getElementById("texto2").innerHTML = cambiart;
```

Trae un nodo hermano anterior

```
var parrafo = document.getElementById("p1");

var mensaje = document.getElementById("mensaje");

mensaje.innerHTML = "Anterior nodo hermano del elemento parrafo: "+ parrafo.previousSibling.nodeName;// Se utiliza la propiedad previousSibling del elemento parrafo con id "p1" para obtener el anterior nodo hermano con su respectivo nombre para mostrarlo en el elemento p con id mensaje.
```

Trae un nodo hermano siguiente

```
var parrafo = document.getElementById("p1");

var mensaje = document.getElementById("mensaje");

mensaje.innerHTML = "Siguiente nodo hermano del elemento parrafo: "+ parrafo.nextSibling.nodeName;// Se utiliza la propiedad nextSibling del elemento parrafo con id "p1" para obtener el siguiente nodo hermano con su respectivo nombre con el fil
```

Trae el primer hijo de un elemento

```
var div = document.getElementById("miDiv");

var mensaje = document.getElementById("mensaje");

mensaje.innerHTML = "Primer nodo del div: "+div.firstChild.nodeName;// Se utiliza la propiedad firstChild del div para obtener el primer nodo hijo con su respectivo nombre para mostrar el elemento p con id mensaje.
```

Trae el nodo padre de un elemento

```
var encabezado = document.getElementsByTagName("h1")[0];//Obtenemos el primero elemento h1 a traves del metodo getElementsByTagName.

document.getElementById("mensaje").innerHTML = encabezado.parentNode.nodeName; // Obtenemos el nombre del nodo padre o elemento padre del elemento encabezado obtenido y los mostramos con id mensaje.
```

**BOM: trabaja con el objeto *window*. Permite manipular y abrir nuevas ventanas a partir de una pagina**

```

<button onclick="abrirVentana()">Abrir Ventana</button>
<button onclick="moverVentana()">Mover Ventana</button>
<button onclick="cambiarDimVentana()">Cambiar dimension Ventana.</button>
<button onclick="verInfoPantalla()">Info Pantalla</button>

<p id="infoPantalla"></p>

<script>
var ventanal1;

function abrirVentana() {
    ventanal1 = window.open("", "_blank", "width=500, height=400");
}

function cambiarDimVentana() {
    ventanal1.resizeTo(800, 800);
    ventanal1.focus();
}

function verInfoPantalla() {
    var info = "";
    info += 'INFORMACION DE PANTALLA<br>';
    info += 'ALTO PERMITIDO : ' + screen.availHeight + '<br>';
    info += 'ANCHO PERMITIDO : ' + screen.availWidth + '<br>';
    info += 'ALTURA TOTAL : ' + screen.height + '<br>';
    info += 'ANCHO TOTAL : ' + screen.width + '<br>';
    info += 'NUMEROS BITS DE COLORES : ' + screen.colorDepth + '<br>';

    document.getElementById("infoPantalla").innerHTML = info;
}

function abrirVentana4(){
    ventana4 = window.open("http://www.youtube.com/","ventana4","location=1");
    ventana4.focus();
}

function cerrarVentana1(){
    ventanal1.close();
}

```

```

var propiedad=location.assign("http://www.google.com");
alert("Se redireccionara a la URL");

var propiedad=location.pathname;
alert("La ruta del sitio actual es: "+propiedad);

var propiedad=location.protocol;
alert("El protocolo de la URL es: "+propiedad);
|

```

BOM :Uso del prompt: permite pedir un dato al usuario con un PopUp

```

var Edad = prompt("Digite su Edad");

if (Edad != null) {
    document.getElementById("mensaje").innerHTML =
    "Usted Tiene " + Edad + " Años";
}

```

BOM : Uso del confirm: permite confirmar un dato al usuario con un PopUp

```

function confirmar(){
    if(confirm("Desea Ingresar?")==true){
        preguntas();
    }else{
        window.close();
    }
}

```

BOM : Navigator: Permite saber el nombre y demás propiedades del navegador actual.

```

function navegador() {

    document.getElementById("mensaje").innerHTML =
    "Nombre: " + navigator.appName + "<br>" +
    "Versión: " + navigator.appVersion + "<br>" +
    "Lenguaje: " + navigator.language + "<br>" +
    "Navegador esta en linea: " + navigator.onLine + "<br>" +
    "Java esta Habilitado: " + navigator.javaEnabled();

}

```

```

mensaje+="Cookies Habilitadas:" + navigator.cookieEnabled + "<br>";
mensaje+="Plataforma: " + navigator.platform + "<br>";
mensaje+="El nombre Motor del Navegador:" + navigator.product + "<br>";
mensaje+="Tipos MIME soportados:" + "<br>";

for(var i=0;i<navigator.mimeTypes.length;i++){
    mensaje+= "-"+navigator.mimeTypes[i].type + "<br>";
}
mensaje+="Plugins:" + "<br>";

for(var j=0; j<navigator.plugins.length;i++){
    mensaje+=navigator.plugins[j].name + "<br>";
}
document.getElementById("mensaje").innerHTML=mensaje;

```

BOM: History: Permite ir a las páginas visitadas en la navegación

```

window.history.back();   atras
window.history.forward(); adelante
window.history.go(2);   ir a especifica

```

BOM: Eventos de Tiempo(timers)

```

var inval;
function usarSetInterval() {
    inval = setInterval(function() {
        alert("Usando setInterval!!!");
    }, 3000);
}

function pararSetInterval() {
    clearInterval(inval)
    alert("setInterval detenido!!!")
}

function usarsetTimeout() {
    setTimeout(function() {
        alert("Usando setTimeout!!!");
    }, 3000);
}

```

Ejemplos eventos tiempo:

```

<script type="text/javascript">
var indice=0;
var funcionComenzar = null;

function comenzar(){
    var listaImagenes = ['imagen1.png','imagen2.png','imagen3.png','imagen4.png'];
    if(indice < 3){
        document.getElementById('miImagen').src = listaImagenes[indice];
    }else{
        indice=0;
        document.getElementById('miImagen').src = listaImagenes[indice];
    }
    indice += 1;
    funcionComenzar = setTimeout(function(){
        comenzar(indice)
    },2000);
}

function terminar(){
    clearTimeout(funcionComenzar);
}

</script>

```

```

<div id="miReloj">
  00:00:00
</div>
<br>

<button onclick="iniciarReloj()">Iniciar Reloj</button>
<button onclick="pararReloj()">Parar Reloj</button>

<script type="text/javascript">
var funcionReloj;
function iniciarReloj(){
  var reloj = new Date();
  var horas = reloj.getHours();
  var minutos = reloj.getMinutes();
  var segundos = reloj.getSeconds();

  if(horas<10){
    horas = "0" + horas;
  }

  if(minutos<10){
    minutos = "0" + minutos;
  }

  if(segundos<10){
    segundos = "0"+segundos;
  }
  document.getElementById('miReloj').innerHTML=horas+":"+minutos+":"+segundos;
  funcionReloj = setTimeout(iniciarReloj,1000);
}

```



## JQUERY

jQuery es un framework, o librería JavaScript, para el desarrollo de sitios web que comprende una gran cantidad de características que permiten al desarrollador implementar elementos de JavaScript fácil y rápidamente. Además de ser la librería de JavaScript más utilizada en la actualidad es también un proyecto de software libre; lo que significa que su uso es completamente gratuito. jQuery fue lanzado en el 2006 por John Resig y desde entonces ha ido evolucionando hasta su versión más reciente: jQuery 3.0, lanzada el 9 de Junio de 2016.

Esta librería se encarga principalmente de ejecutar las acciones que se pueden obtener con JavaScript pero con una reducción de código significativa. Una de las principales tareas de este framework es encargarse de la compatibilidad con todos los navegadores, ya que esta labor sin uso de jQuery requiere una gran cantidad de código. Además, jQuery permite la manipulación de los elementos del DOM con gran facilidad gracias a su motor de selección Sizzle; una manera sencilla de capturar eventos, una gran variedad de animaciones prediseñadas, y de un tamaño muy pequeño que demanda poco espacio en la memoria al momento de incluirlo en un proyecto web.

Cabe anotar que jQuery no es la única librería JavaScript disponible. Sin embargo, es la preferida por los desarrolladores web debido a su estabilidad, buena documentación y al soporte de toda una empresa velando por su mantenimiento y actualización hacia nuevas tecnologías. Otro de los puntos a favor de esta librería se basa en su popularidad, ya que en una comunidad tan grande de usuarios hay muchas personas realizando plantillas, componentes y plug-ins de jQuery disponibles para su uso.

El primer elemento a tener en cuenta es el signo pesos «\$». Este es uno de los caracteres más importantes ya que indica el acceso a la librería de jQuery. O sea, define lo que viene a continuación entre paréntesis como un objeto jQuery que puede hacer un llamado a todas las funcionalidades del framework.

El segundo elemento es el selector, el cual se ubica entre los paréntesis después del signo pesos. Este se encarga de indicar sobre qué elemento del DOM se aplicará la característica de jQuery.

El tercer elemento es el punto «.», que indica la invocación de un método propio de jQuery. Este método se especifica en el cuarto elemento que corresponde a la acción; es decir, a qué efecto se aplicará al elemento seleccionado previamente.

Dentro de los paréntesis de la acción puede ir una función de JavaScript definiendo más acciones en cadena. Así, dentro de una sentencia de jQuery puede ir otra en su interior. Según lo anterior, es muy común ver una estructura como la siguiente:

```

$(selector).accion(function() {
  $(selector).accion();
});

```

**Objetos literales:** es una variable que funciona como objeto con propiedades, métodos y constructor.

```

//Objeto literal llamado empresa
var empresa ={
    nit:null,
    nombre:null,
    direccion:null,
    email:null,
    paginaWeb:null,
    horarios:null,
    init:function(propiedades){
        empresa.propiedades = propiedades;
    },
    leerPropiedades : function(){
        return empresa.propiedades;
    }
};


```

#### Asignar o leer atributos de un elemento

✓ `$( "p" ).attr({ "title": "Parrafo" } );`

`$(selector).attr("propiedad", función( ){ });`

#### Manipulación de Elementos

- ✓ `.html():` Obtener o establecer el contenido HTML.
- ✓ `.text():` Obtener o establecer el contenido del texto.
- ✓ `.attr():` Obtener o establecer el valor del atributo proporcionado.
- ✓ `.width():` Obtener o establecer la anchura en pixeles
- ✓ `.height():` Obtener o establecer la altura en pixeles.
- ✓ `.position ():` Obtener un objeto con información de posición para el primer elemento en la selección.
- ✓ `.val():` Obtener o establecer el valor de los elementos del formularios.

#### Modificación de contenido

<code>text()</code>	<code>prepend()</code>
<code>html()</code>	<code>appendTo()</code>
<code>append()</code>	<code>prependTo()</code>

Con `text()` no se renderizan las etiquetas html que se incluyan en el texto, mientras que con `html()` Si.

Con `append()` agrega texto o etiquetas después del contenido

Con `prepend()` agrega texto o etiquetas antes del contenido

`append() y prepend():`

`appendTo() y prependTo():`

`$(selección).append("nuevoContenido")` `nuevoContenido.appendTo($(selección))`

#### Insertar nuevo elemento html

After agrega contenido después del elemento indicado

`$(elemento).after (nuevoElemento);`

after → <div> Texto </div> Contenido nuevo  
append → <div> Texto Contenido nuevo </div>

```
$(".azul").after("<div class='circulo amarillo'>Nuevo</div>")
$(".azul").append($(".rojo"))
$(".verde").append(function(){
  var elemento = $(".logo").html()
  return elemento
})
$(“elemento”).after(“nuevo Contenido”)
$(“nuevo Contenido”).insertAfter($(“elemento”))
```

before()

```
$('.azul').before("<div class='circulo amarillo'>Nuevo</div>")
$('.azul').before(function(){
  var palabra = " Hola!"
  return |
})
```

Agregar clases de estilos a un elemento

```
var p1 = $( "p" );
p1.addClass( "miEstilo" );
```

Verifica si existe en dato en un arreglo

✓ `$.inArray()` : Verifica si un elemento esta en un arreglo Ej:

```
var arreglo1 = [ 'a', , 'b', 'c', 'd', 'e' ];
if ( $.inArray( 'b', arreglo1 ) !== -1 ) {
  alert("Encotrado! "+$.inArray( 'b', arreglo1 ));
```

Recorre un arreglo

```
var arr= ["Carlos", "Juan", "Maria" ];
$.each(arr, function( indice, valor ) {
  alert( "Indice es: " + idx + " su valor es: " + valor );
});
```

Trae el valor de un `checked` de varias opciones

```
$(“input[name=radio-genero]:checked”).val();
```

Eliminar elementos

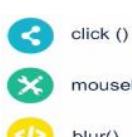
`.remove()` : elimina el elemento indicado junto con su contenido  
`.detach()` : elimina el elemento indicado pero no eliminar su contenido pudiendo ser reutilizado  
`.empty()` : vacia el contenido de un elemento borrando todo el contenido  
`.unwrap()` : elimina su parent directo y ocupa su lugar subiendo un nivel

Reemplazar elementos

```
elementoViejo.replaceWith(elementoNuevo)
```

```
$("#c1").find("span").replaceWith("<i class='material-icons medium'>thumb_up</i>")
$("#c3").replaceWith($("#c8"));
```

Eventos JQUERY



## Eventos nativos

```
$(window).resize(function(){
  $('#h2').html("El ancho de la página es: "+ $(window).width())
})

$(window).scroll(function(){
  $("h4").html("La posición vertical es: "+ $(window).scrollTop())
})
```

## Evento mouseenter():

El evento se ejecuta cuando pasamos el cursor del mouse por algún elemento de la página al cual esté configurado con este evento.

### Ejemplo:

JS:	HTML:
<pre>\$("#b_mouseenter").mouseenter(function(){   alert("El mouse está sobre el botón!") });</pre>	<pre>&lt;input type="button" value="Mouseenter" id="b_mouseenter"&gt;</pre>

## Evento focus():

Se ejecuta cuando se está haciendo foco en un campo de un formulario.

### Ejemplo:

JS:	HTML:
<pre>\$("#t_focus").focus(function(){   \$(this).css("background-color", "#848484"); });</pre>	<pre>&lt;input type="text" id="t_focus"&gt;</pre>

## Evento blur():

Se ejecuta cuando el foco ya no está posicionado en un campo del formulario.

### Ejemplo:

JS:	HTML:
<pre>\$("#t_focus").blur(function(){   \$(this).css("background-color", "#ffffff"); });</pre>	<pre>&lt;input type="text" id="t_focus"&gt;</pre>

## Evento dblclick():

Se ejecuta cuando hacemos doble clic sobre algún elemento de nuestra página.

### Ejemplo:

JS:	HTML:
<pre>\$("#b_dblclick").dblclick(function(){   \$("#p_dblclick").append("Texto nuevo. "); });</pre>	<pre>&lt;input type="button" value="Dblclick" id="b_dblclick"&gt; &lt;p id="p_dblclick"&gt;&lt;/p&gt;</pre>

## Evento Hover: permite activar un efecto en una lista al pasar por encima

```
$(document).ready(function(){
  $("li").filter(":odd").hide().end().filter(":even").hover(function(){
    $(this).toggleClass("active").next().stop(true, true).slideToggle();
  });
});
```

## Evento Change(): al cambiar un valor de un elemento hace algo

```

$(document).ready(function(){
    $("select").change(function(){
        var fruta = "";
        $("select option:selected").each(function(){
            fruta += $(this).text() + " ";
            alert("Tu fruta preferida es: "+fruta);
        });
    }).change();
});

```

Mas eventos: <https://developer.mozilla.org/en-US/docs/Web/Events>

**hide("slow")** Esconde el elemento.

**fadeIn("slow")** Muestra un elemento oculto.

**slideUp("slow")** Desliza hacia arriba el elemento.

**Método .focusin()**: permite hacer algo al entrar a un elemento

```

$("p").focusin(function(){
    $(this).find("span").css("display","inline").fadeOut(2000);
    $(this).find("span").css("color","red");
});

$("#boton").click(function(){
    alert("Informacion Guardada");
});

```

**Manejador de eventos:**

```

//eventos simultaneos a un control
$('p').on({ 'click': function() { console.log('clickeado'); },
           'mouseover': function() { console.log('sobrepasado'); },
           'hover': function() { console.log('hoverado'); } });

```

**Permite**

- ✓ Enlazar cualquier evento activado en los elementos seleccionados.
- ✓ Realizar múltiples eventos a un solo manejador de eventos.
- ✓ Realizar Múltiples eventos a un elemento seleccionado.
- ✓ Usa detalles sobre el evento.
- ✓ Pasar datos al controlador para eventos personalizados.
- ✓ Unión de eventos a los elementos no existentes.

**Disparadores:** permite disparar un evento directamente o saber cuando un evento se ha disparado

## Disparadores de Gestores de Eventos

Jquery proporciona una manera de activar los controladores de eventos unidos a un elemento sin ninguna interacción del usuario a través del método .trigger().

- ✓ \$( "input" ).trigger ( "click" );
- ✓ \$( "input" ).simulate ( "click" );

**Objetos de un evento:** se usa para obtener las propiedades y objetos de los eventos.

```
$(document).ready(function() {
    $(document).mousemove(function(event) {
        $("#parrafo").text("La posicion del mouse es: X: " + event.pageX + ", Y: " + event.pageY);
    });
    $("#enlace").click(function(event) {
        event.preventDefault();
    });
});
```

**Delegación:** con los manejadores permite agregar o Heredar eventos ya creados entre elementos  
**Delegación**



- ✓ Permite agregar eventos ya creados a nuevos elementos.
- ✓ El método utilizado es **delegate()**.
- ✓ Sintaxis básica:

```
$(nodo_padre).delegate(nodo_hijo, evento, función(){
    // Acciones de la función
});
```

Ej: a un <p> que ya existe se le agrega un nuevo <p> con el evento click el cual tendrá el mismo evento.

También se puede undelegar usando el mismo proceso mas una nueva función.

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

    $("body").delegate("p", "click",function() {
        $("#parrafo_nuevo").append(" Este parrafo es nuevo y tiene el mismo evento. ");
    });

    $("#undelegate").click(function() {
        $("body").undelegate();
    })
});
```

**Eventos personalizados:** se usa con .on() y .trigger() para crear eventos personales

```
$(document).ready(function(){
    $( document ).on( "Frutas", {
        fruta: "Fresa"
    }, function( event, arg1, arg2 ) {
        $("#mensaje").html(event.data.fruta + "<br>" + arg1 + "<br>" + arg2);
    });
    $( document ).trigger( "Frutas", [ "Manzana", "Pera" ] );
});
```

## Efectos Jquery

### Efecto show()

Muestra los elementos seleccionados ocultos.

**Sintaxis básica:**

```
$elemento.show(velocidad,funcion)
```

**Ejemplo:**

JS:

```
$("#show").click(function(){
  $("#parrafo").show(5000, function(){
    alert("El párrafo ya no está oculto.");
  });
});
```

HTML:

```
<p id="parrafo">Soy un párrafo.</p>
<input type="button" id="hide" value="Hide">
<input type="button" id="show" value="Show">
```

### Efecto stop()

Detiene la animación que se está ejecutando en un determinado elemento.

**Sintaxis básica:**

```
$elemento.stop(pararTodo,pararDeInmediatoTodo)
```

JS:

```
$("#star").click(function(){
  $("#caja2").animate({"width":"2000px"},10000);
});
$("#stop").click(function(){
  $("#caja2").stop();
});
```

HTML:

```
<div id="caja2"></div><br><br>
<input type="button" id="star" value="Aumentar tamaño">
<input type="button" id="stop" value="Para aumento">
```

Mas efectos: <http://api.jquery.com/category/effects/>

### Efecto animate()

Aplica animaciones a un elemento de la página, a través de CSS.

**Sintaxis básica:**

```
$elemento.animate({estilo}, velocidad, función)
```

**Ejemplo:**

JS:

```
$("#animate").click(function(){
  $("#caja").animate({"left":"+=50px",
  "width":"+=5px"});
});
```

HTML:

```
<div id="caja"></div><br><br>
<input type="button" id="animate" value="Mover">
```

### Efecto hide()

Oculta los elementos seleccionados.

**Sintaxis básica:**

```
$elemento.hide(velocidad,funcion)
```

**Ejemplo:**

JS:

```
$("#hide").click(function(){
  $("#parrafo").hide(5000);
});
```

HTML:

```
<p id="parrafo">Soy un párrafo.</p>
<input type="button" id="hide" value="Hide">
```

```
$(“selector”).show(‘slow’); lento
$(“selector”).show(‘fast’); rápido
$(“selector”).show(4000); 4 seg
$(“selector”).show(‘slow’,function() { //Acción})
$(“selector”).hide(2000,’linear’,function(){ })
```

**Selector.animate(**  
  {propiedades CSS},  
  duración ,  
  efecto,  
  callback  
  )

```
$(this).animate(
  {
    top: "300px"
  }, 1000, function(){
    alert("Animación completada");
  });
});
```

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

$("#show").click(function() {
  $("#parrafo").show(5000, function() {
    alert("El párrafo ya no está oculto.");
  });
});

$("#animate").click(function() {
  $("#caja").animate({"left":"+=50px", "width":"+=5px"});
});

$("#start").click(function() {
  $("#caja2").animate({"width":"2000px"}, 10000);
});

$("#stop").click(function() {
  $("#caja2").stop();
});
```

\$(“div”).slideToggle();

Altera entre deslizar hacia arriba o hacia abajo un elemento según el evento.

\$(“div”).fadeToggle();

Altera entre mostrar y ocultar un elemento difuminado según el evento.

\$(“div”).slideUp();

Oculta un elemento deslizándolo hacia arriba.

\$(“div”).fadeOut();

Oculta un elemento con un efecto de difuminado.

\$(“div”).fadeIn();

Muestra un elemento en pantalla con un efecto de difuminado.

### Efecto toggle: oculta o muestra un elemento

.toggle('slow', 'swing', function(){})

```
$(document).ready(function(){
  $("button").click(function(){
    $("div").toggle("slow");
  });
});
```

slideUp()

slideToggle()

### Efecto fadeIn y fadeout: desliza ocultando un elemento y mostrándolo nuevamente

```
$(document).ready(function(){
  $("button").click(function(){
    $("div").fadeIn(4000, function(){
      $("label").fadeIn(1000);
      $("p").fadeOut(10);
    });
    return false;
  });
});
```

fadeIn('slow')

fadeIn('fast')

fadeIn(3000)

### Queue-dequeue: cola de eventos, permite hacer eventos en cola secuencialmente

```

<script type="text/javascript">
$(document).ready(function() {
    $("#boton1").click(function() {
        colaEfectos();
    });
    function colaEfectos() {
        var miDiv = $("#miDiv");
        miDiv.queue(function() {
            $(this).css({
                "background-color": "blue",
            });
            $("#mensajeFinal").html("Se cambio el color de fondo a azul");
        });
        miDiv.hide(1000);
        miDiv.show(1000);
        miDiv.fadeIn(3000);
        miDiv.fadeOut(3000);
        miDiv.show(1000);
    };
}

```



## Plugins: son js predefinidos para realizar funcionalidades específicas, se pueden descargar de [plugins-jquery.com](http://plugins-jquery.com)

Agregar plugin a la página:

```
<script type="text/javascript" src="jquery-1.11.1.js"></script>
<script type="text/javascript" src="jquery.validate.js"></script>
```

Implementar plugin de validación:

```
<script type="text/javascript">
jQuery(document).ready(function($){
    $("#formulario").validate();
});
</script>
```

### Ej: validar un inicio de sesión con un plugin

```

<script src="jquery.validate.js"></script>
<script type="text/javascript">
$.validator.setDefaults({
    submitHandler: function(){
        alert("Inicio de Sesión Exitoso");
        $("#usuario").val("");
        $("#password").val("");
    }
});

$(document).ready(function(){
    var validator = $("#miForm").validate({
        errorPlacement: function(error,element){
            $(element).closest("form")
            .find("label[for='"+element.attr("id")+"']")
            .append(error);
        },
        errorElement: "span",
        messages:{
            usuario: {
                required: "(Por Favor Ingrese Su Nombre de Usuario)",
                minlength: "(Debe tener mínimo 3 caracteres)"
            },
            password: {
                required: "(Por Favor Ingrese Su Contraseña)",
                minlength: "(Debe Tener entre 5 y 12 caracteres",
                maxlength: "(Debe tener entre 5 y 12 caracteres)"
            }
        }
    });
})

```



## ¿Cómo crear plugin?

### Sintaxis básica:

#### Primera forma:

```
(function($){
    // Aquí van las acciones de la función.
})(jQuery)
```

#### Segunda forma:

```
jQuery.fn.nombre_plugin = function() {
    // Aquí van las acciones de la función.
};
```

## Ejemplo:

### JS

```

jQuery.fn.plugin_hide = function(){
    this.each(function(){
        elementos = $(this),
        elemento.hide(1000);
    });
};

$(document).ready(function(){
    $("#eliminar").click(function(){
        $("#parrafo").plugin_hide();
    })
})

```

### HTML

```

<p id="parrafo">Este párrafo se eliminará mediante un plugin</p>
<p><input type="button" value="Eliminar párrafo" id="eliminar"></p>

```

## Reglas para crear plugins

- ✓ Debemos guardar el plugin de esta forma jquery.nombredelplugin.js.
- ✓ Debemos utilizar la propiedad fn del objeto jQuery.
- ✓ this lo podemos utilizar para acceder a cualquier propiedad de un elemento.
- ✓ Se debe utilizar ";" al finalizar cada método.
- ✓ Se debe retornar el objeto sobre el cual se hizo la ejecución del plugin.
- ✓ Utilizar el método each para iterar entre los elementos.
- ✓ Se debe asignar siempre el plugin al objeto de jQuery.

## Métodos utilitarios

### Método data()



- ✓ La primera función es almacenar datos en un elemento.
  - ✓ Para almacenar datos el método recibe dos parámetros.
  - ✓ El primer parámetro es el nombre del dato a guardar.
  - ✓ El segundo parámetro es el valor del dato a guardar.
  - ✓ Ej:
- ```
$("#miDiv").data("dato1","valorDato1");
```

### Método grep()

- ✓ El método grep() permite filtrar elementos de un arreglo según una condición.
- ✓ Sintaxis:
 

```
$.grep(arreglo, function(valor,index), invertir)
```
- ✓ El primer parámetro es el arreglo a filtrar.
- ✓ El Segundo es una función que se encarga de realizar el filtro.

### Método isEmptyObject()

- ✓ Verifica si un objeto si esta vacío.
- ✓ Recibe un solo parámetro que es el objeto a evaluar.
- ✓ Si el objeto no tiene propiedades retornará true y si las tiene, false.
- ✓ Ej:

```
$.isEmptyObject({}); // True
$.isEmptyObject({nombre : "Juan"}); //False
```



### Método isNumeric()

- ✓ Verifica si un valor es numérico o no.
- ✓ Recibe un solo parámetro que es el valor a evaluar.
- ✓ Devuelve true si es un valor numérico o false si no lo es.
- ✓ Ej:
 

```
$.isNumeric(4.434)// true
$.isNumeric("hola");// false
```

## Método merge()

- ✓ Permite copiar los elementos de un arreglo e insertarlos en otro.
- ✓ Recibe dos parámetros.
- ✓ El primero es arreglo al cual se le van a insertar los elementos.
- ✓ El segundo es el arreglo del cual se copiarán sus elementos para insertarlos en el primer array.

✓ Ej:

```
var array1 = [1, 2, 3, 4];
var array2 = [4, 5, 6];
$.merge(array1, array2);
```

## Método .find()

```
$document.ready(function(){
  var base_datos = "<xml version='1.0'><persona> <nombres>Pedro</nombres> <apellidos>Perez</apellidos> <person
  var $xml = $(base_datos);

  var $nombres = $xml.find('nombres');
  var $apellidos = $xml.find('apellidos');

  $('#nombres').html($nombres);
  $('#apellidos').html($apellidos);
});
```

\$select("seleccion").find("sub-selección")

```
$(".col").find("[a[href^='https:']]").css("color", "yellow")
```

## Obtener o establecer valores

Para check: devuelve el valor seleccionado

\$(["name=opciones"]:checked).val();

Asignar el valor a varias opciones de un multiple

```
$(".checkbox").val(["casilla1", "casilla3", "casilla5"]);
```

Ej: agregar y eliminar datos de un select

```
$('#agregarcarro').click(function(event){
  event.preventDefault();
  var nombrecarro = $('#nombre').val();
  var valorcarro = $('#valor').val();
  $('').attr('value', valorcarro).text(nombrecarro).appendTo('#carros');
});

$('#quitarcarro').click(function(event) {
  event.preventDefault();
  var $select = $('#carros');
  $('option:selected', $select).remove();
});
```

## Estilos

```
$(selector).css("propiedad", "valor");    $(selector).css(["propiedad1", "propiedad2"]);
```

```
$(selector).css("propiedad", función(){ });
$(selector).addClass("nuevaClase");
```

```
$(this).css("height", function(index, value){
  return parseFloat(value)*1.2;
});
$('span').css('heidth', '30px');
```

## Método parseJSON()

- ✓ Permite convertir un JSON en formato cadena de texto a un objeto JavaScript.
- ✓ Recibe un solo parámetro que es el JSON en formato cadena de texto a convertir.
- ✓ El JSON en formato cadena de texto debe estar bien formado.

✓ Ej:

```
var objJS = $.parseJSON('{"Nombre": "Juan"}');
```

## LocalStorage: técnica que almacena datos en el navegador del us(ccokies)

- ✓ Técnica de almacenamiento que ofrece HTML5.
- ✓ Permite almacenar datos en un navegador web.

### localStorage.setItem()

- ✓ Permite almacenar un dato con su respectiva clave y valor.
- ✓ Recibe dos parámetros.
- ✓ El primer parámetro es la clave o identificador del dato.
- ✓ El segundo parámetro es el valor del dato.
- ✓ Ej:

```
localStorage.setItem('nombre', 'Pedro');
```

### localStorage.getItem()

- ✓ Permite obtener el valor de un dato, a través de una clave asociada a este.
- ✓ Recibe un solo parámetro que es la clave asociada al dato.
- ✓ Ej:

```
var valor = localStorage.getItem('objeto1');
```

### localStorage.removeItem()

- ✓ Permite eliminar un dato del localStorage.
- ✓ Recibe un solo parámetro que es la clave del dato a eliminar.
- ✓ Ej:

```
localStorage.removeItem('clave');
```

Como es solo texto, existe JSON.stringify para convertir a texto

```
var objeto1= {nombre: "Juan", apellido: "Perez"};  
localStorage.setItem('objeto1', JSON.stringify(objeto1));
```

## JQUERY UI

Con librería: <http://jqueryui.com/download/>



Accordion



Dialog



Autocomplete



Slider



Button



Tabs



Datepicker



Progressbar

### Efectos



- ✓ Core
- ✓ Blind
- ✓ Bounce
- ✓ Clip
- ✓ Drop
- ✓ Explode
- ✓ Fade
- ✓ Fold
- ✓ Highlight

```
<link rel="stylesheet" href="jquery-ui.min.css">
<script src="external/jquery/jquery.js"></script>
<script src="jquery-ui.min.js"></script>
```

```
$(".panel-imagen")
    .on("click", function(){
        $(".panel-imagen p").addClass("texto-1", 2000)
        $(".panel-imagen").toggleClass("panel-2", 200)
    })
    .on("mouseover", function(){
        $(".texto-1").switchClass("texto-1","texto-2", 1000)
        $(".texto-2").switchClass("texto-2","texto-1", 1000)
    })

    $("#fold").on("click", function(){
        $(".panel-imagen").hide("fold",1000)
    })
    $("#bounce").on("click", function(){
        $(".panel-imagen").show("bounce", 1000)
    })
    $("#explode").on("click", function(){
        $(".panel-imagen").toggle("explode", 1000)
    })
    $("#shake").on("click", function(){
        $(".panel-imagen").effect("shake",10)
    })
})
```

## Draggable(): al arrastrar un elemento

### Opción de arrastrar:

```
$("#miDiv").draggable({
    cursor: 'pointer'
});
```

Valores de la opción cursor	✓ text
✓ default	✓ w-resize
✓ help	✓ wait
✓ nw-resize	✓ se-resize
✓ pointer	✓ move
✓ s-resize	✓ n-resize
✓ crosshair	✓ ne-resize
✓ e-resize	✓ sw-resize
✓ hand	

### Eventos del Draggable()

#### Evento create

- ✓ Este evento se dispara cuando se agrega la funcionalidad arrastrar a través del método draggable. Tiene dos formas de implementarlo.
- ✓ La primera forma es definiéndolo cuando invocamos al método draggable().
- ✓ Ej:

```
$("#miDiv").draggable({
    create: function(event, ui) {
        $("#mensaje").html("Evento create!");
    }
});
```

#### Evento start

- ✓ Este evento se dispara cuando se inicia la funcionalidad arrastrar. Tiene dos formas de implementarlo.
- ✓ La primera forma es definiéndolo cuando invocamos al método draggable().
- ✓ Ej:

```
$("#miDiv").draggable({
    start: function(event, ui) {
        $("#mensaje").html("Evento start!");
    }
});
```

#### Evento drag

- ✓ Este evento se dispara cuando se arrastra un elemento usando el mouse. Tiene dos formas de implementarlo.
- ✓ La primera forma es definiéndolo cuando invocamos al método draggable().
- ✓ Ej:

```
$("#miDiv").draggable({
    drag: function(event, ui) {
        $("#mensaje").html("Evento drag!");
    }
});
```

#### Evento stop

- ✓ La otra forma es de implementarlo agregando un event listener a través del método on().
- ✓ Ej:

```
$("#miDiv").on("stop", function(event, ui) {
    $("#mensaje").html("Evento stop!");
});
```

### Ejemplo:

```

$("#miDiv").draggable({
  create: function(evento, ui) {
    $("#mensaje").html(mensaje += "Evento create!<br>");
  }
});

$("#miDiv").on("dragstart", function(evento, ui) {
  $("#mensaje").html(mensaje += "Evento start!<br>");
})

$("#miDiv").on("drag", function(evento, ui) {
  $("#mensaje").html(mensaje += "Evento drag!<br>");
})

$("#miDiv").on("dragstop", function(evento, ui) {
  $("#mensaje").html(mensaje += "Evento stop!<br>");
})

```

## Hacer un espejo de un elemento al hacer drag

```

$(document).ready(function(){
  $('#miDiv1').draggable({
    cursor: 'move',
    containment : 'document',
    helper: miCopia
  });

  function miCopia(event){
    return '<div id="miDiv2">Un Clon Modificado!!!</div>';
  }
});

```

## Droppable(): al soltar el elemento arrastrado

```

$("#elemento").droppable();

```

Droppable: eventos, métodos, opciones.

tolerance:
accept:
out:
disable:

### Ejemplo:

```

$(document).ready(function() {
  $("#arrastrar").draggable();
  $("#soltar").droppable({
    drop: function() {
      alert("¡Has soltado el elemento en la caja!");
    }
  });
});

```

## Resizable(): para cambiar el tamaño de algún elemento

### Opciones de resizable

- ✓ alsoResize
- ✓ animate
- ✓ animateDuration
- ✓ animateEasing
- ✓ aspectRatio
- ✓ autoHide
- ✓ cancel
- ✓ containment
- ✓ delay
- ✓ disabled
- ✓ distance
- ✓ ghost
- ✓ grid
- ✓ handles
- ✓ helper
- ✓ maxHeight
- ✓ maxWidth
- ✓ minHeight
- ✓ minWidth

`$(selector).resizable (options)`

`$(selector).resizable ({option:valor1,option2:valor2})`

`$(selector).resizable ("action", params)`

**Selectable() y Sortable():** seleccionar y organizar elementos en listas o contenedores arrastrándolas. Se debe usar css para diferenciar las selecciones

`$( "elemento" ).selectable();`

### Selectable: eventos, métodos, opciones.

distance:

selected:

Ejemplo:

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

var eventos = "";
$("#productos").selectable({
    selected: function(event,ui){
        var seleccionados = $("li[class='ui-selected']").length;
        $("#itemSelects").html("Has seleccionado "+ seleccionados+" productos!!!");
        $("#mensaje").html(eventos += "Evento selected!!!<br>");
    },
    unselected: function(e,ui){
        $("#eventos").html(eventos += "Evento unselected!!<br>");
    },
    start:function(e){
        $("#eventos").html(eventos += "Evento start!!<br>");
    },
    stop:function(){
        $("#eventos").html(eventos += "Evento stop!!<br>");
    }
});
});
```

```
<style type="text/css">
.miProducto{
    background-color: #E920B3;
}
#productos .ui-selected{
    color: red;
    background: #FFF415;
    font-weight: bold;
}
#productos{
    margin: 0;
    padding: 0;
    list-style-type: none;
}
</style>
</head>
<body>
<ul id="productos">
<li class="miProducto">Producto1</li>
<li class="miProducto">Producto2</li>
<li class="miProducto">Producto3</li>
<li class="miProducto">Producto4</li>
<li class="miProducto">Producto5</li>
<li class="miProducto">Producto6</li>
<li class="miProducto">Producto7</li>
<li class="miProducto">Producto8</li>
<li class="miProducto">Producto9</li>
</ul>
</body>
```

### Sortable: eventos, métodos, opciones.

placeholder:

connectWith:

Ejemplo:

```
var mensaje="";
$("#miLista1").sortable({
    start: function (event,ui) {
        $("#mensaje").html(mensaje += "Evento start desde lista 1!<br>");
    },
    receive: function(event, ui){
        $("#mensaje").html(mensaje += "Evento receive desde lista 1!<br>");
    },
    stop: function(event,ui){
        $("#mensaje").html(mensaje += "Evento Stop desde lista 1!<br>")
    }
});
```

```
$("#miLista1").sortable({
    connectWith: "#miLista1, #miLista2, #miLista3"
});

$("#miLista2").sortable({
    connectWith: "#miLista1, #miLista2, #miLista3"
});

$("#miLista3").sortable({
    connectWith: "#miLista1, #miLista2, #miLista3"
});
```

## Widgets

**Accordion():** Sección expandible(Acordeon) que debe tener cierta estructura

✓ La primera parte consiste en construir el HTML que representa al widget.

✓ Estructura:

```
<div id="miAccordion">
    <h3>Sección 1</h3>
    <div>Contenido 1...</div>
    <h3>Sección 2</h3>
    <div>Contenido 2...</div>
    <h3>Sección 3</h3>
    <div>Contenido 3...</div>
</div>
```

✓ La segunda parte consiste en la invocación del método accordion().

✓ Ejemplo:

```
$( "#miAccordion" ).accordion();
```

✓ El método accordion() posee una serie de opciones que permiten extender su funcionalidad.

✓ La lista completa de opciones las pude ver en este link:

<http://api.jqueryui.com/accordion/>

## Acordeón: header Acordeón: fillspace Acordeón: active

- |                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>✓ La opción header es el tipo de elemento encabezado que tendrá como título el acordeón.</li> <li>✓ El tipo de header puede ser desde h1 hasta h6.</li> <li>✓ Ej:</li> </ul> <pre>\$( "#miAccordion" ).accordion({   'header': 'h4', });</pre> | <ul style="list-style-type: none"> <li>✓ La opción fillSpace permite ajustar la altura del div donde se definió el widget Accordion.</li> <li>✓ El valor de la opción puede ser true o false.</li> <li>✓ Ej:</li> </ul> <pre>\$( "#miAccordion" ).accordion({   'fillSpace': 'true', });</pre> | <ul style="list-style-type: none"> <li>✓ La opción active permite definir qué sección estará activa.</li> <li>✓ El valor de la opción puede ser desde 0 en adelante.</li> <li>✓ Ej:</li> </ul> <pre>\$( "#miAccordion" ).accordion({   'active': '0', });</pre> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Autocomplete(): realiza busquedas en una caja de texto

<ul style="list-style-type: none"> <li>✓ La primera parte consiste en definir la caja de texto, a la cual se va a agregar el autocomplete.</li> <li>✓ Ej:</li> </ul> <pre>&lt;input type="text" id="cjatxt1"&gt;</pre>	<ul style="list-style-type: none"> <li>✓ La segunda parte consiste en crear un arreglo, donde se almacenarán valores</li> <li>✓ Dichos valores se mostrarán cada vez que el usuario digite cadena de textos que coincidan.</li> <li>✓ Ej:</li> </ul> <pre>var valores = [   "Camerún",   "Colombia",   "Chile",   "Croacia",   "Chipre" ];</pre>	<ul style="list-style-type: none"> <li>✓ La tercera parte consiste en invocar al método autocomplete().</li> <li>✓ La opción source permitirá cargar el arreglo que se creó con los respectivos valores.</li> <li>✓ Ej:</li> </ul> <pre>\$("#cjatxt1").autocomplete({   source: valores })</pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Un accordion con sortable()

```
$(function(){
  $("#mi_accordion").accordion({
    collapsible: true
  });
});

$(function(){
  var icono = {
    header: "ui-icon-circle-arrow-e",
    activeHeader: "ui-icon-circle-arrow-s"
  };
  $("#mi_accordion").accordion({
    icons: icono
  });
});
```

## Button(): elementos de botón

### Opciones de button()



`$(Selector).button (opciones)`



`$(Selector).button ("acción", params)`

## Dialog(): cuadros de información

```
$(function() {
  $("button, p, h1")
    .button()
    .click(function( event ) {
      event.preventDefault();
    });
  $("#carros").menu();
  $("#cuadro").dialog();
});
```

## Opciones de dialog()

- ✓ appendTo
- ✓ autoOpen
- ✓ buttons
- ✓ closeOnEscape
- ✓ closeText
- ✓ dialogClass
- ✓ draggable
- ✓ height
- ✓ hide
- ✓ maxHeight
- ✓ maxWidth
- ✓ minHeight
- ✓ minWidth
- ✓ modal
- ✓ position
- ✓ resizable
- ✓ show
- ✓ title
- ✓ width



`$(Selector).dialog(opciones)`



`$(Selector).dialog("acción", params)`

## Menu() menu con submenus

### Opciones de menu()



- ✓ Disabled: Esta opción si es true, deshabilita el menú. Por defecto su valor es falso.
- ✓ Icons: Esta opción establece los iconos para los submenús.
- ✓ Menus: Esta opción es un selector para los elementos que sirven como el contenedor de menú.
- ✓ Position: Esta opción establece la posición de submenús en relación con el elemento del menú principal asociado.
- ✓ Role: Esta opción es utilizada para personalizar los roles.



`$(Selector).menu(opciones)`



`$(Selector).menu("acción", params)`

## Dialog() desde un button()

```
$(function() {
    $("#dialog").dialog({
        autoOpen: false
    });

    $("#boton").button().click(function(event){
        $("#dialog").dialog("open");
    });
});
```

```
$(".emergente").dialog({
    resizable: false,
    height:"auto",
    width: 400,
    modal: true,
    buttons: {
        "Delete all items": function() {
    }
})
```

## Selector Datepicker(): permite seleccionar fechas

```
$(document).ready(function() {
    $("#fecha").datepicker();
});
```

`$( "elemento" ).selectable();`

Seleccionar días de otros meses

```
$(function(){
    $("#date").datepicker({
        showOtherMonths: true,
        selectOtherMonths: true
    });
});
```

Con una imagen de guía para seleccionar

```
$(function(){
    $("#date").datepicker({
        showOn: "button",
        buttonImage: "images/calendar.gif",
        buttonImageOnly: true,
        buttonText: "Seleccione la fecha"
    });
});
```

## Selector Progressbar(): permite llenar una barra

```
$(document).ready(function(){
    $("#barra_deter").progressbar({
        value: 50
    });
    $("#barra_indet").progressbar({
        value: false
    });
});
```

`$( "elemento" ).progressbar();`

## Spinner(): para seleccionar números automáticamente

## Opciones de Spinner()



- ✓ Culture
- ✓ Disabled
- ✓ Icons
- ✓ Incremental:
- ✓ Max
- ✓ Min
- ✓ numberFormat
- ✓ Page
- ✓ Step

```
<h2>Spineer</h2>
<input id="numeros" value="50"><br>
<h2>Slider</h2>
<div id="rango"></div>
Rango de Valores: <input type="text" id="valores" />
<h2>Select Menu</h2>
<>Selecione una Marca de Carro:</>
<select name="carros" id="carros">
<option value="Toyota">Toyota</option>
```

\$(Selector).spinner(opciones)

\$(Selector).spinner("acción", params)

```
$(>function() {
  $("#numeros").spinner();
  $("#rango").slider({
    range:true,
    min: 0,
    max: 50,
    slide: function ( event, ui ) {
      $("#valores").val(ui.values[ 0 ] + " - " + ui.values [ 1 ] );
    }
  });
  $("#valores").val($("#rango").slider( "values", 1 ));
  $("#carros").selectmenu();
});
```

## Slider(): para seleccionar entre un rango de datos o un valor en una barra

### Opciones de Slider()



- ✓ Disabled
- ✓ Max
- ✓ Min
- ✓ Orientation
- ✓ Range
- ✓ Step
- ✓ Value
- ✓ Values

```
<h2>Spineer</h2>
<input id="numeros" value="50"><br>
<h2>Slider</h2>
<div id="rango"></div>
Rango de Valores: <input type="text" id="valores" />
<h2>Select Menu</h2>
<>Selecione una Marca de Carro:</>
<select name="carros" id="carros">
<option value="Toyota">Toyota</option>
```

\$(Selector).slider(opciones)

\$(Selector).slider("acción", params)

```
$(>function() {
  $("#numeros").spinner();
  $("#rango").slider({
    range:true,
    min: 0,
    max: 50,
    slide: function ( event, ui ) {
      $("#valores").val(ui.values[ 0 ] + " - " + ui.values [ 1 ] );
    }
  });
  $("#valores").val($("#rango").slider( "values", 1 ));
  $("#carros").selectmenu();
});
```

## Selectmenu(): un combobox con opciones

### Opciones de Selectmenu()



- ✓ appendTo: Sirve para añadir un elemento al menú.
- ✓ Disabled: Desactiva el menú cuando está en true.
- ✓ Icons: Se utiliza para colocar iconos a los ítem del menú.
- ✓ Position: Identifica la posición del menú en relación con el elemento asociado.
- ✓ Width: Relaciona el ancho del menú en pixeles.

```
<h2>Spineer</h2>
<input id="numeros" value="50"><br>
<h2>Slider</h2>
<div id="rango"></div>
Rango de Valores: <input type="text" id="valores" />
<h2>Select Menu</h2>
<>Selecione una Marca de Carro:</>
<select name="carros" id="carros">
<option value="Toyota">Toyota</option>
```

\$(Selector).selectmenu(opciones)

\$(Selector).selectmenu("acción", params)

```
$(>function() {
  $("#numeros").spinner();
  $("#rango").slider({
    range:true,
    min: 0,
    max: 50,
    slide: function ( event, ui ) {
      $("#valores").val(ui.values[ 0 ] + " - " + ui.values [ 1 ] );
    }
  });
  $("#valores").val($("#rango").slider( "values", 1 ));
  $("#carros").selectmenu();
});
```

## Tabs(): manejador de pestañas

- ✓ La primera parte consiste en la estructura HTML que representará al widget:
- ✓ La segunda parte es la inclusión del llamado de los tabs en nuestro código de javascript.
- ✓ La sintaxis es:
- ✓ De esta manera convertirá nuestra lista del documento HTML en Tabs y serán visualizados por nuestros usuarios.

```
<div id="misTabs">
  <ul>
    <li><a href="#tab1">Tab 1</a>
    </li>
    <li><a href="#tab2">Tab 2</a>
    </li>
    <li><a href="#tab3">Tab 3</a>
    </li>
  </ul>
</div>
<div id="tab1">
  Mi tab1
</div>
<div id="tab2">
  Mi tab2
</div>
<div id="tab3">
  Mi tab3
</div>
</div>
```

```
$( "#misTabs" ).tabs();
```

```
<div id="misTabs">
  <ul>
    <li><a href="#tab1">Tab 1</a>
    </li>
    <li><a href="#tab2">Tab 2</a>
    </li>
    <li><a href="#tab3">Tab 3</a>
    </li>
  </ul>
</div>
<div id="tab1">
  Mi tab1
</div>
<div id="tab2">
  Mi tab2
</div>
<div id="tab3">
  Mi tab3
</div>
</div>
```

```
$(document).ready(function(){
  $("#misTabs").tabs();
  $("#misTabs").tooltip();
});
```

Permite manejar el evento para cambiar el tab

```
$(function(){
  $("#tab").tabs({
    event: "mouseover"
  });
});
```

permite mover los tabs

```
$(function(){
  var tab = $("#tab").tabs();
  tab.find(".ui-tabs-nav").sortable({
    axis: "x",
    stop: function(){
      tab.tabs("refresh");
    }
  });
});
```

## ToolTip()

✓ La primera parte consiste en definir los elementos que tendrán tooltip.

✓ A los elementos hay que agregarles el atributo title.

Ej:

```
<input title="Mensaje de mi caja de texto!">
```

```
<p title="Mensaje de mi parrafo">Mi parrafo!</p>
```

✓ La segunda parte consiste en invocar al método tooltip().

Ej:

```
$( document ).tooltip();
```

```
$(<document>).ready(function(){
    $("#misTabs").tabs();
    $(document).tooltip();
})
```

## Efectos

**Agregar o quitar clases de CSS:** sobre cualquier elemento para simular animaciones

**Método addClass()**



- ✓ El primer parámetro representa la clase CSS a agregar.
- ✓ El segundo parámetro es el tiempo en milisegundos de duración de la animación.
- ✓ El tercer parámetro es una función easing.
- ✓ El cuarto parámetro es una función que se ejecutará, después que se termine la animación.

```
$("#miDiv").addClass( "miEstilo" , 3000, "easeOutExpo" );
```

Ejemplo:

```
$("#boton1").click(function() {
    $("#miDiv").addClass( "miEstilo2" , 1000, "easeOutExpo" );
});
$("#boton2").click(function() {
    $("#miDiv").removeClass( "miEstilo2" , 1000, "easeInOutQuint" );
});
```

**Método removeClass()**

- ✓ Remueve clases css a uno o varios elementos.
- ✓ Añade animaciones específicas.
- ✓ Puede recibir cuatro parámetros.
- ✓ Sintaxis:

```
removeClass(nombreClaseCss,duración,funcionEasing,funcionCompleto )
```

**\$( '#nextu-text1' ).switchClass();**  
Altera entre dos clases definidas como argumentos sobre el elemento seleccionado, en un tiempo determinado.

**\$( '#nextu-image-panel' ).addClass();**  
Permite agregar clases que se pasan como argumentos, al elemento seleccionado.

**\$( '#nextu-container' ).toggleClass();**  
Permite añadir o quitar las clases que se pasan como argumento, al elemento seleccionado.

**\$( '#nextu-image-panel' ).removeClass();**  
Permite quitar una o varias clases en específico que se pasan como argumentos, al elemento seleccionado.

**Animate():** permite animar con estilos y transición. Solo soporta algunas propiedades

✓ Ejecuta animaciones sobre un elemento.

✓ Cambiando las propiedades CSS.

✓ El método solo soporta una serie de propiedades CSS.

- ✓ backgroundColor
- ✓ borderTopColor
- ✓ borderBottomColor
- ✓ borderLeftColor
- ✓ borderRightColor
- ✓ color
- ✓ outlineColor

```
$("#miDiv").animate(
{
    color: "blue",
    backgroundColor: "green",
}
);
```

**Efecto Easing():** permite crear animaciones rebotes con velocidades variadas

- ✓ Son funciones matemáticas.
- ✓ Cambian la velocidad de una animación entre un punto y otro.
- ✓ El núcleo de JQuery tiene disponible dos funciones easing:
  - linear.
  - swing

```
$(document).ready(function(){
  $("#boton1").click(function(){
    var distance = parseInt($("#miInput").val());
    $("#miDiv1").effect("bounce",{
      distance: distance //La distancia en pixeles en la que podra rebotar el elemento Div
    },3000,function(){
      $(this).show();
    });
  });
});
```

## Effect(): permite crear efectos varios

`.effect(efecto, opciones,duracion,funcionCo)`

- ✓ El primer parámetro representa el tipo de efecto.
- ✓ El segundo parámetro representa a objetos que contiene opciones de configuración.
- ✓ El tercer parámetro representa el tiempo en milisegundos que durará el efecto visual.
- ✓ El cuarto parámetro representa una función que se ejecutará cuando finalice el efecto.

### Efectos

- |             |            |
|-------------|------------|
| ✓ blind     | ✓ slide    |
| ✓ fade      | ✓ transfer |
| ✓ fold      | ✓ drop     |
| ✓ highlight | ✓ explode  |
| ✓ bounce    | ✓ shake    |
| ✓ clip      | ✓ puff     |
| ✓ size      | ✓ pulsate  |

```
$("#boton1").click(function() {
  $("#miDiv1").effect("explode", 4000, function() {
    $(this).show();
  });
});

$("#boton2").click(function() {
  $("#miDiv2").effect("shake", 4000)
});
```

## Hide(): esconde con efecto un elemento

- ✓ El primer parámetro representa el tipo de efecto.
- ✓ El segundo parámetro representa un objeto que contiene opciones de configuración.
- ✓ El tercer parámetro representa la duración en milisegundos del efecto.
- ✓ El cuarto parámetro representa una función que se ejecutara al finalizar el efecto.

## Show(): muestra con efecto un elemento

- ✓ El primer parámetro representa el tipo de efecto.
- ✓ El segundo parámetro representa un objeto que contiene opciones de configuración.
- ✓ El tercer parámetro representa la duración en milisegundos del efecto.
- ✓ El cuarto parámetro representa una función que se ejecutara al finalizar el efecto.

```
$(document).ready(function() {
  $("#boton1").click(function() {
    $("#miDiv1").hide("explode", 4000);
  });

  $("#boton2").click(function() {
    $("#miDiv1").show("shake", 4000)
  });
})
```

## Toggle(): muestra u oculta según sea elementos con efecto

- ✓ Permite mostrar un elemento si esta oculto u ocultarlo si esta visible.
  - ✓ Aplicando un efecto visual.
  - ✓ La sintaxis es la siguiente:
- `.toggle(efecto,opciones,duracion,funcionCo)`
- ✓ Recibe 4 parámetros.

- ✓ El primer parámetro representa el tipo de efecto visual.
- ✓ El segundo parámetro representa un objeto que posee opciones de configuración.
- ✓ El tercer parámetro representa la duración en milisegundos del efecto visual.
- ✓ El cuarto parámetro representa la función que se va ejecutar al finalizar el efecto visual.

## toggleClass(): cambia clases CSS de elementos con efecto

✓ Permite agregar una clase CSS si no la tiene el elemento o removerla si la tiene.

✓ Aplicando un efecto visual.

✓ La sintaxis básica es la siguiente:

```
.toggleClass(nombreClase,switch,duracion,funcionEasing,funcionCompleto)
```

✓ Recibe 5 parámetros.

```
.toggleClass(nombreClase,switch,opciones)
```

✓ El primer parámetro representa el nombre de la clase CSS que va ser removida o agregada.

✓ El segundo parámetro representa un booleano que determina si la clase será removida o agregada.

✓ El tercer parámetro representa un objeto que posee opciones de configuración.

✓ El primer parámetro representa la clase CSS que va ser removida o agregada.

✓ El segundo parámetro representa un booleano que indica si la clase se debe remover o agregar.

✓ El tercer parámetro representa la duración en milisegundos del efecto visual.

✓ El cuarto parámetro indica la función easing se utilizará.

✓ El quinto parámetro representa una función que se ejecutará al finalizar el efecto visual.

Estas son las opciones de configuración:

✓ duration

✓ complete

✓ children

✓ queue

**SwitchClass():** cambia clases CSS por otras colocando efectos en el cambio.

✓ Permite cambiar una clase CSS por otra que en un elemento.

✓ Agrega efectos visuales cuando cambia la clase CSS.

✓ La sintaxis básica es la siguiente:

```
.switchClass(claseRemover,claseAregar,duracion,funcionEasing,funcionCompleto)
```

✓ Posee 5 parámetros.

✓ El primer parámetro representa de la clase CSS que se va remover.

✓ El segundo parámetro representa la clase CSS que se va a agregar.

✓ El tercer parámetro representa la duración en milisegundos del efecto visual.

✓ El cuarto parámetro es la función easing que ejecutará el efecto visual.

✓ El quinto parámetro representa la función que se va ejecutar después que termine el efecto visual.

```
.switchClass(claseRemover,claseAregar,opciones)
```

Opciones de configuración:

✓ duration

✓ easing

✓ complete

✓ children

✓ queue

```
.miEstilo1 {  
background-color: #C52FD9;  
border-top: groove;  
width: 200px;  
height: 200px;  
}  
  
.miEstilo2 {  
background-color: #3CE55B;  
color:#fff;  
font-size:xx-large;  
border-top: groove;  
width: 400px;  
height: 400px;  
}
```

```
$(document).ready(function() {  
  
$("#boton1").click(function() {  
  
$("#miDiv1").switchClass("miEstilo1","miEstilo2",2000,"easeInOutBounce");  
  
});  
})
```

# Utilitarios

## Position(): posiciona elementos a un lado de otro

- ✓ Permite posicionar un elemento respecto a otro.
- ✓ Su sintaxis es la siguiente:

```
.position(opciones)
```
- ✓ Recibe un solo parámetro, el cual es un objeto que contiene opciones de configuración.

```
#miDiv2 {  
    position: absolute;  
    width: 50px;  
    height: 50px;  
    background: #2280FF;  
}  
  
#miDiv3 {  
    position: absolute;  
    width: 50px;  
    height: 50px;  
    background: #FFA888;  
}  
  
.elementoObjetivo {  
    background-color: #050505;  
    position: absolute;  
    width: 300px;  
    height: 300px;  
    left: 100px;  
    top: 50px;  
}
```

```
$("#miDiv1").position({  
    my: "right bottom",  
    at: "right bottom",  
    of: "#elementoObjetivo"  
});  
  
$("#miDiv2").position({  
    my: "right bottom",  
    at: "left top",  
    of: "#elementoObjetivo",  
});
```

Opciones de configuración:

✓ my

✓ at

✓ of

✓ Los valores de my y at pueden ser:

**Horizontales:** left, center y right.

**Verticales:** top, center y bottom.

✓ También pueden ser combinaciones: left top , right bottom ,center top ,etc.

## Widget Factory(): para crear widgets

- ✓ Es una función que se identifica así:  
(\$.widget)
- ✓ Su funcionamiento consiste en tomar dos argumentos básicos.
- ✓ Uno es el nombre del widget y el otro un objeto.
- ✓ Con los dos argumentos crea una clase y un plugin.

- ✓ Su sintaxis es la siguiente:

```
$.widget(nombre,base,prototipo)
```

- ✓ El primer parámetro representa el nombre del widget.

- ✓ El Segundo parámetro es un constructor que permitirá inicializar propiedades y crear métodos.

- ✓ El tercer parámetro representa un objeto que se usará como prototipo.

Opciones del segundo parámetro:	Métodos del segundo parámetro:
✓ disable	✓ enable()
✓ hide	✓ option(nombreOpcion) ✓ option() ✓ option( nombreOpcion, value ) ✓ option( options )
✓ show	✓ _create() ✓ _delay( fn [, delay ] ) ✓ _focusable( elemento ) ✓ _hide( elemento, opciones [, callback ] ) ✓ _setOption( clave, valor ) ✓ _show( elemento, opciones [, callback ] ) ✓ widget()

### Ejemplo: un widget para operar dos numeros

```

$(document).ready(function() {
    $.widget("com.miWidget", {
        _create: function() {
            this._inputNumero1 = $("<input>");
            this._inputNumero1.attr("placeholder", "Digite el primer numero");
            this._inputNumero1.css("width", "200px");
            this._inputNumero1.css("border-style", "solid");
            this._inputNumero1.css("text-align", "left");
            this._inputNumero1.css("margin", "0 auto");
            $(this.element).append(this._inputNumero1);

            this._inputNumero2 = $("<input>");
            this._inputNumero2.attr("placeholder", "Digite el segundo numero");
            this._inputNumero2.css("width", "200px");
            this._inputNumero2.css("border-style", "solid");
            this._inputNumero2.css("text-align", "left");
            this._inputNumero2.css("margin", "0 auto");
            $(this.element).append(this._inputNumero2);

            this._operaciones = $("<select>");
            this._operaciones.css("width", "200px");

            var operacion1 = document.createElement("option");
            operacion1.text = "Elija una operacion...";
            operacion1.value = "";

            var operacion2 = document.createElement("option");
            operacion2.text = "Suma";
            operacion2.value = "1";

            var operacion3 = document.createElement("option");
            operacion3.text = "Resta";
            operacion3.value = "2";

            var operacion4 = document.createElement("option");
            operacion4.text = "Multiplicación";
            operacion4.value = "3";

            var operacion5 = document.createElement("option");
            operacion5.text = "División";
            operacion5.value = "4";

            this._operaciones.append(operacion1);
            this._operaciones.append(operacion2);
            this._operaciones.append(operacion3);
            this._operaciones.append(operacion4);
            this._operaciones.append(operacion5);

            $(this.element).append(this._operaciones);
        }
    });

    calcular: function() {
        if(this.validar()) {
            var opcion = parseInt(this._operaciones.val());
            var resultado;
            var numero1;
            var numero2;

            switch (opcion) {
                case 1:
                    numero1 = parseInt(this._inputNumero1.val());
                    numero2 = parseInt(this._inputNumero2.val());
                    resultado = numero1 + numero2;
                    alert("El resultado de la suma es: " + resultado);
                    break;

                case 2:
                    numero1 = parseInt(this._inputNumero1.val());
                    numero2 = parseInt(this._inputNumero2.val());
                    resultado = numero1 - numero2;
                    alert("El resultado de la resta es: " + resultado);
                    break;

                case 3:
                    numero1 = parseInt(this._inputNumero1.val());
                    numero2 = parseInt(this._inputNumero2.val());
                    resultado = numero1 * numero2;
                    alert("El resultado de la multiplicación es: " + resultado);
                    break;

                case 4:
                    numero1 = parseInt(this._inputNumero1.val());
                    numero2 = parseInt(this._inputNumero2.val());
                    resultado = numero1 / numero2;
                    alert("El resultado de la división es: " + resultado);
                    break;
            }
        }
    }

    $("#boton1").click(function() {
        $("#miDiv").miWidget();
    });

    $("#boton2").click(function() {
        $("#miDiv").miWidget("calcular");
    });
}

```

### Ejemplo: widget animado

```

$(document).ready(function() {
    $.widget("com.myDivW48idget", {
        _create: function() {
            this._div = $("<div>");
            this._div.text("Mi div Widget!!!");
            this._div.css("width", "200px");
            this._div.css("border-style", "solid");
            this._div.css("text-align", "center");

            this._div.css("margin", "0 auto");
            $(this.element).append(this._div);
        },
        animar: function() {
            this._div.css("background-color", "red");
            this._div.toggle('pulsate', 3000);
        }
    });

    $("#boton1").click(function() {
        $("#miDiv").myDivWidget();
    });
    $("#boton2").click(function() {
        $("#miDiv").myDivWidget("animar");
    });
}

```

## JQUERY MOBILE

- ✓ Es un marco de interfaz de usuario basado en JQuery.
- ✓ Funciona en todas las plataformas móviles, tabletas, e-Reader y plataformas de escritorio.
- ✓ Debemos hacer uso de la interfaz de JQuery Mobile:
  - ✓ <http://jquerymobile.com/download>

### Page

- ✓ Una página HTML es el documento o archivo utilizado para crear aplicaciones móviles con la interfaz de JQuery Mobile.
  - ✓ Secciones principales:
    - ✓ Encabezado
    - ✓ Espacio de contenido
    - ✓ Pie de página
- ✓ Composición:
- ✓ data-role
  - ✓ data-theme
  - ✓ data-position
  - ✓ data-transition
  - ✓ data-icon

One single page

```
<meta name="viewport" content="idth = device-width, initial-scale=1">
<script src="jquery-1.11.1.js"></script>
<script src="jquery.mobile-1.4.4.js"></script>
<link rel="stylesheet" href="jquery.mobile-1.4.4.css">

<body>
  <div data-role="page" id="page1">
    <div data-role="header">
      <h1>HEADER</h1>
    </div>
    <div data-role="content">
      <p>      <n4>CONTENIDO</n4>
      </p>
    </div>
    <div data-role="footer">
      <p align="center">FOOTER</p>
    </div>
  </div>
</body>
</html>
```

Con el data-role se crean varias paginas en un solo HTML

```

<div data-role="page" id="pagina1">
  <div data-role="header">
    <h1>Registro Cliente</h1>
  </div>
  <div data-role="content">
    <form>
      <div class="ui-field-contain">
        <label for="nombre">Nombre:</label>
        <input type="text" name="nombre" id="nombre">
        <label for="apellido">Apellido:</label>
        <input type="text" name="apellido" id="apellido">
      </div>
      <a href="#pagina2" class="ui-btn">Siguiente Paso</a>
    </form>
  </div>
  <div data-role="footer">
    <p align="center">Paso 1</p>
  </div>
</div>

<div data-role="page" id="pagina2">
  <div data-role="header">
    <h1>Registro Cliente</h1>
  </div>
  <div data-role="content">
    <form>
      <div class="ui-field-contain">

```



## Cuadro de dialogo y botones

```

<div data-role="page" id="pagina2" data-dialog="true">
  <div data-role="header">
    <h1>Datos Producto</h1>
  </div>
  <div data-role="content">
    <h1 id="datos"></h1>
    <a href="#" id="boton3" class="ui-btn">Regresar</a>
  </div>
</div>

```

## Formas de Navegación

- ✓ Atributo **href** del elemento **<a>**.
  - ✓ Ejemplo:

<a href="#pagina2" class="ui-btn">Siguiente</a>

```

$("#datos").html(datos);
$.mobile.changePage("#pagina4");

```

## Ejemplos

```

<div data-role="page" id="pagina1">
  <div data-role="header">
    <h1>OPERACIONES MATEMÁTICAS</h1>
  </div>
  <div data-role="content">
    <form>

      <a onclick="cambiarPagina('paginaSuma')" class="ui-btn">SUMA</a>
      <a onclick="cambiarPagina('paginaResta')" class="ui-btn">RESTA</a>
      <a onclick="cambiarPagina('paginaMultiplicacion')" class="ui-btn">MULTIPLICACIÓN</a>
      <a onclick="cambiarPagina('paginaDivision')" class="ui-btn">DIVISIÓN</a>

    </form>
  </div>
  <div data-role="footer">
    <p align="center">OPERACIONES MATEMÁTICAS</p>
  </div>
</div>

```

```

function cambiarPagina(page) {
  $.mobile.changePage("#"+page);
}

```

## Loader: widget de carga

- ✓ Se implementa mediante dos métodos:
  - ✓ **show:**  
\$.mobile.loading("show");
  - ✓ **hide:**  
\$.mobile.loading("hide");
- ✓ Opciones de configuración:

- ✓ Muestra el cuadro de diálogo mientras se esté procesando algunas operaciones que se ejecutan a través de nuestro código.

Ejemplo:

```
$(document).on("mobileinit", function() {
    $.mobile.loader.prototype.options.text = "Conectando...";
    $.mobile.loader.prototype.options.textVisible = true;
    $.mobile.loader.prototype.options.textonly = false;
    $.mobile.loader.prototype.options.theme = "a";
    $.mobile.loader.prototype.options.html = "";
});

});
```



Ejemplo 1:

```
$(document).ready(function() {
    $("#boton1").click(function() {
        $.mobile.loading("show");
        setTimeout(function() {
            $.mobile.loading("hide");
            var usuario = $("#usuario").val();
            var mensaje = "Bienvenido " + usuario + " al sistema!!!!";
            $("#mensaje").html(mensaje);
            $.mobile.changePage("#paginaPerfil");
        }, 4000);
    });
});
```

Ejemplo2:

```
$("#boton1").click(function() {
    $.mobile.loading("show", {
        text: "Calculando area del triangulo...",
        textVisible: true,
        theme: "a",
        textonly: false,
        html:""
    });
    setTimeout(function() {
        var baseT = parseFloat($("#baseT").val());
        var alturaT = parseFloat($("#alturaT").val());
        var resultado = (baseT * alturaT) / 2;
        $.mobile.loading("hide");
        alert("El area del triangulo es: " + resultado);
        $("#baseT").val("");
        $("#alturaT").val("");
    }, 4000);
});
```

## Transiciones

- ✓ Algunos efectos son:
  - ✓ fade, pop, filip, turn, flow, slidefade, slide, slideup, slidtdown, none.
- ✓ Existen dos formas de aplicar transición:
  - ✓ Definiendo globalmente el tipo de efecto.
  - ✓ Definiendo el tipo de transición en `$.mobile.changePage()`.

```
$(document).on("mobileinit", function() {
    $.mobile.defaultPageTransition='fade';
});
```

```
function cambiarPagina(page) {
    switch (page) {
        case "paginaRegistro":
            $.mobile.changePage("#" + page, {
                transition: "slidedown"
            });
            break;
        case "paginaConsulta":
            $("#codigoB").val("");
            $("#datos").html("");
            $.mobile.changePage("#" + page, {
                transition: "slideup"
            });
            break
    }
}
```

## Frameworks y clases

- ✓ **ui-corner-all:** Añade esquinas redondeadas al elemento.
- ✓ **ui-shadow:** Agrega una sombra al elemento alrededor del mismo.
- ✓ **ui-overlay-shadow:** Agrega una sombra superpuesta alrededor del elemento.
- ✓ **ui-mini:** Reduce el tamaño de la fuente.

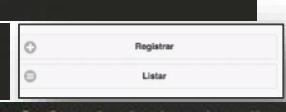
- ✓ **ui-btn:** Esta clase permite indicar que el elemento se comporte como un botón.
- ✓ **ui-btn-inline:** Muestra el botón en línea.
- ✓ **ui-shadow-icon:** Dibuja una sombra alrededor del ícono.

- ✓ **ui-icon-check**
- ✓ **ui-icon-delete**
- ✓ **ui-icon-edit**
- ✓ **ui-icon-home**
- ✓ **ui-icon-minus**
- ✓ **ui-icon-plus**
- ✓ **ui-icon-refresh**

```
<input type="text" class="ui-corner-all ui-shadow ui-overlay-shadow" name="cateto1" id="cateto1">
<label for="cateto2">Cateto 2:</label>
<input type="text" class="ui-corner-all ui-shadow ui-overlay-shadow" name="cateto2" id="cateto2">
<label for="hipotenusa">Hipotenusa:</label>
<input type="text" class="ui-corner-all ui-shadow ui-overlay-shadow" name="hipotenusa" id="hipotenusa" disabled>

<a id="boton1" class="ui-btn ui-corner-all ui-shadow ui-overlay-shadow">Calcular</a>
<a id="boton2" class="ui-btn ui-corner-all ui-shadow ui-overlay-shadow">Limpiar</a>

<a onclick="cambiarPagina('paginaRegistro')" class="ui-btn ui-icon-plus ui-btn-icon-left ui-shadow-icon">Registrar</a>
<a id="btnListarCliente" class="ui-btn ui-icon-bars ui-btn-icon-left ui-shadow-icon">Listar</a>
```



## Grid layout – ui-grid

- ✓ Permite construir columnas basadas en CSS.
- ✓ Clases predefinidas:
  - ✓ **ui-grid-a** (Dos columnas).
  - ✓ **ui-grid-b** (Tres columnas).
  - ✓ **ui-grid-c** (Cuatro columnas).
  - ✓ **ui-grid-d** (Cinco columnas).
- ✓ Son 100% anchas y totalmente invisibles.
- ✓ Están compuestos por cuadriculas.

```
<div class="ui-grid-a">
    <div class="ui-block-a"><a id="boton1" class="ui-btn">Centimetros a Pulg</a>
    </div>
    <div class="ui-block-b"><a id="boton2" class="ui-btn">Yardas a Metros</a>
    </div>
    <div class="ui-block-a"><a id="boton3" class="ui-btn">Km a Millas</a>
    </div>
    <div class="ui-block-b"><a id="boton4" class="ui-btn">Pies a Centimetros</a>
    </div>
```

## Widgets

### Checkboxradio

- ✓ Se debe crear un elemento `input`, con el atributo `type="checkbox"`.

#### Ejemplo:

```
<input type="radio" name="radio-genero" id="radio-genero-1" value="M" checked="checked">
<label for="radio-genero-1">Masculino</label>
<input type="radio" name="radio-genero" id="radio-genero-2" value="F" checked="checked">
<label for="radio-genero-2">Femenino</label>
```

```
<fieldset data-role="controlgroup">
    <legend>Genero</legend>
    <input type="radio" name="radio-genero" id="radio-genero-1" value="M" checked="checked">
    <label for="radio-genero-1">Masculino</label>
    <input type="radio" name="radio-genero" id="radio-genero-2" value="F" checked="checked">
    <label for="radio-genero-2">Femenino</label>
</fieldset>
```

## Collapsible

- ✓ Crea un bloque desplegable con contenidos.
- ✓ Se debe crear un elemento **div**, con y añadir el atributo **data-role**, con el valor **collapsible**.
- ✓ Dentro de la etiqueta **div**, se agregan encabezados (H1 – H6).
- ✓ Automáticamente se agrega un estilo al encabezado.
- ✓ Se agregan código HTML para el contenido desplegable.

```
<div id="listaHombres" data-role="collapsible">
  <h3>Hombres</h3>
  <table id="tablaHombres" style="width: 100%" border="1">
    <tr> <th>Id</th>
      <th>Nombre</th>
      <th>Edad</th>
    </tr>
    </table>
</div>
```

## Collapsibleset

- ✓ Está compuesto por un conjunto de **collapses**.
- ✓ Posee características de acordeón.
- ✓ Se debe crear un **div** con el atributo **data-role="collapsibleset"**
- ✓ Todos los elementos **collapses** se despliegan inicialmente, de lo contrario se hace uso de **data-collapsed="false"**.

## Popup

- ✓ Se debe crear un elemento **div**, y como atributo **data-role="popup"**.
- ✓ Se debe crear un elemento de tipo **a**, con el atributo **data-rel = "popup"**.
- ✓ Posee dos métodos:
  - ✓ **open**
  - ✓ **close**

```
<div data-role="popup" id="popupDialogoResultado" data-dismissible="false" style="width:300px;">
  <div data-role="header">
    <h1>Resultado</h1>
  </div>
  <div role="main" class="ui-content">
    <h3 id="resultado"></h3>
    <a href="#" class="ui-btn" data-rel="back">Ok</a>
  </div>
</div>
```

Esto va en el script para ejecutar el popup

```
$("#resultado").html("El resultado del Seno es: "+resultado);
$('#popupDialogoResultado').popup('open');
```

## Listview

- ✓ Se debe crear una lista desordenada simple **<ul>**.
- ✓ **<ul>** debe llevar el atributo **data-role="listview"**.
- ✓ JQuery Mobile aplicará todos los estilos necesarios para transformar la lista desordenada en un listview.

```
<div data-role="content">

  <ul data-role="listview" id="listaNotas" data-inset="true">
  </ul>

</div>
```

## Navbar

- ✓ Permite navegar entre páginas.
- ✓ Se definen en **header** o un **footer**.
- ✓ Se codifica como una lista desordenada.
- ✓ La lista desordenada debe ir en un contenedor.
- ✓ El contenedor debe tener el atributo **data-role = "navbar"**.

```
<div data-role="navbar">
  <ul>
    <li><a class="ui-btn-active ui-state-persist" href="#">Nueva Nota</a>
    </li>
    <li><a id="btnListarNotas">Lista Notas</a>
    </li>
  </ul>
</div>
```

## SelectMenu

- ✓ Permite elegir un ítem entre varias opciones.

#### Ejemplo:

```
<label for="campoColor" class="select">Color:</label><select name="campoColor" id="campoColor">
    <option value="1">Rojo</option>
    <option value="2">Azul</option>
    <option value="3">Amarillo</option>
</select>
```

**Reflow-table:** Permite crear una tabla adaptable

```
<table data-role="table" id="tablaComputadores" data-mode="reflow"
    class="ui-responsive table-stroke">
    <thead>
        <tr>
            <th data-priority="1">Código</th>
            <th data-priority="2">Marca</th>
            <th data-priority="3">RAM</th>
            <th data-priority="4">Disco Duro</th>
            <th data-priority="5">P</th>
            <th data-priority="6">Código</th>
        </tr>
    </thead>
```

## Eventos

- ✓ **orientationchange:** Se dispara cuando la orientación del dispositivo cambia.
- ✓ **pagebeforechange:** se dispara dos veces en el cambio de páginas.
- ✓ **pagechange:** se dispara luego de la ejecución del método `changePage()`.

Ejemplo: validaciones antes de cambiar de pagina

```
$(document).ready(function(){
    $(document).on("pagebeforechange", function(event, data){
        if(typeof data.toPage == "string"){
            var page=$.mobile.activePage.attr('id');
            if(page=="paginaPaso1"){
                if($("#id").val().trim()==="" || $($("nombre").val().trim()==="")){
                    $("#paginaPaso1").css({
                        "background-color": "#FF3145"
                    });
                    alert("Por favor rellene los campos.");
                    return false;
                }
            }
        }
    });
});
```

Ejemplo al entrar a una pagina nueva

```
$(document).on("pagechange", function(event, data){
    var page = $.mobile.activePage.attr('id');
    if(page=="paginaRegistroExito"){
        $("#paginaRegistroExito").css({
        })
    }
});
```

## Evento Swipe

### swipe:

- ✓ Ocurre cuando se realiza un arrastre (de más de 30px horizontalmente y menos de 75px verticalmente), en una dirección determinada.

### swipeleft:

- ✓ Ocurre cuando se realiza un arrastre (de más de 30px horizontalmente y menos de 75px verticalmente), a la izquierda.

### swiperight:

- ✓ Ocurre cuando se realiza un arrastre (de más de 30px horizontalmente y menos de 75px verticalmente), a la derecha.

```
$("#paginaCalculoDensidad").on("swipe", function()
```

## Google maps

Permite abrir un mapa con marcadores, se necesita que tenga estilos

```
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js"></script>
```

- ✓ Debemos definir el estilo de nuestro mapa (ancho, alto, márgenes, etc.).
  - ✓ Debemos hacer uso de la api de Google Maps.
  - ✓ Definir el contenedor que contendrá el mapa.
  - ✓ Declarar las opciones de configuración del mapa (objeto en javascript).
  - ✓ Debemos crear un objeto de tipo Map, para mostrar el mapa.
- ```
<style type="text/css">
html { height: 100% }
body { height: 100%; margin: 0; padding: 0 }
#map_canvas{ height: 100% }
</style>
```

```
$(document).ready(function(){
  var mapa = "null";
  var LatLngInicial = new google.maps.LatLng(25.760527, -80.192800);
  var marcadores = [];

  $("#btnVerMapa").click(function(){
    cambiarPagina("paginaMapa")

    var opciones = {
      zoom: 5,
      center: LatLngInicial,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    mapa = new google.maps.Map(document.getElementById("divMapa"), opciones);

    var marcador = new google.maps.Marker({
      position: LatLngInicial,
      map: mapa,
      title: "Marcador"
    });
    marcadores[0] = marcador;
  });
});
```

## Ventanas de información (google maps)

- ✓ Muestran contenido en una ventana flotante situada encima del mapa.
  - ✓ Son parecidos a las viñetas de los cómics.
  - ✓ Hace uso del objeto **InfoWindowOptions**.
  - ✓ **InfoWindowOptions** posee los siguientes campos:
    - ✓ content
    - ✓ pixelOffset
    - ✓ position
    - ✓ maxWidth
- ```
var contenido1 = '<div style="width:100px; height:50px;">'+
'<p>Este es el punto1.</p>'+'</div>';
var contenido2 = '<div style="width:100px; height:50px;">'+
'<p>Este es el punto2.</p>'+'</div>';
var contenido3 = '<div style="width:100px; height:50px;">'+
'<p>Este es el punto3.</p>'+'</div>';

var ventanaInfo1 = new google.maps.InfoWindow({
  content: contenido1
});

var ventanaInfo1 = new google.maps.InfoWindow({
  content: contenido1
});

var ventanaInfo1 = new google.maps.InfoWindow({
  content: contenido1
});
```

Se pueden asignar manejadores de eventos para maps

```
google.maps.event.addListener(marcador1, 'click', function(){
  ventanaInfo1.open(mapa, marcador1);
});
google.maps.event.addListener(marcador2, 'click', function(){
  ventanaInfo2.open(mapa, marcador2);
});
google.maps.event.addListener(marcador3, 'click', function(){
  ventanaInfo3.open(mapa, marcador3);
});
google.maps.event.addListener(marcador1, 'mouseover', function(){
  marcador1.setIcon("icon1.png");
});
```

## Eventos del marcador(google maps)

- ✓ google.maps.Marker, puede detectar los siguientes eventos:

- ✓ "click"
- ✓ "dblclick"
- ✓ "mouseup"
- ✓ "mousedown"
- ✓ "mouseover"
- ✓ "mouseout"

```
function mostrarMapa(){
    var opciones = {
        zoom: 3,
        center: latlng1,
        mapTypeId: google.maps.mapTypeId.ROADMAP
    };
    var mapa = new google.maps.Map(document.getElementById("divMapa"), opciones);

    marcador1 = new google.maps.Marker({
        position: latlng1,
        map: mapa,
        draggable: true,
        title:"Punto 1"
    });

    marcador1 = new google.maps.Marker({
        position: latlng1,
        map: mapa,
        draggable: true,
        title:"Punto 1"
    });
}
```

## Marcador arrastrable(google maps)

La propiedad **draggable** debe ser **true**:

```
var mi_marcador = new google.maps.Marker({
    draggable: true
});
```

Para capturar datos, debemos definir el evento **dragend**:

```
google.maps.event.addListener(mi_marcador, 'dragend',
    function(event) {
});
```

```
google.maps.event.addListener(marcador1, 'dragend', function(event){
    alert("Latitud:" + event.latLng.lat() + "-Longitud:" + event.latLng.lng());
});
```

```
marcador1 = new google.maps.Marker({
    position: latlng1,
    map: mapa,
    draggable: true,
    title: "Punto 1"
});
```

## GeoCoder: permite buscar un nombre de ubicación y verla en mapa

- ✓ Permite transformar direcciones en coordenadas geográficas.
- ✓ Usa el método **Geocoder.geocode()** para iniciar una solicitud al servicio de asignación de identificadores geográficos.
- ✓ El objeto **GeocodeRequest**, posee los siguientes campos:
  - ✓ address
  - ✓ latLng
  - ✓ bounds
  - ✓ region

```
function convertirDireccion(){
    var direccion = $("#punto").val();
    var geocoder = new google.maps.Geocoder();
    $.mobile.loading("show", {
        text: "Buscando punto...",
        textVisible: true,
        theme: "a",
        textonly: false,
    });
    geocoder.geocode ({
        'address': direccion
    }, function(resultados, estado){
        if(estado == google.maps.GeocoderStatus.OK){
            marcador = new google.maps.Marker({
                map: mapa,
                position: resultados[0].geometry.location
            });
            mapa.setCenter(resultados[0].geometry.location);
            $.mobile.loading("hide");
        }else{
            $.mobile.loading("hide")
        }
    })
}
```

## Servicio de Rutas

- ✓ El acceso es de forma asíncrona.
- ✓ Se debe incluir un método de devolución de llamada.
- ✓ Para utilizar rutas, se crea un objeto de tipo **DirectionsService**, ejecutándose **DirectionsService.route()**, incluyendo **DirectionsRequest**.
- ✓ **DirectionsRequest**, contiene los siguientes campos:
  - ✓ origin
  - ✓ destination
  - ✓ travelMode

```
var latlng2 = new google.maps.LatLng(4.684236, -74.113471);
var marcador1;
var marcador2;

function mostrarMapa(){
    var directionsDisplay;
    var directionsService = new google.maps.DirectionsService();
    var opciones = {
        zoom: 3,
        center: latlng1,
        mapTypeId: google.maps.mapTypeId.ROADMAP
    };
    var mapa = new google.maps.Map(document.getElementById("divMapa"), opciones);

    marcador1 = new google.maps.Marker({
        position: latlng1,
        map: mapa,
        draggable: true,
        title: "Punto 1"
    });

    marcador2 = new google.maps.Marker({
        position: latlng2,
        map: mapa,
        draggable: true,
        title: "Punto 2"
    });
}
```

```

marcador2 = new google.maps.Marker({
  position: latlng2,
  map: mapa,
  draggable: true,
  title: "Punto 2"
});

directionsDisplay = new google.maps.DirectionsRenderer();
directionsDisplay.setMap(mapa);

var peticion = {
  origin: latlng1,
  destination: latlng2,
  travelMode: google.maps.TravelMode.DRIVING
};
directionsService.route(peticion, function(respuesta, estado){
  if(estado == google.maps.DirectionsStatus.OK){
    directionsDisplay.setDirections(respuesta);
  }else{
    alert("¡Error en el servicio!" + estado);
  }
});

```

Para agregar un marcador con eventos

```

google.maps.event.addListener(mapa, 'click', function(event){
  agregarMarcador(event.LatLng);
});

function agregarMarcador(location){
  var marcadorNuevo = new google.maps.Marker({
    position: location,
    icon: "icon.png",
    map: mapa
  });
  google.maps.event.addListener(marcadorNuevo, 'click', function(){
    verRutas(marcadorNuevo.getPosition());
  });
  marcadores.push(marcadorNuevo);
}
mostrarMapa();

```

## Geolocalización

- ✓ Nos permite obtener la ubicación de un usuario.
- ✓ Se hace uso de tres métodos:
  - ✓ `getCurrentPosition()`
  - ✓ `watchPosition()`
  - ✓ `clearWatch()`

```

function obtenerPositionActual(){
  if(navigator.geolocation){
    navigator.geolocation.getCurrentPosition(exito, fallido, {
      maximumAge: 500000,
      enableHighAccuracy: true,
      timeout: 6000
    });
  }
  obtenerPositionActual();
}

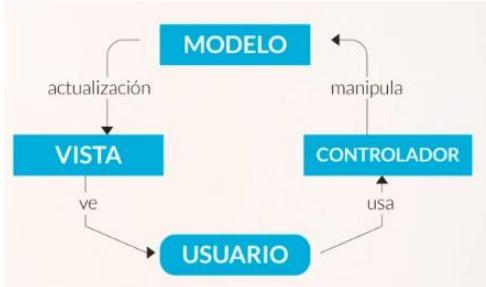
```

```

$(document).ready(function(){
  var marcador1;
  function mostrarMapa(posicion){
    var opciones = {
      zoom: 10,
      center: posicion,
      mapType: google.maps.MapTypeId.ROADMAP
    };
    var mapa = new google.maps.Map(document.getElementById(
      "divMapa"), opciones);
    marcador1 = new google.maps.Marker({
      position: posicion,
      map: mapa,
      draggable: true,
      title: "Mi punto."
    });
    function exito(pos){
      var latlng = new google.maps.LatLng(pos.coords.latitude,
        pos.coords.longitude);
      mostrarMapa(latlng);
    }
    function fallido(error){
      if(error.code == 0){
        alert("¡Ops, No se puede obtener la posición actual!");
      }
    }
  }
});

```

# ANGULAR



**HTML => MODELO**  
**CSS => VISTA**  
**BROWSER => CONTROLADOR**

AngularJS es un framework para aplicaciones web dinámicas. Permite utilizar HTML como lenguaje base y permite extender la sintaxis de HTML para expresar componentes adicionales para su aplicación.

- ✓ El enlace de datos se realiza utilizando la abreviatura `{}`.
- ✓ Estructuras de control DOM para repetir / ocultar fragmentos DOM.
- ✓ El apoyo a las formas y la validación de formularios.
- ✓ Colocación de un nuevo comportamiento de los elementos DOM, tales como el manejo de eventos DOM.
- ✓ Agrupación de HTML en componentes reutilizables.

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/angular.min.js"></script>
```

Sencilla aplicación: {}

Siempre mencionar `ng-app=""` y dentro de divs debe ir el código

```
<html ng-app=""> o <html ng-app="myMetricApp"> con un nombre
<body>
  <div>
    <p>¿Cuánto es 10/15 * 1+1? : {{10/15 * 1+1}}</p>
  </div>
  <script src="angular.js"></script>
</body>
```

## Bootstrapping con AngularJS

Bootstrapping es el concepto que nos permite que nuestros desarrollos se adapten al tamaño de la pantalla. Con AngularJS podemos implementarlo para que se realice en las aplicaciones automáticamente utilizando la directiva `ngApp`. En casos avanzados, como cuando se utilizan cargadores de scripts, podemos inicializarlo manualmente para garantizar que este arranque.

- ✓ Se crea el inyector de código que será utilizado en la inyección de las dependencias.
- ✓ El inyector creará la raíz que se convertirá en el contexto para el modelo de nuestra aplicación.
- ✓ Angular luego "compila" el DOM a partir de la `ngApp` en el elemento raíz, el tratamiento de sus directivas y los enlaces que se encuentran a lo largo del proceso.

## Inicializando Angular manualmente(sin `ng-app`)

```

<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <div ng-controller="MainController">
      Hola soy HTML y estoy esperando a que Angular llegue...
    </div>
    <script type="text/javascript" src="angular.js"></script>
    <script type="text/javascript">
      angular.module('myDelayedApp', []).controller('MainController', ['$scope', function($scope){
        $scope.sayHello = '¡Hola, soy AngularJS, he llegado tarde!';
        alert($scope.sayHello)
      }]);
      function manualStart(){
        angular.element(document).ready(function(){
          angular.bootstrap(document, ['myDelayedApp']);
        });
        clearInterval(intervalo)
      }
      var intervalo = setInterval(function(){manualStart()}, 3000);
    </script>
  </body>
</html>

```

## Inicializando Angular automáticamente

Siempre debe ir:

```

<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<html lang="en" ng-app>
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <script type="text/javascript" src="js/angular.js"></script>
  </head>

  <body>
    <div class="container"> Angular inicializado y listo {{ :D }}</div>
  </body>
</html>

```

## Controladores Angular

En angular, un controlador es una función JavaScript constructora que se utiliza para aumentar el alcance de angular.

Se usa para:



Configurar el estado inicial del objeto \$scope.



Añadir comportamiento al objeto \$scope

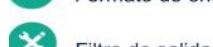
NO usar para:



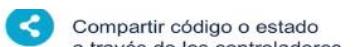
Manipular DOM



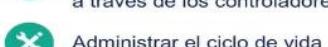
Formato de entrada



Filtro de salida



Compartir código o estado a través de los controladores.



Administrar el ciclo de vida de los otros componentes

Eje: conversión de unidades con funciones angular

```

<html ng-app="myMetricApp">
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <div ng-controller="MainController">
      <label>Digite distancia en metros:</label>
      <input ng-model="distance">
      <ul>
        <li>Nanómetros = {{metersToNanometers(distance)}}</li>
        <li>Micrómetros = {{metersToMicrometers(distance)}}</li>
        <li>Milímetros = {{metersToMillimeters(distance)}}</li>
        <li>Centímetros = {{metersToCentimeters(distance)}}</li>
        <li>Decímetros = {{metersToDecimeters(distance)}}</li>
        <li>Metros = {{metersToMeters(distance)}}</li>
        <li>Decámetros = {{metersToDecameters(distance)}}</li>
        <li>Hectómetros = {{metersToHectometers(distance)}}</li>
        <li>Kilómetros = {{metersToKilometers(distance)}}</li>
      </ul>
    </div>
    <script type="text/javascript">
      angular.module('MyMetricApp', []).controller('MainController', ['$scope', function($scope){
        $scope.distance = 0.0;
        $scope.metersToNanometers = function(meters){
          return meters * 1000000000;
        }
      }]);
    </script>
  </body>
</html>

```

Ej: se definen archivos independientes(forma de estructurar proyectos)

App.js

```
'use strict';

var carApp = angular.module('carApp', [
  'carAppControllers'
]);
```

Controller.js

```
'use strict';

var carAppControllers = angular.module('carAppControllers', []);

carAppControllers.controller('CarListCtrl', ['$scope',
  function($scope){
    $scope.value = "Welcome to CarApp";
  }
]);
```

Index.html

```
<html lang="en" ng-app="carApp">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <script type="text/javascript" src="js/angular.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="js/controllers.js"></script>
  </head>

  <body ng-controller="CarListCtrl">

  </body>
</html>
```

## Servicios Angular

Los servicios en Angular son objetos sustituibles que están conectados entre sí mediante la inyección de dependencias (DI). Los servicios en Angular se utilizan para organizar y compartir código a través de la aplicación.



Si es necesario – "Lazily instantiated"



Singleton

Como se crean:

- ✓ Al igual que otros identificadores básicos en Angular, los servicios integrados siempre comienzan con \$ (por ejemplo, \$ http ).
- ✓ Los desarrolladores de aplicaciones son libres de definir sus propios servicios.
- ✓ La función de servicios factory genera el objeto único o función que representa el servicio al resto de la aplicación.

```
<script type="text/javascript" src="angular.js"></script>
<script type="text/javascript">
  angular.module('cookieApp', []).controller('MainController', ['$scope', 'cookie', function(
    $scope, cookie){
    $scope.value = '';
    $scope.saveCookie = function(value){
      cookie.write('cap_value', value);
    }
    $scope.getCookie = function(){
      return cookie.read('cap_value');
    }
  }]);
  factory('cookie', [function(){
    return {
      read: function(name){
        var i, c, nameEQ = name + "=";
        var ca = document.cookie.split(';');
        for(i=0; i<ca.length; i++){
          c = ca[i];
          if(c.indexOf(nameEQ) === 0)
            return c.substring(nameEQ.length, c.length);
        }
      }
    }
  }]);
}</script>
```

Ej:

App.js

```
'use strict';

var carApp = angular.module('carApp', [
  'carAppControllers',
  'carAppServices'
]);
```

Controller.js

```
'use strict';

var carAppControllers = angular.module('carAppControllers', []);

carAppControllers.controller('CarListCtrl', ['$scope',
  function($scope){
    $scope.value = "Welcome to CarApp";
  }
]);
```

Index.html

```
<html lang="en" ng-app="carApp">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <script type="text/javascript" src="js/angular.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="js/controllers.js"></script>
    <script type="text/javascript" src="js/services.js"></script>
  </head>

  <body ng-controller="CarListCtrl">

  </body>
</html>
```

services.js

```
'use strict';

var carAppServices = angular.module('carAppServices', []);

carAppServices.factory('Car', [
  function(){
    return {
      notify: function(msg){
        alert(msg);
      }
    };
  }
]);
```

## Scopes

Los Scopes o el “alcance” es un objeto que está relacionado con la aplicación del modelo (la información de la aplicación). Es un contexto de ejecución para expresiones. Los Scopes en Angular tienen estructuras jerárquicas que imitan la estructura DOM de la aplicación.

#### Características

- ✓ Los Scopes proporcionan APIs (`$watch`) para observar cambios o mutaciones modelo.
- ✓ Los Scopes proporcionan API (`$apply`) para propagar los cambios modelo
- ✓ Los Scopes se pueden anidar para limitar el acceso a las propiedades de los componentes de la aplicación al tiempo que proporcionan acceso a las propiedades modelo compartido.
- ✓ Los Scopes proporcionan contexto contra el que las expresiones evalúan.

#### Cambiar el valor de un scope: un numero aleatorio

```
<!doctype html>
<html ng-app="randomizeApp">
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <div ng-controller="MainController">
      <button ng-click="getNumber()">Pulsar para obtener un número cualquiera</button>
      <h1>{{randomNumber}}</h1>
    </div>
    <script type="text/javascript" src="angular.js"></script>
    <script type="text/javascript">
      angular.module('randomizeApp', []).controller('MainController', ['$scope', function($scope) {
        $scope.randomNumber = 8.0;
        $scope.getNumber = function() {
          $scope.randomNumber = "Aquí tienes tu número :(" + Math.random();
        };
      }]);
    </script>
  </body>
</html>
```

#### Scope en listas

```
<li ng-repeat="car in cars">
  <span>{{car.name}}</span>
  <p>{{car.snippet}}</p>
</li>
```

### Inyección de Dependencias en AngularJS

Inyección de Dependencia (DI) es un patrón de diseño de software que se ocupa de cómo los componentes se apoderan de sus dependencias.

- ✓ Los componentes como los servicios, las directivas, los filtros y las animaciones son definidos por un método factory o función constructora.
- ✓ Los controladores se definen mediante una función constructora.
- ✓ El método `run` acepta una función, que puede ser inyectado con “servicio”, “valor” y componentes “constantes” como dependencias.
- ✓ El método `config` acepta una función, que puede ser inyectado con “proveedor” y componentes “constantes” como dependencias.

Ej:

Services.js

Controllers.js (inyección de ‘Temp’)

```
'use strict';

var realTempApp = angular.module('realTempAppServices', []);
realTempApp.factory('Temp', function(){
  return {
    celciusToFahrenheit: function(temp){
      var tempInFahrenheit = 0.0;
      var tempInCelcius = temp;

      tempInFahrenheit = tempInCelcius * 9/5 + 32;
      return tempInFahrenheit;
    },
    celciusToKelvin: function(temp){
      var tempInKelvin = 0.0;
      var tempInCelcius = temp;

      tempInKelvin = tempInCelcius - 273.15;
      return tempInKelvin;
    }
  };
});

var realTempApp = angular.module('realTempAppControllers', []);
realTempApp.controller('TempController', function($scope, Temp){
  $scope.tempInCelcius = 0.0;
  $scope.resultInFahrenheit = 0.0;
  $scope.resultInKelvins = 0.0;

  $scope.convertTemperatures = function(){
    $scope.resultInFahrenheit = Temp.celciusToFahrenheit($scope.tempInCelcius);
    $scope.resultInKelvins = Temp.celciusToKelvin($scope.tempInCelcius);
  };
});
```

## App.js

```
'use strict';

var realTempApp = angular.module('realTempApp', [
  'realTempAppControllers',
  'realTempAppServices'
]);
```

```
<!DOCTYPE html>
<html ng-app='realTempApp'>
  <head>
    <script type="text/javascript" src="js/angular.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="js/services.js"></script>
    <script type="text/javascript" src="js/controllers.js"></script>
  </head>

  <body>
    <div ng-controller="TempController">
      <h1>Convertidor de temperatura (Educativo)</h1>

      <form>
        <label>Ingrese la temperatura en grados Centígrados (C°)</label>
        <input type="text" ng-model="tempInCelcius">
        <button ng-click = "convertTemperatures()">Convertir</button>
      </form>

      <br>
      <h2>Como convertir Celsius a Fahrenheit </h2>
      <h3>Formula usada</h3>
      <p>°F = {{ tempInCelcius }} °C x 9/5 + 32</p>
      <p><br>Resultado : {{ resultInFahrenheit }} °F</p>
    </div>
  </body>
</html>
```

## Templates

En Angular, las plantillas se escriben con HTML que contiene elementos y atributos específicos Angular.

### Tipos de elementos



Ej: se refencian el texto del html directamente en el controlador

```
'use strict';

var carListApp = angular.module('carListAppController', []);
carListApp.controller('ListController', function($scope){
  $scope.appTitle = "CarList App 2014";
  $scope.formHeader = "Aregar un carro a tu increíble lista";
  $scope.deleteText = "Eliminar";
  $scope.addText = "Aregar";
  $scope.formTextBrand = "Marca";
  $scope.formTextYear = "Año";

  $scope.brand;
  $scope.year;

  $scope.cars = [
    {
      brand: 'BMW M4 Coupe',
      year: '2014'
    },
    {
      brand: 'Chevrolet Corvette Stingray'
      year: '2013'
    }
  ];

  $scope.addCar = function(){
    var car = { brand: $scope.brand,
               year: $scope.year};
    $scope.cars.push(car);
  };

  $scope.deleteCar = function(idx){
    $scope.cars.splice(idx, 1);
  }
});
```

## Expresiones

Las expresiones en Angular son fragmentos de código JavaScript que se colocan generalmente entre un doble juego de llaves, como {{expresión}}.

## Expresiones en Angular vs expresiones javascript

- ✓ Contexto
- ✓ Forgiving
- ✓ No hay Estado de Flujo de Control
- ✓ No hay declaraciones de funciones
- ✓ Sin Creación RegExp con notación literal
- ✓ Filtros

Ejemplo: con filtros(formato del campo)

### Controller.js

```
var smartExpressionApp = angular.module('smartExpressionAppController', []);
smartExpressionApp.controller('EvalController', function($scope){
    $scope.expression = '';
    $scope.expressions = [
        {
            value: '3*10|currency'
        },
        {
            value: '1288323623006 |date:"yyyy/MMM/dd"'
        },
        {
            value: '8/2+5'
        },
        {
            value: '(8/2) * 16 + 1'
        },
        {
            value: '12 * 2'
        },
        {
            value: '1 + 1 +1 +1 + 2 * 5 +0'
        },
        {
            value: '2500*1000|currency'
        }
    ];
    $scope.evaluate = function(){
        var exp = {value: $scope.expression};
        $scope.expressions.push(exp);
        $scope.expression = '';
    };
});
```

### index.html

```
<html lang="en" ng-app="smartExpressionApp">
    <head>
        <meta charset="UTF-8">
        <script type="text/javascript" src="js/angular.js"></script>
        <script type="text/javascript" src="js/app.js"></script>
        <script type="text/javascript" src="js/controllers.js"></script>
    </head>
    <body>
        <div ng-controller="EvalController">
            <h3>Evaluador de Expresiones</h3>
            <form>
                <label>Digite la expresión que desea evaluar</label>
                <br>
                Expression: <input type="text" ng-model="expression">
                <button ng-click = "evaluate()">Evaluar</button>
            </form>
            <br>
            <ul ng-repeat = "expression in expressions">
                <button ng-click="deleteExp($index)">Eliminar</button>
                {{expression.value}} => <span ng-bind="$parent.$eval(expression.value)"></span>
            </ul>
        </div>
    </body>
</html>
```

### Evaluador de Expresiones

Digite la expresión que desea evaluar

- Eliminar 3\*10|currency => \$30.00
- Eliminar 1288323623006 |date:"yyyy/MMM/dd" => 2010/October/28
- Eliminar 8/2+5 => 13
- Eliminar (8/2) \* 16 + 1 => 65
- Eliminar 12 \* 2 => 24
- Eliminar 1 + 1 +1 +1 + 2 \* 5 +0 => 14
- Eliminar 2500\*1000|currency => \$2,500,000.00

## Filtros

Un filtro formatea el valor de una expresión para la visualización al usuario. Pueden ser utilizados en las plantillas de vista, controladores o servicios. Se pueden crear filtros personalizados.

### Filtros personalizado

Para escribir un filtro personalizado, basta con registrar una nueva función de filtro de fábrica con su módulo. Internamente, este utiliza el filterProvider.

La función de filtro debe ser una función pura, lo que significa que no debe tener estado, debe estar sin estado.

Ejemplo: para filtrar datos con query

```
<body ng-controller="CarListCtrl">
    <ul>
        <li ng-repeat="car in cars | filter:query">
            <span>{{car.name}}</span>
            <p>{{car.snippet}}</p>
        </li>
    </ul>

    <form class="form-horizontal" role="form">
        <div class="form-group">
            <label>Filtro</label>
            <input type="text" ng-model="query">
        </div>
    </form>
```

## XHRs(XmlHttpRequest)

XMLHttpRequest (XHR), también referida como XMLHTTP (Extensible Markup Language / Hypertext Transfer Protocol), es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores Web.

Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas.

Se usa en el controller con el \$http

```
'use strict';

/* Controllers */

var carAppControllers = angular.module('carAppControllers', []);

//Inyectamos el servicio Car creado en services JS, que nos permite extraer los vehículos del proyecto
carAppControllers.controller('CarListCtrl', ['$scope', 'Car', '$http',
  function($scope, Car, $http) {
    /*Extraemos nuestros vehículos desde cars.json*/
    $http.get('cars/cars.json').success(function(data) {
      $scope.cars = data;
    });
}]);
```

## Enrutamiento y múltiples vistas

La funcionalidad de enrutamiento es proporcionada por Angular en el módulo ngRoute, que se distribuye por separado al marco del núcleo de Angular.

\$Route    ngView    ngRoute

Se debe descargar el modulo correspondiente de la página de angular, y luego implementar las vistas parciales para enrutarlas. Se usa la directiva ng-View en el index para cargar las vistas.

Ejemplo:

Car-list.html

```
<div class="blog-header">
  <h1 class="blog-title">Car Cluster 2015</h1>
  <p class="lead blog-description">Official blog car</p>
</div>

<div class="row">
  <div class="col-sm-7 blog-main">
    <ul ng-repeat="car in cars | filter:query"
        class="thumbnail car-listing">
      <a href="#/cars/{{car.id}}>{{car.name}}</a>
      <p>{{car.snippet}}</p>
    </ul>
  </div>
  <div class="col-sm-5 blog-sidebar">
    <form class="form-horizontal" role="form">
      <div class="form-group">
        <label for="name" class="col-sm-2 control-label">Filtro</label>
        <div class="col-sm-10">
          <input type="text" class="form-control" ng-model="query">
        </div>
      </div>
      <div class="form-group">
        <label for="inputPassword3" class="col-sm-2 control-label">Organizar por:</label>
        <div class="col-sm-10">
          <input type="text" class="form-control" ng-model="order_by">
        </div>
      </div>
    </form>
  </div>
</div>
```

App.js

```
'use strict';

var carApp = angular.module('carApp', [
  'ngRoute',
  'carAppControllers',
  'carAppServices'
]);

carApp.config(['$routeProvider',
  function($routeProvider){
    $routeProvider.
      when('/cars',{
        templateUrl: 'partials/car-list.html',
        controller: 'CarListCtrl'
      }).
      when('/cars/:carId', {
        templateUrl: 'partials/car-detail.html',
        controller: 'CarDetailCtrl'
      }).
      otherwise({
        redirectTo: '/cars'
      });
}]);
```

car-detail.html

```
<div class="row">
  <div class="col-sm-7 blog-main">
    TODO: DETAIL {{carId}}
  </div>
</div>
```

controllers.js

```
'use strict';

var carAppControllers = angular.module('carAppControllers', []);

carAppControllers.controller('CarListCtrl', ['$scope', 'Car', '$http',
  function($scope, Car, $http){
    $http.get('cars/cars.json').success(function(data) {
      $scope.cars = data;
    });
}]);

carAppControllers.controller('CarDetailCtrl', ['$scope', '$routeParams'],
  function($scope, $routeParams) {
    $scope.carId = $routeParams.carId;
}));
```

## Index.html

```
<html lang="en" ng-app="carApp">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Car App</title>
    <link rel="stylesheet" type="text/css" href="css/bootstrap.css">
    <link rel="stylesheet" type="text/css" href="css/jumbotron-narrow.css">

    <script type="text/javascript" src="js/angular.js"></script>
    <script type="text/javascript" src="js/angular-route.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="js/controllers.js"></script>
    <script type="text/javascript" src="js/services.js"></script>
  </head>

  <body ng-controller="CarListCtrl">
    <div class="blog-masthead">
      <div class="container">
        <nav class="blog-nav">
          <a class="blog-nav-item active" href="#">CarApp</a>
        </nav>
      </div>
    </div>
    <div ng-view class="container"></div>
  </body>
</html>
```



## Servicios RestFul

REST define un set de principios arquitectónicos con los cuales se diseñan servicios web haciendo énfasis en los recursos del sistema, incluyendo la manera de acceder al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes.

Un concepto importante en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso).

Para usarla en angular, se descarga libreria independiente llamada angular-resource.js

Ej:

## Index.html

```
<html lang="en" ng-app="carApp">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>Car App</title>
    <link rel="stylesheet" type="text/css" href="css/bootstrap.css">
    <link rel="stylesheet" type="text/css" href="css/jumbotron-narrow.css">

    <script type="text/javascript" src="js/angular.js"></script>
    <script type="text/javascript" src="js/angular-route.js"></script>
    <script type="text/javascript" src="js/angular-resource.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="js/controllers.js"></script>
    <script type="text/javascript" src="js/services.js"></script>
  </head>

  <body ng-controller="CarListCtrl">
    <div class="blog-masthead">
      <div class="container">
        <nav class="blog-nav">
          <a class="blog-nav-item active" href="#">CarApp</a>
        </nav>
      </div>
    </div>
    <div ng-view class="container"></div>
  </body>
</html>
```

## carro.json(uno por cada carro)

```
{
  "manufacturer": {
    "name": "NISSAN",
    "country": "Japan"
  },
  "engine": {
    "gears": "6",
    "transmission": "Automatic",
    "type": "Diesel"
  },
  "description": "The All New Nissan Qashqai has been harshly tested to provide the ultimate in robustness, reliability and safety.",
  "id": "new-qashqai",
  "images": [
    "img/cars/new_qashqai.jpg",
    "img/cars/new_qashqai2.jpg",
    "img/cars/new_qashqai3.jpg"
  ],
  "name": "New Qashqai",
  "sizeAndWeight": {
    "dimensions": [
      "1866.9 mm (W)",
      "1595.8 mm (H)",
      "4377.4 mm (L)"
    ],
    "weight": "2646.0 Kg"
  },
  "space": {
    "volumen": "430l"
  }
}
```

## car-detail.html(parcial)

```
<div class="blog-header">
  <h1 class="blog-title">{{car.name}}</h1>
  <p class="lead blog-description">{{car.description}}</p>
</div>
<div class="row">
  <div class="col-sm-7 blog-main">
    <div class="cars">
      <li ng-repeat="car in cars | filter:query | orderBy:orderProp">
        <a href="#/cars/{{car.id}}></a>
        <a href="#/cars/{{car.id}}>{{car.name}}</a>
        
      <ul class="car-thumbs">
        <li ng-repeat="img in car.images">
          
        </li>
      </ul>
    </div>
  </div>
  <ul class="specs">
    <li>
      <span>Engine</span>
      <dl>
        <dt>Type</dt>
        <dt>{{car.engine.type}}</dt>
        <dt>Gears</dt>
      </dl>
    </li>
  </ul>
</div>
```

## services.js

```
'use strict';

var carAppServices = angular.module('carAppServices', ['$ngResource']);

carAppServices.factory('Car', ['$resource',
  function($resource){
    return $resource('cars/:carId.json', {}, {
      query: {method: 'GET', params: {carId:'cars'}, isArray:true}
    });
  }]);
```

## Controllers.js

```
'use strict';

var carAppControllers = angular.module('carAppControllers', []);

carAppControllers.controller('CarListCtrl', ['$scope', 'Car',
  function($scope, Car){
    $scope.cars = Car.query();
    $scope.orderProp = 'model';
}]);

carAppControllers.controller('CarDetailCtrl', ['$scope', '$routeParams', 'Car',
  function($scope, $routeParams, Car) {
    $scope.car = Car.get({carId: $routeParams.carId}, function(car){
      $scope.mainImageUrl = car.images[0];
    });
    $scope.setImage = function(imageUrl){
      $scope.mainImageUrl = imageUrl;
    }
}]);
```

## Directivas

Definiéndolas de manera muy general; las directivas son marcadores en un elemento DOM (como un atributo, nombre del elemento, comentario o clase CSS) de Angular para agregar comportamientos específicos a los elementos del DOM.



ngBind



ngModel



ngClass

Con el fin de evitar colisiones con algún estándar futuro, lo mejor es usar el prefijo y sus propios nombres de directivas.

Utilice la opción scope para crear ámbitos independientes al hacer componentes que desea volver a utilizar a lo largo de su aplicación.

Para registrar una directiva, se utiliza el modulo.directive API.modulo

La directiva toma el nombre de la directiva normalizada seguida de una función de fábrica.

Ej: app de cronometro

### App.js

```
'use strict';

var cronoApp = angular.module('cronoApp', [
  'cronoAppControllers'
]).directive('ngClock', function() {
  return {
    restrict: 'A',
    scope: {
      getTime: "="
    },
    template: '<div class="sparkline title clock-font">{{getTime}}</div>',
    controller: ['$scope', '$interval', function($scope, $interval){
      $interval(function(){
        var time = moment(new Date()).format("hh:mm:ss");
        $scope.getTime = time;
      }, 1000);
    }]
  };
});
```

### Index.html

```
<!DOCTYPE html>
<html ng-app="cronoApp">
  <head>
    <link rel="stylesheet" type="text/css" href="css/app.css">
  </head>
  <body>
    <div ng-controller="clockController">
      <div class="container">
        <h1 class="title">Cronometro App</h1>
        <div class="center-block" ng-clock get-time="time"></div>
        <button class="center-block" ng-click="getTimeEnlapse()">GET TIME</button>

        <div class="center-block">
          <div class="fronzen-font title">{{fronzenTime}}</div>
        </div>
      </div>
    </div>
    <script type="text/javascript" src="js/angular.js"></script>
    <script type="text/javascript" src="js/moment.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
    <script type="text/javascript" src="js/controllers.js"></script>
  </body>
</html>
```

### Controllers.js

```
'use strict';

var cronoApp = angular.module('cronoAppControllers', []);

cronoApp.controller('clockController', function($scope){
  $scope.time = moment(new Date()).format("hh:mm:ss");

  $scope.getTimeEnlapse = function(){
    $scope.fronzenTime = $scope.time;
  }
});
```

## Módulos

Usted puede pensar en un módulo como un contenedor para las diferentes partes de su aplicación - controladores, servicios, filtros, directivas, etc.

- ✓ El proceso declarativo es más fácil de entender.
- ✓ Puede empaquetar código y construir módulos reutilizables.
- ✓ Los módulos pueden ser cargados en cualquier orden.
- ✓ Las pruebas unitarias sólo tienen que cargar los módulos pertinentes.
- ✓ Pruebas de extremo a extremo pueden utilizar módulos para anular la configuración.



Bloques de configuración



Bloques iniciales

## Ej: app de películas (sin controladores)

### App.js

```
'use strict';

var moviesBDapp = angular.module('moviesBDapp', []);
.moviesBDapp .run(function ($rootScope){
  $rootScope.orderProp = "name";
  $rootScope.reverse = false;
  $rootScope.movies = [
    {
      name: 'The Shawshank Redemption',
      director: 'Frank Darabont',
      review: 9.3,
      publishYear: 1994
    },
    {
      name: 'Pulp Fiction',
      director: 'Quentin Tarantino',
      review: 8.9,
      publishYear: 1995
    },
    {
      name: 'Interstellar',
      director: 'Christopher Nolan',
      review: 8.9,
      publishYear: 1994
    }
  ];
});
```

### Index.html

```
<head>
  <meta charset="UTF-8">
</head>
<body>
  <div>
    <form class="title">
      <h1>Base de Datos de películas</h1>
      <label>Escoger filtro</label>
      <select ng-model="orderProp">
        <option value="name">Nombre película</option>
        <option value="director">Nombre director</option>
        <option value="review">Calificación</option>
        <option value="publishYear">Año de publicación</option>
      </select>

      <label>Orden</label>
      <select ng-model="reverse">
        <option value="false">t A-Z</option>
        <option value="true">t Z-A</option>
      </select>
    </form>

    <hr>
    <div class="wrapper">
      <div class="container">
        <table>
          <thead>
            <th>Nombre</th>
            <th>Director</th>
            <th>Calificación</th>
            <th>Año de estreno</th>
          </thead>
          <tbody>
            <tr ng-repeat="movie in movies | orderBy:orderProp:reverse">
              <th>{{movie.name}}</th>
              <th>{{movie.director}}</th>
              <th>{{movie.review}}</th>
              <th>{{movie.publishYear}}</th>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
    <script type="text/javascript" src="js/angular.js"></script>
    <script type="text/javascript" src="js/app.js"></script>
  </body>
</html>
```

## Base de Datos de películas

Escoger filtro: Nombre director Orden t A-Z

Nombre	Director	Calificación	Año de estreno
The Matrix	Andy Wachowski	8.7	1999
Interstellar	Christopher Nolan	8.9	2014
City of God	Fernando Meirelles	9.5	2011
The Shawshank Redemption	Frank Darabont	9.3	1994
Pulp Fiction	Quentin Tarantino	8.9	1995
Forrest Gump	Robert Zemeckis	8.8	2002
Saving Private Ryan	Steven Spielberg	8.6	1998

## Ganchos de animación

AngularJS 1.3 proporciona ganchos de animación para las directivas comunes como ngRepeat , ngSwitch y ngView , así como para las directivas personalizadas a través de la sintaxis \$servicio.animate.

Angular JS está atento a los cambios de la clase CSS en elementos mediante la activación del complemento y puede eliminar los ganchos.

Esto significa que si una clase CSS se añade o se elimina de un elemento, una animación puede ser ejecutada en la mitad, antes de la adición de tipo CSS o que la eliminación esté finalizada.

Funciona con la librería independiente llamada angular-animate.js(trabajar con el .min)

## App.js

```
var app = angular.module('app', ['ngAnimate']);

app.controller('controllerAnimacion', function($scope){
    $scope.visible = true;
});

app.animation(".animacion", function(){
    return {
        enter: function(element, done){
            element.text('Hola, cómo estás?');
            element.addClass('efectoAnimacion');
            done();
        },
        leave: function(element, done) {
            element.text('¡Hasta luego!');
            element.addClass('efectoAnimacion');
            done();
        }
    }
});
```

## index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <link rel="stylesheet" type="text/css" href="css/bootstrap.css">

        <script type="text/javascript" src="js/angular.js"></script>
        <script type="text/javascript" src="js/angular-animate.js"></script>
        <script type="text/javascript" src="js/app.js"></script>
    </head>
    <body ng-app="app">
        <div class="container" ng-controller="controllerAnimacion">
            <div class="page-header">
                <h1>Mi primera animación con Angular JS</h1>
            </div>
            <div class="panel panel-primary">
                <div class="panel-heading">Mi animación!</div>
                <div class="panel-body animacion" ng-if="visible">Hola, cómo estás?</div>
            </div>
            <button class="btn btn-lg btn-danger" ng-click="visible=!visible">Iniciar</button>
        </div>
    </body>
</html>
```

## Estilo.css

```
.efectoAnimacion{
    transition: ease-in-out 2s;
    -ms-transition: ease-in-out 2s;
    -o-transition: ease-in-out 2s;
    -webkit-transition: ease-in-out 2s;
    -moz-transition: ease-in-out 2s;
}

.miEstilo{
    font-size: large;
}

.efectoAnimacion.ng-enter{
    opacity: 0;
}
.efectoAnimacion.ng-enter-active {
    opacity: 2;
    font-size: x-large;
}

.efectoAnimacion.ng-leave {
    opacity: 2;
}
.efectoAnimacion.ng-leave-active {
    font-size: small;
    opacity: 0;
}
```

Un grupo común y útil de directivas de Angular soporta los disparadores de las animaciones y la activación de los ganchos durante el ciclo de vida, cuando se producen eventos importantes. Sin embargo, debemos tener en cuenta que tipo de evento es soportado por cual de las directivas.

Directiva	Animaciones compatibles
ngRepeat	entrar, salir, y mover
ngView	entrar y salir
ngInclude	entrar y salir
ngSwitch	entrar y salir
ngIf	entrar y salir
ngClass or	agregar y quitar
ngShow y ngHide	agregar y quitar (el valor de la clase ng-ocultar)

Ej: una animación usando jquery y angular-animation.js  
Animation.js

app.js

```

var carAppAnimations = angular.module('carAppAnimations', ['ngAnimate']);
carAppAnimations.animation('.car', function(){
  var animateUp = function(element, className, done){
    if(className != 'active'){
      return;
    }
    element.css({
      position: 'absolute',
      top: 500,
      left: 0,
      display: 'block'
    });
    jQuery(element).animate({
      top: 0
    }, done);
    return function(cancel){
      if(cancel){
        element.stop();
      }
    };
  };

  var animateDown = function(element, className, done){
    if(className != 'active'){
      return;
    }
    element.css({
      position: 'absolute',
      top: 0
    });
    jQuery(element).animate({
      top: 500
    }, done);
    return function(cancel){
      if(cancel){
        element.stop();
      }
    };
  };
});

```

### animation.css

```

.car-listing.ng-enter,
.car-listing.ng-leave,
.car-listing.ng-move {
  -webkit-transition: 0.5s linear all;
  -moz-transition: 0.5s linear all;
  -o-transition: 0.5s linear all;
  transition: 0.5s linear all;
}

.car-listing.ng-enter,
.car-listing.ng-move{
  opacity: 0;
  height: 0;
  overflow: hidden;
}

.car-listing.ng-move.ng-move-active,
.car-listing.ng-enter.ng-enter-active {
  opacity: 1;
  height: 120px;
}

.car-listing.ng-leave {
  opacity: 1;
  overflow: hidden;
}

```

```

'use strict';

var carApp = angular.module('carApp', [
  'ngRoute',
  'carAppAnimations',
  'carAppControllers',
  'carAppServices'
]);

carApp.config(['$routeProvider',
  function($routeProvider){
    $routeProvider.
      when('/cars',{
        templateUrl: 'partials/car-list.html',
        controller: 'CarListCtrl'
      }).
      when('/cars/:carId', {
        templateUrl: 'partials/car-detail.html',
        controller: 'CarDetailCtrl'
      }).
      otherwise({
        redirectTo: '/cars'
      });
  }]);

```

## Usando \$Location

El servicio \$Location analiza la URL en la barra de direcciones del navegador (basado en el window.location) y hace que la URL esté a disposición de la aplicación construida en Angular JS.



Expone la URL actual en la barra de direcciones del navegador.



Mantiene la sincronización entre el mismo y la URL del navegador cuando el usuario navega.



Representa el objeto URL como un conjunto de métodos.

### Ej: controller.js

```

miApp.controller("controllerRegistroEmpleado", function controllerRegistroEmpleado($scope, $rootScope, $location){
  $scope.irInicio = function(){
    $location.url("/");
  }

  $scope.guardarEmpleado = function(){
    if(localStorage.getItem("empleados") == null){
      $rootScope.empleados = [];
    }else{
      $rootScope.empleados = JSON.parse(localStorage.getItem("empleados"));
    }

    $rootScope.empleados.push({
      nombre: $scope.nuevoEmpleado.nombre,
      apellido: $scope.nuevoEmpleado.apellido,
      salario: $scope.nuevoEmpleado.salario,
      telefono: $scope.nuevoEmpleado.telefono,
      email: $scope.nuevoEmpleado.email
    });

    localStorage.setItem("empleados", JSON.stringify($rootScope.empleados));
    $scope.nuevoEmpleado = null;
  };
});

miApp.controller("controllerListaEmpleados", function controllerListaEmpleados($scope, $rootScope, $location){
  $scope.irInicio = function(){
    $location.url("/");
  }
});

```

Index.html

### app.js

```

var miApp = angular.module('miApp',['ngRoute']);

miApp.config(function($routeProvider){
  $routeProvider.when("/",{
    templateUrl: "vistas/inicio.html",
    controller: "controllerInicio"
  }).
  when("/listaEmpleados",{
    templateUrl: "vistas/listaEmpleados.html",
    controller: "controllerListaEmpleados"
  }).
  when("/registroEmpleado",{
    templateUrl: "vistas/registroEmpleado.html",
    controller: "controllerRegistroEmpleado"
  }).
  otherwise({redirectTo: "/" });
})

```

lista.empleado.html

```

<html ng-app="miApp">
</head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
<script type="text/javascript" src="js/angular.js"></script>
<script type="text/javascript" src="js/angular-route.js"></script>
<script type="text/javascript" src="js/app.js"></script>
<script type="text/javascript" src="js/controllers.js"></script>

</head>
<body>
<div class="container">
<nav class="navbar navbar-inverse">
<div class="container-fluid">
<div class="navbar-header">
<a class="navbar-brand" href="#">Gestion Empleados</a>
</div>
<div class="nav navbar-nav">
<ul>
<li><a href="#">Inicio</a></li>
<li><a href="#/registroEmpleado">Registro Empleado</a></li>
<li><a href="#/listaEmpleados">Listado Empleado</a></li>
</ul>
</div>
</div>
</nav>
<ng-view></ng-view>
</div>
</body>
</html>

```

```

<div class="container">
<p ng-show="empleados.length == 0">No hay Empleados para Mostrar</p>
<div class="list-group">
<a href="#" class="list-group-item active">Lista Empleados</a>
<a href="#" data-ng-repeat="empleado in empleados" class="list-group-item">
<b>Nombre:</b>{{empleado.nombre}}<br>
<b>Apellido:</b>{{empleado.apellido}}<br>
<b>Salario:</b>{{empleado.salario}}<br>
<b>Telefono:</b>{{empleado.telefono}}<br>
<b>E-mail:</b>{{empleado.email}}<br>
</b>
</div>
<button class="btn btn-sucess" ng-click="irInicio()">Inicio</button>
</div>

```

## HTML Compiler

El compilador HTML de Angular le permite al desarrollador incluir la nueva sintaxis HTML en el navegador. El compilador permite adjuntar un comportamiento a cualquier elemento HTML o atributo e incluso crear nuevos elementos HTML o atributos para obtener comportamiento personalizado.

# NODE. JS

- ✓ Es una plataforma de software basada en el Motor V8 de Chrome.
- ✓ Permite crear aplicaciones escalables en tiempo real:
  - ✓ Servidor - Cliente
  - ✓ Servidor - Servidor
- ✓ Orientado a eventos:
  - ✓ Event Queue
  - ✓ Event Loop
- ✓ MonoHilo
- ✓ Thread Pool – libuv
- ✓ Non-Blocking – No Bloqueante

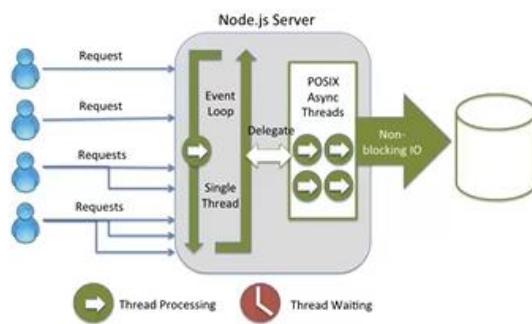
## ¿Para qué sirve?



- ✓ Aplicaciones con uso intensivo de datos en tiempo real.
- ✓ Juegos multi-player.
- ✓ Aplicaciones colaborativas.
- ✓ Analíticas y estadísticas.
- ✓ APIS REST entre servidor-cliente, servidor-servidor.
- ✓ Múltiples usos más.

## ¿Cómo funciona?

Non-Blocking



## Desarrollar aplicaciones

Ejemplos:

```
var http = require('http');
http.createServer(function(request, response){
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hola Mundo');
}).listen(3000);
console.log("Servidor Corriendo en http://127.0.0.1:3000");
```

```
1 var http = require('http');
2 var url = require('url');
3
4 http.createServer(function(request, response){
5   response.writeHead(200, {'Content-Type': 'text/plain'});
6   var params = url.parse(request.url, true).query;
7   var numero = params.numero;
8   response.write("Descendiendo "+numero+" a 0: \n");
9
10  for(var i = numero; i>=0; i--){
11    response.write("-"+i+"\n");
12  }
13  response.end();
14 }).listen(3000);
15 console.log("Servidor ")
```

cmd: node ejemplo.js

```
var http = require('http');
var url = require('url');

http.createServer(function(request, response){
  response.writeHead(200, {'Content-Type': 'text/plain'});
  var params = url.parse(request.url, true).query;
  var nombre = params.nombre;
  response.end('Hola '+nombre);
}).listen(3000);
console.log("Servidor corriendo en http://127.0.0.1:3000");
```

```
var http = require('http');
var url = require('url');

http.createServer(function(request, response){
  response.writeHead(200, {'Content-Type': 'text/plain'});
  var params = url.parse(request.url, true).query;
  var numero = params.numero;
  response.write("Descendiendo "+numero+" a 0: \n");
  var descendenteSync = function(numero, callback){
    for(var i=numero; i>=0; i--){
      callback(i);
    }
  }
  descendenteSync(numero, function(i){
    response.write("-"+i+"\n");
  });
  response.end();
}).listen(3000);
console.log("Servidor Corriendo");
```

# Tipos de Servers y Sockets



- ✓ HTTP – Server
- ✓ TCP – Server
- ✓ TLS – Server
- ✓ Socket UDP – Socket

## Servidor TCP

Diferencias con HTTP

- ✓ HTTP  
`var http = require('http');`
- ✓ TCP  
`var net = require('net');`

## Servidor TCP

Funcionalidades

- Escuchar cuando un socket cliente:
  - ✓ Se conecte
  - ✓ Envíe un dato
  - ✓ Cuando se desconecte

## Conversación entre cliente y servidor en Node JS

Ej: servidor.js

```
var net = require('net');
var server = net.createServer(function(con){
    con.on('data', function(chunk){
        console.log('Datos Enviados desde el Cliente:' + chunk);
        con.write('Repetimos: ' + chunk);
    });

    con.on('close', function(){
        console.log('Cliente cerró Conexión!');
    });

    con.on("error", function(err){
        console.log("Se ha Perdido la Conexión con el Cliente");
    });
});

}).listen(8000);
console.log("Escuchando Servidor en el Puerto 8000");
```

cliente.js

```
var net = require('net');
var client = net.connect({port: 8000}, function () {
    client.setEncoding('utf8');
    console.log('Conectado');
    client.write('Hola Servidor');
});

process.stdin.resume();

process.stdin.on('data', function(chunk){
    client.write(chunk);
});

client.on('data', function(chunk){
    console.log(chunk);
});

client.on('close', function(){
    console.log('Se Cerró la Conexión');
});

client.on('error', function(err){
    console.log('Se ha Perdido la Conexión con el Servidor');
});
```

## Streams

Una de las promesas de Node.js es ser altamente óptimo para procesar altas cantidades de datos, de una manera no bloqueante y en tiempo real. La razón principal para sostener esto es su **Interfaz Abstracta Stream**. Un **Stream** puede ser **Writable** o **Readable**, o ambos (**Duplex**). Lo que permite el intercambio de información fluida.

Proceso óptimo de datos

Asíncrono

Paralelo

No bloqueante

Tiempo real

Ej:

```
var http = require('http');

http.createServer(function(request, response){
    response.writeHead(200);

    request.on('data', function(chunk){
        console.log(chunk.toString());
        response.write(chunk);
    });
    request.on('end', function(){
        response.end();
    });
}).listen(8000);
console.log("Servidor Corriendo en el Puerto 8000");
```

cmd: curl -d 'Hola servidor' http://localhost:8000

Ej: subiendo archivos o imágenes

```
var fs = require('fs');
var http = require('http');

http.createServer(function(request, response){
    var new_file = new fs.createWriteStream('nueva_imagen.jpg');
    var bytes_totales = request.headers['content-length'];
    var bytes_subidos = 0;

    request.pipe(new_file);

    request.on('data', function(chunk){
        bytes_subidos += chunk.length;
        var progreso = (bytes_subidos / bytes_totales)*100;
        response.write("progress: " + parseInt(progreso, 10) + "%\n");
    });

    request.on('end', function(){
        response.end('Carga Completa!');
    });
}).listen(8000);
console.log('Servidor Corriendo en el Puerto 8000');
```

## Timers

**Los Timers** son un conjunto de funciones globales. No necesitan ser requeridas para usarlas. Tienen como fin controlar, en cuanto a tiempo, el flujo y orden de ejecución de una o más funciones. Las más comunes son: setTimeout, setInterval, clearInterval, clearTimeout.

### setTimeout

- ✓ Pausa de tiempo antes de ejecutar una función.
- ✓ Retorna un **timeoutObject**.

### clearTimeout

- ✓ Detiene un Timeout.
- ✓ Recibe un **timeoutObject** como argumento para detenerlo.

### setInterval

- ✓ Intervalo de tiempo en que continuamente se ejecuta un procedimiento.
- ✓ Retorna un **intervalObject**.

### clearInterval

- ✓ Detiene un Interval.
- ✓ Recibe un **intervalObject** como argumento para detenerlo.

Ej:

```
process.stdin.setEncoding('utf8');

process.stdin.on('data', function(chunk){
  if(isNaN(chunk)){
    decremento(chunk);
  }else{
    process.stdout.write('No es un numero valido');
  }
});

process.stdin.on('end', function(){
  process.stdout.write('end');
});

function decremento(numero){
  var intervalo = setInterval(function (){
    numero-=1;
    process.stdout.write(numero+"\n");
    if(numero == 0){
      process.stdout.write("Secuencia Terminada!");
      clearInterval(intervalo);
    }
  }, 1000);
}
```

## Módulos y NPM

**Los módulos** en Node.js son objetos que extienden su funcionalidad mediante un prototipo. Esto se hace para evitar tener funcionalidades separadas de un mismo contexto.

Como funcionan

- ✓ Funciones contextualizadas a ser reutilizadas.
- ✓ Los módulos pueden ser reutilizados.
- ✓ Distribuir sus funciones a nuestro acomodo.

como se buscan

- require("./mi\_modulo"); - En el mismo directorio
- require("../mi\_modulo"); - En el directorio pariente contenedor
- require("http"); - En el directorio node\_modules

## NPM (Node Package Manager)



- ✓ Se instala con Node.js.
- ✓ Tiene un repositorio de Módulos: <http://npmjs.org>.
- ✓ Manejo de dependencias.
- ✓ Fácil para publicar módulos.
- ✓ Instala módulos de manera local y global.

```
exports.queso = function(){
  return 'Queso';
}

exports.chorizo = function(){
  return 'Chorizo';
}

exports.salami = function(){
  return 'Salami';
}

exports.jamon = function(){
  return 'Jamon';
}
```

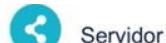
npm install "paquete" – Instalar un paquete específico

npm init- Permite crear un package.json

## Express.js

**Express.js** – Framework de Node.js, rápido, minimalista y no convencional. Especializado en crear aplicaciones web y APIs Restful.

- ✓ Global  
**npm install -g express**



Servidor

- ✓ Local  
**npm install express --save**



Modelos



Controladores



Rutas



Vistas



package.json

Ej:

```
var express = require('express'),
app = express(),
server = require('http').createServer(app);

app.get('/', function(req, res){
  res.send("Hola Mundo");
});

var port = Number(process.env.PORT || 3000);

server.listen(port, function(){
  console.log('Servidor corriendo en: ' + port);
});
```

Ej: mostrar un texto

```
var express = require('express'),
app = express(),
server = require('http').createServer(app);

var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var path = require('path');
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended:false}));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.get('/', function(req, res){
  res.send("Hola Mundo");
});

var port = Number(process.env.PORT || 3000);

server.listen(port, function(){
  console.log('Servidor corriendo en: ' + port);
})
```

cmd: npm start

Routing en Express JS

**Routing en Express** es la técnica utilizada para responder a peticiones HTTP de una aplicación. La manera para definir rutas es a través de los verbos HTTP. Los más conocidos son GET y POST, en segundo lugar DELETE y PUT, y como menos populares OPTIONS, HEAD y TRACE.

## GET

- ✓ Usado para obtener un recurso.
- ✓ http://ejemplo.com/acerca
- ✓ http://ejemplo.com/usuario/1
- ✓ Éxito: 200 (Ok).
- ✓ Error: 404 (Not found) – 400 (Bad request).
- ✓ Usado para crear un recurso.
- ✓ http://ejemplo.com/create/user
- ✓ Éxito: 201 (Created)
- ✓ Error: 404 (Not found)

Ej:

```
var express = require('express'),
app = express(),
server = require('http').createServer(app);

var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var path = require('path');
var fs = require('fs');
var _ = require('lodash');

app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.get('/', function(req, res){
  res.write('Hola Mundo');
  res.end();
});

app.get('/usuarios', function(req, res){
  var stream = fs.createReadStream(__dirname+'/base_datos.json');
  stream.setEncoding('utf8');
  stream.pipe(res);
});

app.post('/usuario', function(req, res){
  var user = {
    "_id": "54b73e7e14f84d59250577c7",
    "age": 28,
    "name": "Nuevo Usuario",
    "gender": "male",
    "email": "carpenterayers@xumonk.com",
    "phone": "+1 (933) 527-3679"
  };

  var datos = [];
  var streamR = fs.createReadStream(__dirname+'/base_datos.json');
  streamR.setEncoding('utf8');

  streamR.on('data', function(chunk){
    chunk = JSON.parse(chunk);
    chunk.push(user);
    datos = chunk.slice();
  });

  streamR.on('end', function(){
    var streamW = fs.createWriteStream(__dirname+'/base_datos.json');
    streamW.write(JSON.stringify(datos));
    res.send(datos);
  });
});

app.put('/usuario', function(req, res){
  var id = "54b73e7e14f84d59250577c7";

  var datos = [];
  var streamR = fs.createReadStream(__dirname+'/base_datos.json');
  streamR.setEncoding('utf8');

  streamR.on('data', function(chunk){
    chunk = JSON.parse(chunk);
    var index = _.findIndex(chunk, {'_id':id});
    chunk[index].name = "Nombre Editado";
    datos = chunk.slice();
  });

  streamR.on('end', function(){
    var streamW = fs.createWriteStream(__dirname+'/base_datos.json');
    streamW.write(JSON.stringify(datos));
    res.send(datos);
  });
});

app.delete('/usuario', function(req, res){
  var id = "54b73e7e14f84d59250577c7";

  var datos = [];
  var streamR = fs.createReadStream(__dirname+'/base_datos.json');
  streamR.setEncoding('utf8');

  streamR.on('data', function(chunk){
    chunk = JSON.parse(chunk);
    var index = _.findIndex(chunk, {'_id':id});
    console.log(index);
    delete chunk[index];
    datos = _.compact(chunk.slice());
  });

  streamR.on('end', function(){
    var streamW = fs.createWriteStream(__dirname+'/base_datos.json');
    streamW.write(JSON.stringify(datos));
    res.send(datos);
  });
});

var port = Number(process.env.PORT || 3000);
server.listen(port, function(){
  console.log('Servidor Corriendo en: ' + port);
});
```

**Motor de plantillas Swig**

- ✓ Motor de plantillas no muy alejado del HTML común.
- ✓ Instalación npm install swig –save

**Render HTML** es la lectura y evaluación de un template mediante JavaScript y traducido finalmente a HTML común.

## Controladores

**Los controladores** son un conjunto de funciones encargadas de intermediar los datos entre el modelo y la vista, a través de las diferentes rutas que exija nuestra lógica del negocio.

**Request**, Alias req, es el mismo objeto request del módulo HTTP de Node.js

**Response**, Alias res, es el mismo objeto response del módulo HTTP de Node.js

**Next()**: un controlador está asignado a una ruta, y en más de una ocasión tiene un conjunto de funciones encargadas de diferentes procesos de datos en el ciclo request-response. El orden de ejecución de los middlewares será el mismo orden de registro, por lo que su forma de ejecución será en serie. Por tal razón cuando un middleware finaliza, es necesario llamar el siguiente middleware con una función llamada next().

## Store y Sessions

### Base de datos MongoDB

**MongoDB** es una base de datos orientada a documentos, que sigue el paradigma NoSQL.

- ✓ Almacenamiento orientado a documentos.
- ✓ Querying JavaScript.
- ✓ Soporte de índices.
- ✓ Replicación y alta disponibilidad.

### Instalar la base de datos MongoDB

```
iMac-de-Desarrollo:~ Desarrollo@15 mongo
MongoDB shell version: 3.0.2
connecting to: test
> db
test
> show collections
> show dbs
local 0.078GB
> use mydb
switched to db mydb
> db
mydb
> show collections
> db.users({username:"mviera", age:26, city:"Sevilla"})
2015-04-20T10:52:00.623+0500 E QUERY    TypeError: Property 'users' of object my
db is not a function
at (shell):1:4
> db.users.insert({username:"mviera", age:26, city:"Sevilla"})
WriteResult({ "nInserted" : 1 })
> db.users.insert({username:"robot", age:32, city:"Cadiz"})
WriteResult({ "nInserted" : 1 })
> db.users.insert({username:"robot", age:32, city:"Co@iz"})
```

## Mongoose

**Mongoose:** es una librería que utiliza el plugin original de NODE.JS para mongo, y permite modelar de una forma simple y elegante.

**Plugins:** tiene un conjunto de plugins o librerías tercera, que extienden su funcionalidad y hacen mucho más fácil algunos procesos de datos localizados en su sitio web.

**Schemas:** Mongoose nos ofrece un conjunto de herramientas con el fin de enriquecer nuestra experiencia en MongoDB. A partir de Schemas, podemos definir la estructura de una colección en formato JSON. Esto nos permite identificar por defecto el contenido de una colección.

Ej:

```
model.js          routes.js      controller.js      routes.js      Servidor.js
1 var mongoose = require('mongoose');
2 mongoose.connect('mongodb://localhost/prueba1');
3 module.exports = mongoose;
```

## Redis

**Redis:** Es un sistema de almacenamiento de datos NoSQL basado en una estructura de llave-valor. Su almacenamiento se da en la memoria RAM del servidor, como un tipo de cacheo preventivo, con el fin de acelerar la consulta de datos.

# Tipos de Datos

- ✓ Cadenas de caracteres
- ✓ Listas
- ✓ Juegos de sets (conjuntos)
- ✓ Juegos ordenados o Sorted Sets (conjuntos ordenados)
- ✓ Hashes

Ej:

```
(nil)
127.0.0.1:6379> SET llaveNumerica 1
OK
127.0.0.1:6379> INCR llaveNumerica
(integer) 2
127.0.0.1:6379> INCRBY llaveNumerica 5
(integer) 7
127.0.0.1:6379> EXPIRE llaveNumerica 20
(integer) 1
127.0.0.1:6379> TTL llaveNumerica
(integer) 6
127.0.0.1:6379> SADD juegos ajedrez
(integer) 1
127.0.0.1:6379> SADD juegos ludo
(integer) 1
127.0.0.1:6379> SADD juegos monopolio
(integer) 1
127.0.0.1:6379> SMEMBERS juegos
1) "ludo"
2) "monopolio"
3) "ajedrez"
127.0.0.1:6379> SREM juegos ajedrez
(integer) 1
127.0.0.1:6379> SISMEMBER juegos ajedrez
```

## Sistema de autenticación PassportJS Local

**PassportJS**: es un middleware de autenticación para Node.js, extremadamente flexible y modular. Basado en estrategias de autenticación local y social

## Características

- ✓ Más de 140 estrategias de autenticación.
- ✓ Implementación de OpenID y Oauth.
- ✓ Fácil manejo de errores y éxitos.
- ✓ Soporta sesiones persistentes.



## Flash Messages



- ✓ Soportado por PassportJS.
- ✓ Enviar mensajes cuando ocurra.
- ✓ Exclusivo para mensajes de error y éxito.

### Sistema de autenticación PassportJS Social

#### ✓ Facebook:

- ✓ Creamos una app: <https://developers.facebook.com>
- ✓ `npm install passport-facebook`

#### ✓ Twitter:

- ✓ Creamos una app: <https://apps.twitter.com>
- ✓ `npm install passport-twitter`

## RealTime

### El Websocket y el Socket.IO

**Websocket:** Es una tecnología popularizada con la llegada de HTML5, que proporciona un canal bidireccional y full-dúplex sobre un único socket TCP.

Es la encargada de hacer el sueño del Realtime una realidad óptima, y amigable al desarrollador.

## ¿Como funciona?

- ✓ El cliente le pide al servidor que quiere iniciar una conexión.
- ✓ Se valida la identidad con unos datos secretos.
- ✓ El servidor acepta, enviándole un mensaje.
- ✓ Dejan de utilizar HTTP y pasan al protocolo Websocket.

**Socket.IO:** Es una librería que nos facilita el uso de Websockets en NodeJS. Nos entrega un API para nuestro servidor y para nuestro cliente.

### Funciones para la comunicación de sockets

## On y Emit

Funciones para la comunicación de sockets:



- ✓ `Socket.on("evento", function(){})`
- ✓ `Socket.emit("evento", dato)`
- ✓ `Socket.broadcast.in("room").emit("evento", dato)`

### Socket.IO con Redis

## Socket.IO y Redis

Envío y persistencia de datos.



`Client.lpush("mensajes", dato, function(){})`



`Client.ltrim("mensajes", 0, 1);`



`Client.lrange("mensajes", 0, 1, function(){})`

**Lpush:** Guardamos los datos en nuestro cliente Redis.

**Ltrim:** Lo usamos para mantener un límite de solo diez mensajes en nuestra base de datos Redis. Siempre se elimina el último después del decimo mensaje.

**Rrange:** Lo usamos para devolver un rango entre 0 y todos los demás mensajes existentes. Y posteriormente emitirlos a los nuevos usuarios conectados.

## Passport, Socket.IO y Redis

### Passport

Cumple el rol de la gestión de sesiones.



passport.use('user', new LocalStrategy({}));



passport.authenticate('user', ... );



passport.use('user', new TwitterStrategy({}));

**LocalStrategy:** Utilizamos la estrategia local de passport, con la que podemos gestionar la sesión de un usuario que ingresa su usuario y contraseña.

**Passport.authenticate:** Autenticamos nuestra estrategia y validamos hacia donde se dirige en caso de éxito o error.

**TwitterStrategy:** Utilizamos la estrategia social de passport para iniciar sesión por medio de Twitter.

### GIT

#### configuracion

```
User@DESKTOP-8CQ3NWF MINGW32 ~
$ git config --global user.name "gabriel"

User@DESKTOP-8CQ3NWF MINGW32 ~
$ git config --global user.mail "franidos@yahoo.es"
```

## Clonar proyectos

```
User@DESKTOP-8CQ3NWF MINGW32 ~/Documents/Git  
$ git clone https://github.com/mariaclaraNextu/Cenicienta.git
```

## Arquitectura MEAN

- ✓ Implementar la estructura de un proyecto basado en MEAN Stack.
- ✓ Identificar los roles de cada una de las tecnologías utilizadas (MongoDB, Express JS, Angular JS, Node JS).
- ✓ Delegar el manejo de rutas del servidor al cliente mediante la herramienta "ui.router" de Angular JS.
- ✓ Implementar la estructura de las bases de datos mediante "Schemas" con la herramienta "Mongoose JS".

## El Full Stack usando MEAN

**Full Stack JavaScript** – Cuando hablamos de Full Stack JavaScript nos referimos a usar JavaScript en todas las partes que componen una aplicación web actual (Frontend, Backend, Base de datos). A esto también le llamamos JavaScript de extremo a extremo.

**MEAN** - Existen diversas formas de implementar Full Stack JavaScript, una de las mas conocidas y utilizadas es **MEAN**.

El acrónimo **MEAN** viene de (**M**ongo + (**E**xpress + (**A**ngular + (**N**ode.

## MEAN

- ✓ **MongoDB**: base de datos no relacional que guarda los datos en documentos de tipo JSON (JavaScript Object Notation).
- ✓ **ExpressJS**: framework que nos permite utilizar funcionalidades de Node.js de manera mas sencilla.
- ✓ **AngularJS**: framework del lado del cliente o Frontend que representa el paradigma MVC.
- ✓ **Node.JS**: plataforma que nos permite programar en JavaScript para el Backend.

## Modulación MEAN I

```
Sudo npm Install -g bower
```

```
bower init
```

package.json

```
1 {  
2   "name": "Modulacion",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "Servidor.js",  
6   "scripts": {  
7     "test": "null"  
8   },  
9   "author": "",  
10  "license": "ISC",  
11  "dependencies": {  
12    "body-parser": "^1.12.3",  
13    "express": "^4.12.3",  
14    "morgan": "1.5.2",  
15    "swig": "1.4.2"  
16  }  
17 }
```

bower.json

```
1 {  
2   "name": "Modulacion",  
3   "version": "0.0.0",  
4   "authors": [  
5     "NextU"  
6   ],  
7   "license": "MIT"  
8 }
```

## Ejercicio

Servidor.js

```
package.json      Servidor.js  
1 var express = require('express'),  
2 app = express(),  
3 server = require('http').createServer(app);  
4  
5 var swig = require('swig');  
6 var logger = require('morgan');  
7 var bodyParser = require('body-parser');  
8  
9 app.engine('html', swig.renderFile);  
0 app.set('view engine', 'html');  
1 app.set('views', __dirname+ '/views');  
2  
3 app.set('view cache', false);  
4 swig.setDefaults({cache: false});  
5 app.use(logger('dev'));  
6 app.use(bodyParser());  
7 app.use(express.static(__dirname+ '/public'));  
8  
9 app.get('/partials/*', function(req, res){  
0   console.log(req.params);  
1   res.render('../public/app/' + req.params['0']);  
2 });  
3  
4 app.get('*', function(req, res){  
5   res.render('index');  
6 });  
7  
8 server.listen(3000, function(){  
9   console.log("Servidor corriendo en el puerto: 3000");  
0 });
```

Index.html

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8">  
<title>Team App | {% block title %} {% endblock %}</title>  
<link rel="stylesheet" type="text/css" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css" />  
<link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/font-awesome/4.2.0/css/font-awesome.min.css" />  
<link rel="stylesheet" type="text/css" href="https://code.ionicframework.com/1.5.2/css/ionicons.min.css" />  
<link rel="stylesheet" type="text/css" href="/css/AdminLTE.css">  
</head>  
<body class="skin-blue">  
  <header class="header">  
    <a href="#" class="logo">TeamApp</a>  
    <nav class="navbar navbar-static-top" role="navigation">  
      <a href="#" class="navbar-btn sidebar-toggle" data-toggle="offcanvas" role="button">  
        <span class="sr-only">Toggle Navigation</span>  
        <span class="icon-bar"></span>  
        <span class="icon-bar"></span>  
        <span class="icon-bar"></span>  
      </a>  
      <div class="navbar-right">  
        <ul class="nav navbar-nav">  
          <li class="dropdown user user-menu">  
            <a href="#" class="dropdown-toggle" data-toggle="dropdown" aria-expanded="false">  
              <i class="glyphicon glyphicon-user"></i>  
              <span>Fulano de Tal<i class="caret"></i></span>  
            </a>  
            <ul class="dropdown-menu">  
              <li class="user-header bg-light-blue"></li>
```

```

        
        <p>Fulanito de Tal - Fulahito</p>
    </li>
    <li class="user-footer">
        <div class="pull-right">
            <a href="#" class="btn btr-default btn-flat">Cerrar Sesión</a>
        </div>
    </li>
</ul>
</li>
</ul>
</div>
</nav>
</header>
<div class="wrapper row-offcanvas row-offcanvas-left">
    <aside class="left-side sidebar-offcanvas">
        <section class="sidebar">
            <div class="user-panel">
                <div class="pull-left image">
                    
                </div>
                <div class="pull-left info">
                    <p>Hello, Fulanito de Tal</p>
                    <a href="#"><i class="fa fa-circle text-success"></i>Online</a>
                </div>
            </div>
            <ul class="sidebar-menu">
                <li class="">
                    <a ui-sref="app.dashboard">
                        <i class="fa fa-dashboard"></i><span>Dashboard</span>
                    </a>
                <li class="">
                    <a ui-sref="app.chat">
                        <i class="fa fa-comment"></i><span>Dashboard</span>
                    </a>
                <li class="">
                    <a ui-sref="app.tareas">
                        <i class="fa fa-check"></i><span>Tareas</span>
                    </a>
                <li class="">
                    <a ui-sref="app.recursos">
                        <i class="fa fa-envelope"></i><span>Recursos</span>
                    </a>
                </ul>
            </section>
        </aside>
    <aside class="right-side">
        <section class="content-header">
            <h1>Dashboard<small>Control Panel</small></h1>
        </section>
        <section class="content">
        </section>
    </aside>
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
|
```

## Modulación MEAN II

Express.js

```

express.js
1 var logger = require('morgan');
2 var bodyParser = require('body-parser');
3 var swig = require('swig');
4 var express = require('express');
5
6 module.exports = function(app, config){
7     app.engine('html', swig.renderFile);
8     app.set('view engine', 'html');
9     app.set('views', config.rootPath + '/server/views');
10    app.set('view cache', false);
11    swig.setDefaults({cache: false});
12
13    app.use(logger('dev'));
14    app.use(bodyParser());
15    app.use(express.static(config.rootPath + '/public'));
16
17};
```

Routes.js

```

routes.js
module.exports = function(app){
    app.get('/partials/*', function(req, res){
        res.render('../public/app/' + req.params['0']);
    });

    app.get('*', function(req, res){
        res.render('index');
    });
};
```

Servidor.js

```

var express = require('express'),
app = express(),
server = require('http').createServer(app);

var config = {
  rootPath : __dirname
};

require('./server/config/express')(app, config);
require('./server/config/routes')(app);

server.listen(3000, function(){
  console.log("Servidor Corriendo en el puerto: 3000");
});

```

### Scritps.html

```

<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
<script type="text/javascript" src="/app/app.js"></script>
<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>

```

### Link.html

```

<meta charset="utf-8">
<title>Team App | {% block title %} {% endblock %}</title>
<link rel="stylesheet" type="text/css" href="//maxcdn.bootstrapcdn.com/bootstrap/2.2.1/css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/font-awesome/4.2.0/css/font-awesome.min.css">
<link rel="stylesheet" type="text/css" href="//code.ionicframework.com/ionicons/1.5.2/css/ionicons.min.css">
<link rel="stylesheet" type="text/css" href="/css/AdminLTE.css">

```

### Layout.html

```

<!DOCTYPE html>
<html ng-app="Teamapp">
<head>
  {% include 'links.html' %}
</head>
<body class="skin-blue">
  {% block content %}
  {% endblock %}
  {% include 'scripts.html' %}
</body>
</html>

```

### index.html

```

  {% extends '../includes/layout.html' %}

  {% block content %}
    <div id="wrap" ui-view>
    </div>
  {% endblock %}

```

### App.js

```

var app = angular.module('Teamapp', ['ui.router']);

app.config(['$stateProvider', "$urlRouterProvider", function($stateProvider, $urlRouterProvider){
  $urlRouterProvider.otherwise('/app');
  $stateProvider.state('app',{
    url: '/app',
    templateUrl : 'partials/index/templates/index.html',
    controller: 'indexCtrl'
  });
}]);

```

### indexCtrl.js

```

var app = angular.module('Teamapp');

app.controller('indexCtrl', function(){

});

```

### Sidebar.html

```

<section class="sidebar">
  <div class="user-panel">
    <div class="pull-left image">
      
    </div>
    <div class="pull-left info">
      <p>Hello, Fulano de Tal</p>
      <a href="#"><i class="fa fa-circle text-sucess"></i>Online</a>
    </div>
  </div>
  <ul class="sidebar-menu">
    <li class="">
      <a href="#">
        <i class="fa fa-dashboard"></i><span>Dashboard</span>
      </a>
    </li>
    <li class="">
      <a href="#">
        <i class="fa fa-comment"></i><span>Chat</span>
      </a>
    </li>
    <li class="">
      <a href="#">
        <i class="fa fa-check"></i><span>Tareas</span>
      </a>
    </li>
    <li class="">
      <a href="#">
        <i class="fa fa-envelope"></i><span>Dashboard</span>
      </a>
    </li>
  </ul>
</section>

```

## Navbar.html

```

<nav class="navbar navbar-static-top" role="navigation">

  <a href="#" class="navbar-btn sidebar-toggle" data-toggle="offcanvas" role="button">
    <span class="sr-only">Toggle Navigation</span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
  </a>
  <div class="navbar-right">
    <ul class="nav navbar-nav">
      <li class="dropdown user user-menu">
        <a href="#" class="dropdown-toggle" data-toggle="dropdown" aria-expanded="false">
          <i class="glyphicon glyphicon-user"></i>
          <span>Fulano de Tal <i class="caret"></i></span>
        </a>
        <ul class="dropdown-menu">
          <li class="user-header bg-light-blue">
            
            <p>Fulano de Tal - Fulanito</p>
          </li>
          <li class="user-footer">
            <div class="pull-right">
              <a href="#" class="btn btn-default btn-flat">Cerrar Sesión</a>
            </div>
          </li>
        </ul>
      </li>
    </ul>
  </div>
</nav>

```

## Index.html(/templates)

```

<header class="header">
  <a href="#" class="logo">TeamApp</a>
  <ng-include src="'/partials/includes/navbar.html'"></ng-include>
</header>

<div class="wrapper row-offcanvas row-offcanvas-left">
  <aside class="left-side sidebar-offcanvas">
    <ng-include src="'/partials/includes/sidebar.html'"></ng-include>
  </aside>
</div>

<aside class="right-side">
  <section class="content-header">
    <h1>Dashboard<small>Control Panel</small></h1>
  </section>
  <section class="content" ui-view>
  </section>
</aside>

```

## Conexión de rutas y vistas en Angular y Node

**Funcionamiento rutas y vistas:** Dada una ruta o petición (GET), esta es procesada por **Express.js**, quien renderiza una vista.

**Angular.js** realiza un proceso similar, sólo que en este caso es NG-ROUTE el módulo de angular por defecto encargado de gestionar el enrutamiento y el enlace a vistas. Además de NG-ROUTE, existe otro gestor de enrutamiento desarrollado por Angular UI, llamado UI-ROUTER, que nos permite extender algunas funcionalidades de NG-ROUTE.

## Ejercicio

### Express.js

```
var logger = require('morgan');
var bodyParser = require('body-parser');
var swig = require('swig');
var express = require('express');

module.exports = function(app, config){
    app.engine('html', swig.renderFile);
    app.set('view engine', 'html');
    app.set('views', config.rootPath + '/server/views');

    app.set('view cache', false);
    swig.setDefaults({cache: false, varControls: ['{', '}']}); 

    app.use(logger('dev'));
    app.use(bodyParser());
    app.use(express.static(config.rootPath + '/public'));
};


```

### indeCtrl.js

```
var app = angular.module('Teamapp');

app.controller('indexCtrl', function($rootScope, $state, $scope){
    $scope.modulo = new Modulo($state.current).getName();

    function Modulo(state){
        this.state = state.name;
        this.name = state.name.split('.')[1];
        this.getName = function(){
            return this.name[0].toUpperCase() + this.name.slice(1);
        };
    }

    $rootScope.$on("$stateChangeStart", function(event, toState, toParams, fromState, fromParams){
        $scope.modulo = new Modulo(toState).getName();
        console.log(toState);
    });
});


```

### Sidebar.html

```
<section class="sidebar">

<div class="user-panel">
    <div class="pull-left image">
        
    </div>
    <div class="pull-left info">
        <p>Hello, Fulano de tal</p>
        <a href="#"><i class="fa fa-circle text-success"></i> Online</a>
    </div>
</div>

<ul class="sidebar-menu">
    <li class="">
        <a ui-sref="app.dashboard">
            <i class="fa fa-dashboard"></i> <span>Dashboard</span>
        </a>
    </li>
    <li class="">
        <a ui-sref="app.chat">
            <i class="fa fa-comment"></i> <span>Chat</span>
        </a>
    </li>
    <li class="">
        <a ui-sref="app.tareas">
            <i class="fa fa-check"></i> <span>Tareas</span>
        </a>
    </li>
    <li class="">
        <a ui-sref="app.recursos">

```

### App.js

```

var app = angular.module('Teamapp',['ui.router']);

app.config(['$stateProvider', '$urlRouterProvider', function($stateProvider, $urlRouterProvider){
    $urlRouterProvider.otherwise('/app/dashboard');

    $stateProvider
        .state('app',{
            url : '/app',
            templateUrl : 'partials/index/templates/index.html',
            controller : 'indexCtrl'
        })
        .state('app.dashboard',{
            url : '/dashboard',
            templateUrl : 'partials/dashboard/templates/dashboard.html',
        })
        .state('app.chat',{
            url : '/chat',
            templateUrl : 'partials/chat/templates/chat.html',
        })
        .state('app.tareas',{
            url : '/tareas',
            templateUrl : 'partials/tareas/templates/tareas.html',
        })
        .state('app.recursos',{
            url : '/recursos',
            templateUrl : 'partials/recursos/templates/recursos.html',
        })
    });
});

```

## Chat.html

```

<div class="col-xs-12">
    <div class="box box-solid">
        <div class="box-body">
            <div class="row">
                <div class="col-md-3 col-sm-4">
                    <div class="box-header">
                        <i class="fa fa-inbox"></i>
                        <h3 class="box-title">Chat</h3>
                    </div>
                    <div style="margin-top: 15px;">
                        <ul class="nav nav-pills nav-stacked" style="overflow:auto;height:300px;">
                            <li class="header">Equipo</li>
                            <li><a href="#"><i class="fa fa-group zz"></i>General</a></li>
                            <li>
                                <a href="#">
                                    <i class="fa fa-circle text-sucess"></i>Susano Jose
                                </a>
                            </li>
                            <li>
                                <a href="#">
                                    <i class="fa fa-circle text-sucess"></i>Andres Pimienta
                                </a>
                            </li>
                        </ul>
                    </div>
                <div class="col-md-9 col-sm-8 ng-scope" ui-view="">
                    <div class="callout callout-info">
                        <h4>Bienvenido al Chat del Equipo</h4>
                        <p>Selecciona a la Izquierda con quien Chatear.</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

```

## Dashboard.html

```

<div class="row">
    <div class="col-md-12">
        <ul class="timeline">
            <li class="time-label">
                <span class="bg-red">10 Feb 2015</span>
            </li>

            <li>
                <i class="fa fa-check bg-blue"></i>
                <div class="timeline-item">
                    <h3 class="timeline-header"><a href="javascript:void(0);">Fulano de Tal</a> Termino una Tarea</h3>
                    <div class="timeline-body">
                        Diseñar la maqueta del sitio web
                    </div>
                </div>
            </li>
            <li>
                <i class="fa fa-envelope bg-blue"></i>
                <div class="timeline-item">
                    <h3 class="timeline-header"><a href="javascript:void(0);">Susano Jose</a> Compartio un Recurso</h3>
                    <div class="timeline-body">
                        Imagenes del prototipo de maquetacion
                    </div>
                </div>
            </li>
        </ul>
    </div>

```

## Index.html(/templates)

```
<header class="header">
  <a href="#" class="logo">TeamApp</a>

  <ng-include src="'partials/includes/navbar.html'" ></ng-include>
</header>

<div class="wrapper row-offcanvas row-offcanvas-left">
  <aside class="left-side sidebar-offcanvas" >
    <ng-include src="'partials/includes/sidebar.html'" ></ng-include>
  </aside>
</div>

<aside class="right-side">

  <section class="content-header">
    <h1>{{modulo}}<small>Modulo</small>
  </h1>
  </section>

  <section class="content" ui-view>

  </section>
</aside>
```



## Recursos.html

```
<div class="mailbox row">
  <div class="col-xs-12">
    <div class="col box-solid">
      <div class="box-body">
        <div class="row">
          <div class="col-md-3 col-sm-4">
            <div class="box-header">
              <i class="fa fa-inbox"></i>
              <h3 class="box-title">Inbox</h3>
            </div>
            <a class="btn btn-block btn-primary"><i class="fa fa-pencil"></i>Enviar Recursos</a>
            <div style="margin-top: 15px;">
              <ul class="nav nav-pills nav-stacked">
                <li class="header">Folders</li>
                <li ui-sref-active="active"><a href="#"><i class="fa fa-inbox"></i>Recibidos</a></li>
                <li ui-sref-active="active"><a href="#"><i class="fa fa-pencil-square-o"></i>Enviados</a></li>
              </ul>
            </div>
          </div>
          <div class="col-md-9 col-sm-8" ui-view>
            <div class="callout callout-info">
              <h4>Modulo de Recursos!</h4>
              <p>Subir y Recibir Archivos sin limite de Peso.</p>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

## Tareas.html

```
<div class="row">
  <div class="col-md-12">
    <div class="box box-primary">
      <div class="box-header">
        <i class="ion ion-clipboard"></i>
        <h3 class="box-title">To Do List</h3>
      </div>
      <div class="box-body">
        <form>
          <div class="form-group">
            <div class="col-xs-3">
              <input name="descripcion" type="text" class="form-control" placeholder="Descripcion">
            </div>
            <div class="col-xs-3">
              <button type="submit" class="btn btn-primary pull-left"><i class="fa fa-check"></i></button>
            </div>
          </div>
        </form>
        <br>
        <br>
        <ul class="todo-list" style="overflow:auto;height:200px;">
          <li>
            <input type="checkbox">
            <span class="text">Tarea 1</span>
          </li>
        </ul>
      </div>
      <div class="box-footer clearfix no-border">
        <button class="btn btn-primary pull-right">Guardar</button>
      </div>
    </div>
```

```

        </div>
    </div>
</div>
<div class="box">
    <div class="box-header">
        <h3 class="box-title">Historial de Tareas Finalizadas</h3>
    </div>
    <div class="box-body no-padding">
        <table class="table table-condensed">
            <tbody>
                <tr>
                    <th>Descripcion</th>
                    <th>Fecha</th>
                    <th style="width: 40px">Estado</th>
                </tr>
                <tr>
                    <td>Tarea 2</td>
                    <td>28/12/1992</td>
                    <td><span class="badge bg-green">Finalizada</span></td>
                </tr>
                <tr>
                    <td>Tarea 3</td>
                    <td>29/12/1992</td>
                    <td><span class="badge bg-green">Finalizada</span></td>
                </tr>
            </tbody>
        </table>
    </div>
</div>

```

## Modelos Mongo

Instalar mongoose : npm install mongoose --save

Podemos crear y utilizar modelos con nuestra base de datos MongoDB.

**Mongoose** es una librería de NodeJS para la implementación de modelos de nuestra base de datos.

Modelos para ejercicio anterior:

Models.js

```

var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/teamapp');

var db = mongoose.connection;
db.on('error', console.error.bind(console, 'Error de Conexion'));
db.once('open', function callback(){
    console.log('Base de Datos Teamapp abierta');
});

module.exports = mongoose;

```

Chat.js

```

var models = require('./models');
Schema = models.Schema;

var chatSchema = new Schema({
    remitente : {type : Schema.Types.ObjectId, ref : 'Usuario'},
    destinatario : {type : Schema.Types.ObjectId, ref : 'Usuario'},
    tipo : String,
    mensajes : [
        {remitente : {type : Schema.Types.ObjectId, ref : 'Usuario'},
        destinatario : {type : Schema.Types.ObjectId, ref : 'Usuario'},
        contenido : String}
    ]
});

var Chat = models.model('Chat', chatSchema, 'chats');
module.exports = Chat;

```

Recursos.js

```

var models = require('./models'),
Schema = models.Schema;

var recursosSchema = new Schema({
    archivos : [{type: String}],
    remitente : {type : Schema.Types.ObjectId, ref : 'Usuario'},
    destinatarios : [{type: String}],
    fecha : {type : Date, default : Date()},
    asunto : String
});

var Recursos = models.model('Recurso', recursosSchema, 'recursos');
module.exports = Recursos;

```

### Tareas.js

```

var models = require('./models'),
Schema = models.Schema;

var tareasSchema = new Schema({
    descripcion : String,
    usuario : {type : Schema.Types.ObjectId, ref : 'Usuario'},
    finalizada : {
        estado : {type : Boolean, default : false},
        fecha : Date
    }
});

var Tareas = models.model('Tarea', tareasSchema, 'tareas');
module.exports = Tareas;

```

### Timeline.js

```

var models = require('./models'),
Schema = models.Schema;

var timelineSchema = new Schema({
    usuario : {type : Schema.Types.ObjectId, ref : 'Usuario'},
    tarea : {type : Schema.Types.ObjectId, ref : 'Tarea'},
    recurso : {type : Schema.Types.ObjectId, ref : 'Recurso'},
    tipo : String,
    fecha : {type : Date, default : Date()},
    accion : String,
    descripcion : String
});

var Timeline = models.model('Timeline', timelineSchema);
module.exports = Timeline;

```

### Usuarios.js

```

var models = require('./models'),
Schema = models.Schema;

var usuariosSchema = new Schema({
    nombre : String,
    nombre_usuario : String,
    password : String,
    twitter : String
});

var Usuario = models.model('Usuario', usuariosSchema, 'usuarios');
module.exports = Usuario;

```

## Passport.js I

### Registro de usuarios e inicio de sesión con PassportJS

#### Express.js

```

var express = require('express');
var logger = require('morgan');
var bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');
var swig = require('swig');
var express = require('express');
var passport = require('./passport');
var session = require('express-session');
var redisStore = require('connect-redis')(session);

module.exports = function(app, config){
    app.engine('html', swig.renderFile);
    app.set('view engine', 'html');
    app.set('views', config.rootPath + '/server/views');
    app.set('view cache', false);
    swig.setDefaults({cache: false, varControls: ['{', '}']});

    app.use(cookieParser());
    app.use(logger('dev'));
    app.use(bodyParser());

    app.use(session({store : new redisStore({}), secret : 'teamapp next'}));
    app.use(passport.initialize());
    app.use(passport.session());
    app.use(express.static(config.rootPath + '/public'));
}

```

#### packa.json

```

{
    "name": "Modulacion",
    "version": "1.0.0",
    "description": "",
    "main": "Servidor.js",
    "scripts": {
        "test": "null"
    },
    "author": "",
    "license": "ISC",
    "dependencies": {
        "body-parser": "^1.12.3",
        "connect-redis": "2.3.0",
        "cookie-parser": "1.3.4",
        "express": "4.12.3",
        "express-session": "1.11.1",
        "mongoose": "4.0.2",
        "morgan": "1.5.2",
        "passport": "0.2.1",
        "swig": "1.4.2"
    }
}

```

## Routes.js

```
var usuarios = require('../controllers/usuarios');
module.exports = function(app){
    app.get('/partials/*', function(req, res){
        res.render('....../public/app/' + req.params['0']);
    });

    app.post('/registro', usuarios.registro);

    app.get('*', function(req, res){
        res.render('index');
    });
};
```

## Usuarios.js

```
var Usuario = require('../models/usuarios');
var passport = require('../config/passport');

exports.registro = function(req, res, next){
    var usuario = new Usuario(req.body);
    usuario.save(function(err, usuario){
        if(err){
            res.send({sucess: false, message: err});
        }else{
            req.logIn(usuario, function(err){
                if(!err){
                    res.send({logged:true, sucess: true, usuario : req.session.passport});
                }else{
                    console.log(err);
                    res.send({logged:false, sucess:true, usuario : usuario});
                }
            });
        }
    });
};
```

## Passport.js

```
var passport = require('passport');
var Usuario = require('../models/usuarios');

passport.serializeUser(function(user, done){
    if(user){
        done(null, user);
    }
});

passport.deserializeUser(function(user, done){
    Usuario.findOne({_id:user._id})
    .exec(function(err, user){
        if(user){
            return done(null, user);
        }else{
            return done(null, false);
        }
    });
});
module.exports = passport;
```

## Registro.html

```
<div class="form-box" id="login-box">
    <div class="header">Registro de Usuarios</div>
    <form ng-submit="register()">
        <div class="body bg-gray">
            <div class="form-group">
                <input type="text" name="nombre" ng-model="usuario.nombre" class="form-control" placeholder="Nombre Completo" />
            </div>
            <div class="form-group">
                <input type="text" name="nombre_usuario" ng-model="usuario.nombre_usuario" class="form-control" placeholder="Nombre de Usuario" />
            </div>
            <div class="form-group">
                <input type="password" name="password" ng-model="usuario.password" class="form-control" placeholder="Password" />
            </div>
        </div>
        <div class="footer">
            <button type="submit" ng-disabled="enviando" class="btn bg-olive btn-block">Registrar</button>
        </div>
    </form>
</div>
```

## RegistroCtrl.js

```
angular.module('Teamapp').controller('registroCtrl', function($scope, $http, $state){  
    $scope.usuario = {};  
    $scope.register = function(){  
        $scope.enviado = true;  
        $http.post('/registro', $scope.usuario)  
        .then(function(response){  
            var data = response.data;  
            if(data.sucess){  
                if(data.logged){  
                    $state.transitionTo('app.dashboard');  
                }else{  
                    $state.go('login');  
                }  
            }else{  
                console.log("Error al Registrarse");  
                $scope.enviado = false;  
            }  
        });  
    }  
});
```

## Scripts.js

```
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>  
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>  
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>  
<script type="text/javascript" src="/vendor/angular/angular.js"></script>  
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>  
<script type="text/javascript" src="/app/app.js"></script>  
<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>  
<script type="text/javascript" src="/app/sign/controllers/registroCtrl.js"></script>
```

## App.js

```
app.config(['$stateProvider','$urlRouterProvider', function($stateProvider, $urlRouterProvider){  
    $urlRouterProvider.otherwise('/app/dashboard');  
  
    $stateProvider  
        .state('app',{  
            url : '/app',  
            templateUrl : 'partials/index/templates/index.html',  
            controller : 'indexCtrl'  
        })  
        .state('app.dashboard',{  
            url : '/dashboard',  
            templateUrl : 'partials/dashboard/templates/dashboard.html',  
        })  
        .state('app.chat',{  
            url : '/chat',  
            templateUrl : 'partials/chat/templates/chat.html',  
        })  
        .state('app.tareas',{  
            url : '/tareas',  
            templateUrl : 'partials/tareas/templates/tareas.html',  
        })  
        .state('app.recursos',{  
            url : '/recursos',  
            templateUrl : 'partials/recursos/templates/recursos.html',  
        })  
        .state('registro',{  
            url : '/registro',  
            templateUrl : 'partials/sign/templates/registro.html',  
            controller : 'registroCtrl'  
        })  
});
```

## Autenticación y gestión de sesiones con PassportJS

## Usuarios.js

```
var Usuario = require('../models/usuarios');
var passport = require('../config/passport');

exports.registro = function(req, res, next){
    var usuario = new Usuario(req.body);
    usuario.save(function(err, usuario){
        if(err){
            res.send({sucess: false, message: err});
        }else{
            req.logIn(usuario, function(err){
                if(!err){
                    res.send({logged:true, sucess: true, usuario : req.session.passport});
                }else{
                    console.log(err);
                    res.send({logged:false, sucess:true, usuario : usuario});
                }
            });
        }
    });
};

exports.login = function(req, res, next){
    var auth = passport.authenticate('local', function(err, user){
        if(err){
            console.log(err);
            return next(err);
        }
        if(!err){
            console.log("No hay usuario");
            res.send({sucess:false});
        }
        req.logIn(user, function(err){
            if(err){
                console.log("Error al loguearse");
                return next(err);
            }
            res.send({sucess: true, user: user});
        });
    });
    auth(req, res, next);
};

exports.userAuthenticated = function(req, res, next){
    if(req.isAuthenticated()){
        res.send({user:session.passport, isLoggedIn:true});
    }else{
        res.send({user:{}, isLoggedIn:false});
    }
};

exports.logout = function(req, res, next){
    req.session.destroy(function(err){
        console.log("Logout");
        if(!err){
            res.send({destroy:true});
        }else{
            console.log(err);
        }
    });
};
```

## Express.js

```
var logger = require('morgan');
var bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');
var swig = require('swig');
var express = require('express');
var passport = require('../passport');
var session = require('express-session');
var redisStore = require('connect-redis')(session);

module.exports = function(app, config){
    app.engine('html', swig.renderFile);
    app.set('view engine', 'html');
    app.set('views', config.rootPath + '/server/views');

    app.set('view cache', false);
    swig.setDefaults({cache: false, varControls: ['{\w+}', '\w+']});

    app.use(cookieParser());
    app.use(logger('dev'));
    app.use(bodyParser());

    app.use(session({store : new redisStore({
        disableTTL:true
    }), secret : 'teamapp next'}));
    app.use(passport.initialize());
    app.use(passport.session());
    app.use(express.static(config.rootPath+'/public'));
};
```

## Passport.js

```
var Usuario = require('../models/usuarios');

passport.serializeUser(function(user, done){
  if(user){
    done(null, user);
  }
});

passport.deserializeUser(function(user, done){
  Usuario.findOne({_id:user._id})
    .exec(function(err, user){
      if(user){
        return done(null, user);
      }else{
        return done(null, false);
      }
    });
});

passport.use('local', new LocalStrategy(
  function(username, password, done){
    Usuario.findOne({nombre_usuario : username})
      .exec(function(err, user){
        if(user && user.authenticate(password)){
          return done(null, user)
        }else{
          return done(null, false)
        }
      })
  }));
module.exports = passport;
```

## routes.js

```
var usuarios = require('../controllers/usuarios');
module.exports = function(app){
  app.get('/partials/*', function(req, res){
    res.render('....../public/app/' + req.params['0']);
  });

  app.post('/registro', usuarios.registro);

  app.post('/login', usuarios.login);

  app.post('/logout', usuarios.logout);

  app.post('/session', usuarios.userAuthenticated);

  app.get('*', function(req, res){
    res.render('index');
  });
};
```

## Script.js

```
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
<script type="text/javascript" src="/vendor/angular-animate/angular-animate.js"></script>
<script type="text/javascript" src="/vendor/angular-toastr/dist/angular-toastr.js"></script>
<script type="text/javascript" src="/vendor/lodash/lodash.js"></script>

<script type="text/javascript" src="/app/app.js"></script>

<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/registroCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/loginCtrl.js"></script>
<script type="text/javascript" src="/app/dashboard/controllers/dashboardCtrl.js"></script>

<script type="text/javascript" src="/app/services/toastr.js"></script>
<script type="text/javascript" src="/app/services/session.js"></script>
```

## Links.js

```
<meta charset="utf-8">
<title>Team App | {% block title %} {% endblock %}</title>
<link rel="stylesheet" type="text/css" href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.2.0/css/font-awesome.min.css">
<link rel="stylesheet" type="text/css" href="https://code.ionicframework.com/ionicons/1.5.2/css/ionicons.min.css">
<link rel="stylesheet" type="text/css" href="/vendor/angular-toastr/dist/angular-toastr.css">
<link rel="stylesheet" type="text/css" href="/css/AdminLTE.css">
```

Ahora se modifican:

sidebar.html  
navbar.html

login.html

```

<div class="form-box" id="login-box">
  <div class="header">Inicio de Sesión</div>
  <form name="form" ng-submit="signin()">
    <div class="body bg-gray">
      <div class="form-group">
        <input type="text" name="username" ng-model="usuario.username" class="form-control" placeholder="Nombre de Usuario" />
      </div>
      <div class="form-group">
        <input type="password" name="password" ng-model="usuario.password" class="form-control" placeholder="Password" />
      </div>
    </div>
    <div class="footer">
      <button type="submit" class="btn bg-olive btn-block">Entrar</button>
      <p>Si eres nuevo, por favor <a ui-sref="registro" class="text-center">register</a></p>
    </div>
  </form>
  <div class="margin text-center">
    <span>Social Login</span>
    <br>
    <button class="btn bg-aqua btn-circle"><i class="fa fa-twitter"></i></button>
  </div>
</div>

```

### loginCtrl.js

```

angular.module('Teamapp').controller('loginCtrl', function($scope, $http, $state, toastF, Session){
  $scope.master = {};
  $scope.signin = function(){
    var usuario = {username : $scope.usuario.username, password : $scope.usuario.password};
    Session.logIn(usuario)
    .then(function(response){
      if(response.data.sucess){
        toastF.sucess('Iniciaste sesion Correctamente');
        $state.transitionTo('app.dashboard');
      }else{
        toastF.error('Error de autenticacion, verifica tus datos');
        $scope.usuario = angular.copy($scope.master);
        $scope.form.$setPristine();
      }
    });
  }

  Session.isLoggedIn()
  .then(function(response){
    var isLoggedIn = response.data.isLoggedIn;
    if(isLoggedIn){
      $state.go('app.dashboard');
    }
  });
});

```

### dashboardCtrl.js

```

angular.module('Teamapp').controller('dashboardCtrl', function($scope, Session){
});

```

### indexCtrl.js

```

var app = angular.module('Teamapp');

app.controller('indexCtrl', function($rootScope, $state, $scope, Session){
  function Modulo(state){
    this.state = state.name;
    this.name = state.name.split('.')[1];
    this.getName = function(){
      return this.name && this.name[0].toUpperCase() + this.name.slice(1);
    };
  }

  $scope.modulo = new Modulo($state.current).getName();
  $scope.logout = function(){
    Session.logOut()
    .then(function(response){
      if(response.data.destroy){
        $state.go('login');
      }
    });
  }

  Session.getUsuario()
  .then(function(response){
    $scope.usuario = response.data.user.user;
  });
});

```

```

Session.isLoggedIn()
.then(function(response){
  var isLoggedIn = response.data.isLoggedIn;
  if(!isLoggedIn){
    $state.go('login');
  }
});

$rootScope.$on("$stateChangeStart", function(event, toState, toParams, fromState, fromParams){
  $scope.modulo = new Modulo(toState).getName();
});
});

Toastr.js
angular.module('Teamapp').service('toastF', function(toastr){
  this.success = function(msg){
    toastr.success(msg);
  },
  this.error = function(msg){
    toastr.error(msg);
  }
});

```

### Session.js

```

angular.module('Teamapp').factory('Session', function($http, $state, $rootScope){
  function Usuario(){
    this.logIn = function(usuario){
      return $http.post('/login', usuario);
    }

    this.getUsuario = function(){
      return $http.get('/session');
    }

    this.logOut = function(){
      return $http.post('/logout');
    }

    this.isLoggedIn = function(usuario){
      return $http.get('/session');
    }
  }

  var Usuario = new Usuario();
  var Session = {
    logIn : Usuario.logIn,
    getUsuario : Usuario.getUsuario,
    logOut : Usuario.logOut,
    isLoggedIn : Usuario.isLoggedIn
  }
}

Usuario.getUsuario();
Usuario.isLoggedIn();
return Session;
});

```

Se modifica app.js y se crea .bower.c

```
{
  "name": "Modulacion",
  "version": "1.0.0",
  "description": "",
  "main": "Servidor.js",
  "scripts": {
    "test": "null"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.12.3",
    "connect-redis": "^2.3.0",
    "cookie-parser": "^1.3.4",
    "express": "4.12.3",
    "express-session": "1.11.1",
    "mongoose": "4.0.2",
    "morgan": "1.5.2",
    "passport": "0.2.1",
    "passport-local": "1.0.0",
    "swig": "1.4.2"
  }
}

{
  "name": "Modulacion",
  "version": "0.0.0",
  "authors": [
    "NextU"
  ],
  "license": "MIT",
  "dependencies": {
    "angular-toastr": "0.5.2",
    "angular-animate": "1.3.15",
    "lodash": "3.7.0",
    "async": "0.9.2"
  }
}
```

Nombre de Usuario

Password

Entrar

Si eres nuevo, por favor registrate

Social Login

## Inicio de sesión usando redes sociales con PassportJS

### Usuarios.js

```
var models = require('./models'),
Schema = models.Schema;

var usuariosSchema = new Schema({
  nombre : String,
  nombre_usuario : String,
  password : String,
  twitter : Schema.Types.Mixed
});

usuariosSchema.methods = {
  authenticate : function(password){
    return this.password == password;
  }
}

var Usuario = models.model('Usuario', usuariosSchema, 'usuarios');
module.exports = Usuario;
```

### package.json

```
{
  "name": "Modulacion",
  "version": "1.0.0",
  "description": "",
  "main": "Servidor.js",
  "scripts": {
    "test": "null"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "1.12.3",
    "connect-redis": "2.3.0",
    "cookie-parser": "1.3.4",
    "express": "4.12.3",
    "express-session": "1.11.1",
    "mongoose": "4.0.2",
    "morgan": "1.5.2",
    "passport": "0.2.1",
    "passport-local": "1.0.0",
    "passport-twitter": "1.0.3",
    "swig": "1.4.2"
  }
}
```

### Passport.js

```

var passport = require('passport'),
LocalStrategy = require('passport-local').Strategy,
TwitterStrategy = require('passport-twitter').Strategy;
var Usuario = require('../models/usuarios');

passport.serializeUser(function(user, done){
  if(user){
    done(null, user);
  }
});

passport.deserializeUser(function(user, done){
  Usuario.findOne({_id:user._id})
  .exec(function(err, user){
    if(user){
      return done(null, user);
    }else{
      return done(null, false);
    }
  });
});

passport.use('local', new LocalStrategy(
  function(username, password, done){
    Usuario.findOne({nombre_usuario : username})
    .exec(function(err, user){
      if(user && user.authenticate(password)){
        return done(null, user)
      }else{
        return done(null, false)
      }
    })
  })
);

passport.use(new TwitterStrategy({
  consumerKey: '8N5MQYF9pcTugBxQSLNAHdIx8',
  consumerSecret: 'KFDEVIZoS7CwzKiDWmz5F3ViJ8cQZ9cNG5vioXKw8HpRyQheJC',
  callbackURL: 'http://127.0.0.1:3000/auth/twitter/callback'
}),
function(token, tokenSecret, profile, done){
  Usuario.findOne({nombre_usuario:profile.username})
  .exec(function(err, usuario){
    if(err){
      console.log(err);
      return done(err);
    }
    if(usuario){
      usuario.twitter = profile;
      usuario.save(function(err, user){
        if(err){
          return done(err);
        }
        done(null, user);
      });
    }else{
      new Usuario({
        nombre_usuario : profile.username,
        nombre : profile.displayName,
        twitter : profile
      }).save(function(err, usuario){
        if(!err){
          return done(null, usuario);
        }
      })
    }
  })
});

```

Routes.js

```

var usuarios = require('../controllers/usuarios');
var passport = require('./passport');

module.exports = function(app){
    app.get('/partials/*', function(req, res){
        res.render('../public/app/' + req.params['0']);
    });

    app.post('/registro', usuarios.registro);

    app.post('/login', usuarios.login);

    app.post('/logout', usuarios.logout);

    app.get('/session', usuarios.userAuthenticated);

    app.get('auth/twitter', passport.authenticate('twitter'));

    app.get('/auth/twitter/callback',
        passport.authenticate('twitter',{
            successRedirect: '/',
            failureRedirect: '/login'
        }));
}

app.get('*', function(req, res){
    res.render('index');
});

```

Login.html

```

<div class="form-box" id="login-box">
    <div class="header">Inicio de Sesión</div>
    <form name="form" ng-submit="signin()">
        <div class="body bg-gray">
            <div class="form-group">
                <input type="text" name="username" ng-model="usuario.username" class="form-control" placeholder="Nombre de Usuario" />
            </div>
            <div class="form-group">
                <input type="password" name="password" ng-model="usuario.password" class="form-control" placeholder="Password" />
            </div>
            <div class="form-group">
                <button type="submit" class="btn bg-olive btn-block">Entrar</button>
                <p>Si eres nuevo, por favor<a ui-sref="register" class="text-center">regístrate</a></p>
            </div>
        </form>
        <div class="margin text-center">
            <span>Social Login</span>
            <br>
            <a href="/auth/twitter" target="_self" class="btn bg-aqua btn-circle"><i class="fa fa-twitter"></i></a>
        </div>
    </div>

```

Se modifica: app.js, navbar.html

## Tareas I

### Módulos para el sistema de tareas

Ejercicio

## Tareas.js

```
var Tareas = require('../models/tareas');
var ObjectId = require('mongoose').Types.ObjectId;
var _ = require('lodash');

exports.guardar = function(req, res, next){
    var tareas = new Tareas({
        descripcion: req.body.descripcion,
        usuario: req.session.passport.user._id.toString()
    });

    tareas.save(function(err, tarea){
        if(err){
            console.log("err:"+err);
            res.send({success:false, message:err});
        }else{
            console.log("Guardado con Exito");
            res.send({success: true, tarea: tarea});
        }
    });
};

exports.getTareas = function(req, res, next){
    Tareas.find({usuario: req.session.passport.user._id.toString()})
    .exec(function(err, tareas){
        if(err){
            console.log(err);
        }else{
            res.send(tareas);
        }
    });
};

exports.guardarFinalizadas = function(req, res, next){
    var ids = req.body.ids;

    Tareas.find({_id: {$in:ids}});
    .exec(function(err, tareas){
        if(err){
            console.log(err);
        }else{
            _.each(tareas, function(tarea){
                tarea.finalizada.estado = true;
                tarea.finalizada.fecha = new Date();
                tarea.save();
            });
            res.send(tareas);
        }
    });
};
};
```

## Routes.js

```
var usuarios = require('../controllers/usuarios');
var tareas = require('../controllers/tareas');
var passport = require('../passport');

module.exports = function(app){
    app.get('/partials/*', function(req, res){
        res.render('../public/app/' + req.params['0']);
    });

    app.post('/registro', usuarios.registro);

    app.post('/login', usuarios.login);

    app.post('/logout', usuarios.logout);

    app.get('/session', usuarios.userAuthenticated);

    app.get('auth/twitter', passport.authenticate('twitter'));

    app.get('/auth/twitter/callback',
        passport.authenticate('twitter',{
            successRedirect: '/',
            failureRedirect: '/login'
        }));
    app.post('/tareas', tareas.guardar);

    app.get('/tareas', tareas.getTareas);

    app.post('/tareas/finalizadas', tareas.guardarFinalizadas);
```

### tareasCtrl.js

```
angular.module('Teamapp').controller('tareasCtrl', function($scope, TareasService){
  var checkeados = [];

  $scope.tareas = [];
  $scope.item_master = {descripcion : "", fecha : "", finalizada : false};
  $scope.date = new Date();

  $scope.guardar = function(){
    TareasService.guardarTareas({descripcion: $scope.item.descripcion})
    .then(function(response){
      if(response.data.success){
        $scope.tareas.push(response.data.tarea);
        angular.copy($scope.item_master, $scope.item);
      }
    });
  }

  $scope.enviar_finalizadas = function(){
    var ids = _.pluck(checkeados, '_id');
    TareasService.guardarFinalizadas({ids : ids})
    .then(function(response){
      _.each(response.data, function(item){
        var item = item;
        _.remove($scope.tareas, function(tarea){
          return tarea._id == item._id;
        });
      });
    });
  }
}

TareasService.getTareas()
.then(function(response){
  _.each(response.data, function(item){
    if(item.finalizada.estado == false){
      $scope.tareas.push(item);
    }
  });
});

$scope.$watchCollection("tareas | filter : {finalizada : {estado : true}}", function(newv, old){
  checkeados = newv;
});
});
```

### Tareas.js(servicios)

```
angular.module('Teamapp').factory('TareasService', function($http){
  return {
    getTareas : function(){
      return $http.get('/tareas');
    },
    getTareasFinalizadas : function(){
      return $http.get('/tareas/finalizadas');
    },
    guardarTareas : function(datos){
      return $http.post('/tareas', datos);
    },
    guardarFinalizadas : function(ids){
      return $http.post('/tareas/finalizadas', ids);
    }
  }
});
```

### Tareas.html

```

<div class="row">
  <div class="col-md-12">
    <div class="box box-primary">
      <div class="box-header">
        <i class="ion ion-clipboard"></i>
        <h3 class="box-title">To Do List</h3>
      </div>
      <div class="box-body">
        <form>
          <div class="form-group">
            <div class="col-xs-3">
              <input name="descripcion" ng-model="item.descripcion" type="text" class="form-control" placeholder="Descripcion">
            </div>
            <div class="col-xs-3">
              <button type="submit" class="btn btn-primary pull-left"><i class="fa fa-check"></i>Crear</button>
            </div>
          </div>
        </form>
        <br>
        <br>
        <br>
        <ul class="todo-list" style="overflow:auto;height:200px;">
          <li ng-repeat="tarea in tareas">
            <input type="checkbox" ng-model="tarea.finalizada.estado" value="{{tarea.finalizada.estado}}"/>
            <span class="text">{{tarea.descripcion}}</span>
          </li>
        </ul>
      </div>
      <div class="box-footer clearfix no-border">

```

## Scripts.html

```

<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
<script type="text/javascript" src="/vendor/angular-animate/angular-animate.js"></script>
<script type="text/javascript" src="/vendor/angular-toastr/dist/angular-toastr.js"></script>
<script type="text/javascript" src="/vendor/lodash/lodash.js"></script>
<script type="text/javascript" src="/vendor/async/lib/async.js"></script>

<script type="text/javascript" src="/app/app.js"></script>

<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/registroCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/loginCtrl.js"></script>
<script type="text/javascript" src="/app/dashboard/controllers/dashboardCtrl.js"></script>
<script type="text/javascript" src="/app/tareas/controllers/tareasCtrl.js"></script>

<script type="text/javascript" src="/app/services/toastr.js"></script>
<script type="text/javascript" src="/app/services/session.js"></script>
<script type="text/javascript" src="/app/services/tareas.js"></script>

```

## Se modifica: app.js

```

{
  "name": "Modulacion",
  "version": "1.0.0",
  "description": "",
  "main": "Servidor.js",
  "scripts": {
    "test": "null"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.12.3",
    "connect-redis": "^2.3.0",
    "cookie-parser": "^1.3.4",
    "express": "^4.12.3",
    "express-session": "^1.11.1",
    "lodash": "3.8.0",
    "mongoose": "4.0.2",
    "morgan": "1.5.2",
    "passport": "0.2.1",
    "passport-local": "1.0.0",
    "passport-twitter": "1.0.3",
    "swig": "1.4.2"
  }
}

```

## Tareas y la base de datos

## Tareas.html(agregar al final)

```
<div class="box">
  <div class="box-header">
    <h3 class="box-title">Historial de Tareas Finalizadas</h3>
  </div>
  <div class="box-body no-padding">
    <div class="callout callout-danger" ng-show="!tareas_finalizadas.length">
      <h4>Aun no has finalizado Tareas!</h4>
      <p>Aqui mostraremos tu historial de tareas finalizadas.</p>
    </div>
    <table class="table table-condensed" ng-hide="!tareas_finalizadas.length">
      <tbody>
        <tr>
          <th>Descripcion</th>
          <th>Fecha</th>
          <th style="width: 40px">Estado</th>
        </tr>
        <tr ng-repeat="finalizada in tareas_finalizadas">
          <td>{{finalizada.descripcion}}</td>
          <td>{{finalizada.finalizada.fecha | date: 'yyyy-MM-dd'}}</td>
          <td><span class="badge bg-green">Finalizada</span></td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

## Se modifica tareasCtrl.js(agregar al final)

```
$scope.ordenarListas = function(response){
  var finalizadas = [];
  var no_finalizadas = [];
  _.each(response, function(item){
    if(item.finalizada.estado){
      finalizadas.push(item);
    }else{
      no_finalizadas.push(item);
    }
  });
  angular.copy(no_finalizadas, $scope.tareas);
  angular.copy(finalizadas, $scope.tareas_finalizadas);
}

TareasService.getTareas()
.then(function(response){
  $scope.ordenarListas(response.data);
});

$scope.$watchCollection("tareas | filter : {finalizada : {estado : true}}", function(newv, oldv){
  checkeados = newv;
});
});
```

## Recursos I

### Subir archivos a la base de datos

## Crear.html

```
<div class="box box-primary">
  <div class="box-header">
    <h3 class="box-title">Enviar Archivos</h3>
  </div>

  <form ng-submit="uploadFile()" enctype="multipart/form-data">
    <div class="box-body">
      <div class="form-group">
        <label for="destinatarios">Usuarios</label>
        <input type="text" class="form-control" id="destinatarios" ng-model="destinatarios" placeholder="Destinatarios">
      </div>
      <div class="form-group">
        <label for="asunto">Asunto</label>
        <input type="text" class="form-control" id="asunto" ng-model="asunto" placeholder="Asunto">
      </div>
      <div class="form-group">
        <label for="archivos">Enviar Archivos</label>
        <input type="file" file-input="files" multiple />
      </div>
      <div class="form-group">
        <ul ng-repeat="file in files">{{file.name}}</ul>
      </div>
    </div>
    <div class="box-footer">
      <button type="submit" class="btn btn-primary">Enviar</button>
    </div>
  </form>
</div>
```

## recursosCtrl.js

```
angular.module('Teamapp').controller('recursosCtrl', function($scope, $http, $state, ToastService, RecursosService){
  $scope.filesChanged = function(elm){
    $scope.files = elm.files;
    $scope.$apply();
  }

  $scope.uploadFile = function(){
    var fd = new FormData();
    angular.forEach($scope.files, function(file){
      fd.append('file', file);
    });
    fd.append('destinatarios', $scope.destinatarios);
    fd.append('asunto', $scope.asunto);

    $http.post('/recurso', fd, {
      transformRequest: angular.identity,
      headers : {'Content-Type' : undefined}
    })
    .success(function(d){
      console.log(d);
      ToastService.success('Enviado Correctamente');
      $state.transitionTo('app.recursos');
    });
  };
});
```

## Toastr.js

```
angular.module('Teamapp').service('ToastService', function(toastr){
  this.success = function(msg){
    toastr.success(msg);
  },
  this.error = function(msg){
    toastr.error(msg);
  }
});
```

## Recursos.js(servicios)

```
angular.module('Teamapp').factory('RecursosService', function($http){
  return{
    getRecursosRecibidos : function(){
      return $http.get('/recursos/recibidos');
    },
    getRecursosEnviados : function(){
      return $http.get('/recursos/enviados');
    },
    getDetalle : function(recurso){
      return $http.get('/recurso/'+recurso);
    }
  }
});
```

## loginCtrl.js

```
angular.module('Teamapp').controller('loginCtrl', function($scope, $http, $state, ToastService, Session){  
    $scope.master = {};  
    $scope.signin = function(){  
        var usuario = {username : $scope.usuario.username, password : $scope.usuario.password};  
        Session.logIn(usuario)  
        .then(function(response){  
            if(response.data.success){  
                ToastService.success('Iniciaste sesion Correctamente');  
                $state.transitionTo('app.dashboard');  
            }else{  
                ToastService.error('Error de autenticacion, verifica tus datos');  
                $scope.usuario = angular.copy($scope.master);  
                $scope.form.$setPristine();  
            }  
        });  
    };  
  
    Session.isLoggedIn()  
    .then(function(response){  
        var isLoggedIn = response.data.isLoggedIn;  
        if(isLoggedIn){  
            $state.go('app.dashboard');  
        }  
    });  
});
```

## routes.js

```
var usuarios = require('../controllers/usuarios');  
var tareas = require('../controllers/tareas');  
var recursos = require('../controllers/recursos');  
var passport = require('../passport');  
var multiparty = require('connect-multiparty')();  
  
module.exports = function(app){  
    app.get('/partials/*', function(req, res){  
        res.render('../public/app/' + req.params['0']);  
    });  
  
    app.post('/registro', usuarios.registro);  
  
    app.post('/login', usuarios.login);  
  
    app.post('/logout', usuarios.logout);  
  
    app.get('/session', usuarios.userAuthenticated);  
  
    app.get('auth/twitter', passport.authenticate('twitter'));  
  
    app.get('/auth/twitter/callback',  
        passport.authenticate('twitter',{  
            successRedirect: '/',  
            failureRedirect: '/login'  
        }));  
  
    app.post('/tareas', tareas.guardar);  
  
    app.get('/tareas', tareas.getTareas);  
  
    app.post('/tareas/finalizadas', tareas.guardarFinalizadas);  
  
    app.post('/recurso', multiparty, recursos.guardar_recurso);  
  
    app.get('*', function(req, res){  
        res.render('index');  
    });  
};
```

```
{  
    "name": "Modulacion",  
    "version": "1.0.0",  
    "description": "",  
    "main": "Servidor.js",  
    "scripts": {  
        "test": "null"  
    },  
    "author": "",  
    "license": "ISC",  
    "dependencies": {  
        "async": "^0.9.0",  
        "body-parser": "^1.12.3",  
        "connect-multiparty": "1.2.5",  
        "connect-redis": "2.3.0",  
        "cookie-parser": "^1.3.4",  
        "express": "^4.12.3",  
        "express-session": "1.11.1",  
        "lodash": "^3.8.0",  
        "mongoose": "4.0.2",  
        "morgan": "1.5.2",  
        "passport": "0.2.1",  
        "passport-local": "1.0.0",  
        "passport-twitter": "1.0.3",  
        "swig": "1.4.2"  
    }  
}
```

Se modifica: app.js,

## Recursos.js

```
var Recurso = require('../models/recursos');
var ObjectId = require('mongoose').Types.ObjectId;

var fs = require('fs');
var _ = require('lodash');
var path = require('path');
var async = require('async');

var newRecurso = new Recurso({});

exports.guardar_recurso = function(req, res, next){
  async.series({
    archivos : function(callback){
      if(req.files.file.length > 0){
        var result = _.map(req.files.file, function(file,i){
          return guardar_archivos(req, res, i, file);
        });
        callback(null, result);
      }else{
        callback(null, guardar_archivos(req, res, 0, req.files.file));
      }
    },
    datos: function(callback){
      var data = {remitente : ObjectId(req.session.passport.user._id.toString()),
                  destinatarios : req.body.destinatarios.split(','), asunto : req.body.asunto
                }
      callback(null, data);
    }
  }, function(err, result){
    if(!err){
      res.send({msj : "Fallo"});
      console.log(err);
    }
  });
};

function guardar_archivos(req, res, i, file){
  var root = path.dirname(require.main.filename);
  var originalFilename = file.originalFilename.split('.');
  var ext = originalFilename[originalFilename.length - 1];
  var nombre_archivo = newRecurso._id.toString()+'_'+i+'. '+ext;
  var newPath = root + '/public/recursos/'+nombre_archivo;
  var newFile = new fs.createWriteStream(newPath);
  var oldFile = new fs.createReadStream(file.path);
  var bytes_totales = req.headers['content-length'];
  var bytes_subidos = 0;

  oldFile.pipe(newFile);
  oldFile.on('data', function(chunk){
    bytes_subidos += chunk.length;
    var progreso = (bytes_subidos / bytes_totales) * 100;
    res.write("progress: "+ parseInt(progreso, 10)+ '%\n');
  });

  oldFile.on('end', function(){
    console.log('Carga Completa');
    res.end('Carga Completa');
  });
  return nombre_archivo;
}

function guardar_recurso(result, callback){
  if(Array.isArray(result.archivos)){
    newRecurso.archivos = result.archivos;
  }else{
    newRecurso.archivos.push(result.archivos);
  }
  newRecurso.asunto = result.datos.asunto;
  newRecurso.destinatarios = result.datos.destinatarios;
  newRecurso.remitente = result.datos.remitente;
  newRecurso.save();
  callback(newRecurso);
}
```

## Scripts.js

```
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
<script type="text/javascript" src="/vendor/angular-animate/angular-animate.js"></script>
<script type="text/javascript" src="/vendor/angular-toastr/dist/angular-toastr.js"></script>
<script type="text/javascript" src="/vendor/lodash/lodash.js"></script>
<script type="text/javascript" src="/vendor/async/lib/async.js"></script>

<script type="text/javascript" src="/app/app.js"></script>

<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/registroCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/loginCtrl.js"></script>
<script type="text/javascript" src="/app/dashboard/controllers/dashboardCtrl.js"></script>
<script type="text/javascript" src="/app/tareas/controllers/tareasCtrl.js"></script>
<script type="text/javascript" src="/app/recursos/controllers/recursosCtrl.js"></script>

<script type="text/javascript" src="/app/services/toastr.js"></script>
<script type="text/javascript" src="/app/services/session.js"></script>
<script type="text/javascript" src="/app/services/tareas.js"></script>
<script type="text/javascript" src="/app/services/recursos.js"></script>

<script type="text/javascript" src="/app/directives/files.js"></script>
```

## Recursos.html

```
<div class="mailbox row">
  <div class="col-xs-12">
    <div class="col box-solid">
      <div class="box-body">
        <div class="row">
          <div class="col-md-3 col-sm-4">
            <div class="box-header">
              <i class="fa fa-inbox"></i>
              <h3 class="box-title">Inbox</h3>
            </div>
            <a class="btn btn-block btn-primary" ui-sref="app.recursos.crear"><i class="fa fa-pencil"></i> Enviar Recursos</a>
            <div style="margin-top: 15px;">
              <ul class="nav nav-pills nav-stacked">
                <li class="header">Folders</li>
                <li ui-sref-active="active"><a href="#"><i class="fa fa-inbox"></i>Recibidos</a></li>
                <li ui-sref-active="active"><a href="#"><i class="fa fa-pencil-square-o"></i>Enviados</a></li>
              </ul>
            </div>
          </div>
          <div class="col-md-9 col-sm-8" ui-view>
            <div class="callout callout-info">
              <h4>Modulo de Recursos!</h4>
              <p>Subir y Recibir Archivos sin limite de Peso.</p>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

## Files.js

```
angular.module('Teamapp').directive('fileInput', function($parse){
  return {
    restrict : 'A',
    link : function(scope, elm, attrs){
      elm.bind('change', function(){
        $parse(attrs.fileInput)
          .assign(scope, elm[0].files)
        scope.$apply();
      });
    }
  };
});
```

## Recursos II

### El uso de Streams

Se modifica al final recursos.js

```
exports.getRecursosRecibidos = function(req, res, next){
  Recurso.find({destinatarios : req.session.passport.user.nombre_usuario})
    .populate('remitente')
    .exec(function(err, recursos){
      if(err){
        console.log(err);
      }else{
        res.send(recursos);
      }
    });
};

exports.getRecursosEnviados = function(req, res,next){
  console.log('Recursos Enviados');
  Recurso.find({remitente : req.session.passport.user._id})
    .populate('remitente')
    .exec(function(err, recursos){
      if(err){
        console.log(err);
      }else{
        res.send(recursos)
      }
    });
};
```

se modifica al final routes.js

```
app.post('/recurso', multiparty, recursos.guardar_recurso);

app.get('/recursos/recibidos', recursos.getRecursosRecibidos);

app.get('/recursos/enviados', recursos.getRecursosEnviados);
```

se modifica al final recursosCtrl.js

```
app.controller('enviadosCtrl', function($scope, RecursosService){
  RecursosService.getRecursosEnviados()
    .success(function(response){
      console.log(response);
      $scope.enviados = response;
    });
});
app.controller('recibidosCtrl', function($scope, RecursosService){
  RecursosService.getRecursosRecibidos()
    .success(function(response){
      $scope.recibidos = response;
    });
});
```

se modifica al final app.js

```
.state('app.recursos',{
  url : '/recursos',
  templateUrl : 'partials/recursos/templates/recursos.html',
  controller: 'recursosCtrl'
})
.state('app.recursos.crear',{
  url : '/crear',
  templateUrl : 'partials/recursos/templates/crear.html',
  controller: 'recursosCtrl'
})
.state('app.recursos.enviados',{
  url : '/enviados',
  templateUrl : 'partials/recursos/templates/enviados.html',
  controller: 'enviadosCtrl'
})
.state('app.recursos.recibidos',{
  url : '/recibidos',
  templateUrl : 'partials/recursos/templates/recibidos.html',
  controller: 'recibidosCtrl'
})
```

## Recursos.html

```
<div class="mailbox row">
  <div class="col-xs-12">
    <div class="col box-solid">
      <div class="box-body">
        <div class="row">
          <div class="col-md-3 col-sm-4">
            <div class="box-header">
              <i class="fa fa-inbox"></i>
              <h3 class="box-title">Inbox</h3>
            </div>
            <a class="btn btn-block btn-primary" ui-sref="app.recursos.crear"><i class="fa fa-pencil"></i>
            Enviar Recursos</a>
            <div style="margin-top: 15px;">
              <ul class="nav nav-pills nav-stacked">
                <li class="header">Folders</li>
                <li ui-sref-active="active"><a ui-sref="app.recursos.recibidos"><i class="fa fa-inbox"></i>
                Recibidos</a></li>
                <li ui-sref-active="active"><a ui-sref="app.recursos.enviados"><i class="fa fa-pencil-square-o"></i>Enviados</a></li>
              </ul>
            </div>
          </div>
        </div>
        <div class="col-md-9 col-sm-8" ui-view>
          <div class="callout callout-info">
            <h4>Modulo de Recursos!</h4>
            <p>Subir y Recibir Archivos sin limite de Peso.</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

## Recibidos.html

```
<div class="table-responsive">
  <table class="table table-mailbox">
    <tr ng-show="!recibidos.length">
      <td>No hay Recursos Compartidos Contigo.</td>
    </tr>
    <tr ng-repeat="recurso in recibidos" class="unread">
      <td class="name">{{recurso.remitente.nombre}}</td>
      <td class="subject">{{recurso.asunto}}</td>
      <td class="time">12:30 PM</td>
    </tr>
  </table>
</div>
```

## Enviados.html

```
<div class="table-responsive">
  <table class="table table-mailbox">
    <tr ng-show="!enviados.length">
      <td>No haz enviado ningun Recurso.</td>
    </tr>
    <tr ng-repeat="recurso in enviados" class="unread">
      <td class="name">{{recurso.remitente.nombre}}</td>
      <td class="subject">{{recurso.asunto}}</td>
      <td class="time">12:30 PM</td>
    </tr>
  </table>
</div>
```

## Recursos III

### Las rutas y el filtro de mensajes

se modifica al final recursosCtrl.js

```
app.controller('enviadosCtrl', function($scope, RecursosService){
    RecursosService.getRecursosEnviados()
    .success(function(response){
        console.log(response);
        $scope.enviados = response;
    });
});
app.controller('recibidosCtrl', function($scope, RecursosService){
    RecursosService.getRecursosEnviados()
    .success(function(response){
        $scope.recibidos = response;
    });
});
app.controller('detalleCtrl', function($scope, $stateParams, RecursosService){
    if($stateParams.hasOwnProperty('id_recurso')){
        var id_recurso = $stateParams.id_recurso;
        RecursosService.getDetalle({id:id_recurso})
        .success(function(response){
            $scope.recurso = response;
        });
    }
});
```

Details.html

```
<div class="box-body chat" id="chat-box" style="overflow: auto; width: auto; height: 250px;">
<div class="item">
    
    <p class="message">
        <a href="#" class="name">
            <small class="text-muted pull-right"><i class="fa fa-clock-o"></i>2:15</small>
            {{recurso.remitente.nombre}}
        </a>
        {{recurso.asunto}}
    </p>
    <div class="attachment">
        <h4>Archivos Adjuntos</h4>
        <p class="filename" ng-repeat="archivo in recurso.archivos">
            <a href="/public/recursos/{{archivo}}" download="{{archivo}}" target="_self">{{archivo}}</a>
        </p>
    </div>
</div>
</div>
```

se modifica al final app.js

```
.state('app.recursos.detalle',{
    url : '/:id_recurso',
    templateUrl : 'partials/recursos/templates/detalle.html',
    controller: 'detalleCtrl'
})
.state('registro',{
    url : '/registro',
    templateUrl : 'partials/sign/templates/registro.html',
    controller : 'registroCtrl'
})
.state('login',{
    url : '/login',
    templateUrl : 'partials/sign/templates/login.html',
    controller : 'loginCtrl'
})
});
```

Enviados.html

```
<div class="table-responsive">
    <table class="table table-mailbox">
        <tr ng-show="!enviados.length">
            <td>No haz enviado ningun Recurso.</td>
        </tr>
        <tr ng-repeat="recurso in enviados" class="unread" ui-sref="app.recursos.detalle({id_recurso : '{{recurso._id}}'})">
            <td class="name">{{recurso.remitente.nombre}}</td>
            <td class="subject">{{recurso.asunto}}</td>
            <td class="time">12:30 PM</td>
        </tr>
    </table>
</div>
```

## Recibidos.html

```
<div class="table-responsive">
  <table class="table table-mailbox">
    <tr ng-show="!recibidos.length">
      <td>No hay Recursos Compartidos Contigo.</td>
    </tr>
    <tr ng-repeat="recurso in recibidos" class="unread" ui-sref="app.recursos.detalle({id_recurso : '{{recurso._id}}', toString()})">
      <td class="name">{{recurso.remitente.nombre}}</td>
      <td class="subject">{{recurso.asunto}}</td>
      <td class="time">12:30 PM</td>
    </tr>
  </table>
</div>
```

## Recursos.js(servicio)

```
angular.module('Teamapp').factory('RecursosService', function($http){
  return{
    getRecursosRecibidos : function(){
      return $http.get('/recursos/recibidos');
    },
    getRecursosEnviados : function(){
      return $http.get('/recursos/enviados');
    },
    getDetalle : function(recurso){
      return $http.get('/recurso/'+recurso.id);
    }
  }
});
```

se modifica al final recursos.js

```
exports.getDetalleRecurso = function(req, res, next){
  console.log(req.params.id_recurso, "Detalle");
  Recurso.findOne({_id:req.params.id_recurso})
  .populate('remitente')
  .exec(function(err, recurso){
    if(!err){
      res.send(recurso);
    }else{
      console.log(recurso);
    }
  });
};
```

se modifica al final routes.js

```
app.get('/recurso/:id_recurso', recursos.getDetalleRecurso);
```

## Recursos y Timeline

### Dashboard.js(servicio)

```
angular.module('Teamapp').factory('DashboardService', function($http){
  return {
    getTimeLine : function(){
      return $http.get('/timeline');
    }
  }
});
```

### dashboardCtrl.js

```
angular.module('Teamapp').controller('dashboardCtrl', function($scope, DashboardService){
  $scope.today = new Date();
  $scope.timeline = [];

  DashboardService.getTimeLine()
  .success(function(response){
    $scope.unshiftTimeline(response);
  });

  $scope.unshiftTimeline = function(data){
    _.each(data, function(item,i){
      $scope.timeline.unshiftTimeline(item);
    });
  };
});
```

## Dashboard.html

```
1 <div class="row">
2   <div class="col-md-12">
3     <ul class="timeline">
4       <li class="time-label">
5         <span class="bg-red">{{today | date: 'dd MM yyyy'}}</span>
6       </li>
7
8       <li ng-repeat="item in timeline">
9         <i class="fa bg-blue" ng-class="{'fa-check' : item.tipo == 'tarea', 'fa-envelope' : item.tipo == 'recurso'}"></i>
10        <div class="timeline-item" ng-show="item.tipo == 'tarea'">
11          <h3 class="timeline-header"><a href="javascript:void(0);">{{item.usuario.nombre}}</a>{{item.accion}}</h3>
12          <div class="timeline-body">
13            {{item.descripcion}}
14          </div>
15        </div>
16
17        <div class="timeline-item" ng-show="item.tipo == 'recurso'">
18          <h3 class="timeline-header"><a href="javascript:void(0);">{{item.recurso.remitente.nombre}}</a>{{item.accion}} con {{item.recurso.destinatarios.join(',')}}</h3>
19          <div class="timeline-body">
20            {{item.descripcion}}
21          </div>
22        </div>
23      </li>
24    </ul>
25  </div>
26</div>
```

## Timeline.js

```
var Timeline = require('../models/timeline');
var Tareas = require('../models/tareas');

var ObjectId = require('mongoose').Types.ObjectId;
var _ = require('lodash');
var async = require('async');

exports.tareaFinalizada = function(req, res, next){
  var items = function(tarea, callback){
    var timeline = new Timeline({
      usuario : ObjectId(tarea.usuario.toString()),
      tarea : ObjectId(tarea._id.toString()),
      accion : 'Finalizó una Tarea',
      descripcion : tarea.descripcion,
      tipo : 'tarea'
    });

    timeline.save(function(err, item){
      if(!err){
        console.log("Accion Guardada");
        return callback(null, item);
      }
    });
  };
  async.map(req.body.tareas, items, function(err, result){
    async.waterfall([
      function(callback){
        Timeline.populate(result, {path : 'usuario', model : 'Usuario'}, function(err, items){
          callback(null, items);
        });
      }
    ], function(err, result){
      res.json(result);
    });
  });
}
```

```

        function(items, callback){
            Timeline.populate(items, {path : 'tarea', model : 'Tarea'}, function(err, items){
                callback(null, items);
            });
        },function(err, data){
            if(!err){
                res.send(data);
            }else{
                console.log(err);
            }
        });
    });
};

exports.recursoEnviado = function(req, res, next){
    var timeline = new Timeline({
        recurso : req.body.recurso._id,
        accion : 'Compartio un Recurso',
        descripcion : req.body.recurso.asunto,
        tipo : 'recurso'
    });

    timeline.save(function(err, recurso){
        if(!err){
            console.log("Accion Guardada");
            console.log("recurso");
        }
    });
};

exports.getTimeline = function(req, res, next){
    var d = new Date();
    var anio = d.getFullYear();
    var mes = d.getMonth();
    var dia = d.getDate();
    console.log("Fecha: ", new Date(anio, mes, dia));

    Timeline.find({fecha : {$gte : new Date(anio, mes, dia)}})
    .populate('usuario tarea recurso')
    .exec(function(err, docs){
        if(!err){
            Timeline.populate(docs, {path : 'recurso.remitente', model: 'Usuario'}, function(err, items){
                res.send(items);
            })
        }else{
            console.log(err);
        }
    });
};

```

se modifica al final routes.js

```

app.post('/tareas/finalizadas', tareas.guardarFinalizadas, timeline.tareaFinalizada);

app.post('/recurso', multiparty, recursos.guardar_recurso, timeline.recursoEnviado);

app.get('/recursos/recibidos', recursos.getRecursosRecibidos);

app.get('/recursos/enviados', recursos.getRecursosEnviados);

app.get('/recurso/:id_recurso', recursos.getDetalleRecurso);

app.get('/timeline', timeline.getTimeline);

app.get('*', function(req, res){
    res.render('index');
});

```

se modifica al final tareas.js

```

exports.guardarFinalizadas = function(req, res, next){
  var ids = req.body.ids;

  Tareas.find({_id: {$in:ids}})
  .exec(function(err, tareas){
    if(err){
      console.log(err);
    }else{
      _.each(tareas, function(tarea){
        tarea.finalizada.estado = true;
        tarea.finalizada.fecha = new Date();
        tarea.save();
      });
      req.body.tareas = tareas;
      res.send(tareas);
      next();
    }
  });
};

```

## Scripts.js

```

<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
<script type="text/javascript" src="/vendor/angular-animate/angular-animate.js"></script>
<script type="text/javascript" src="/vendor/angular-toastr/dist/angular-toastr.js"></script>
<script type="text/javascript" src="/vendor/lodash/lodash.js"></script>
<script type="text/javascript" src="/vendor/async/lib/async.js"></script>

<script type="text/javascript" src="/app/app.js"></script>

<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/registroCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/loginCtrl.js"></script>
<script type="text/javascript" src="/app/dashboard/controllers/dashboardCtrl.js"></script>
<script type="text/javascript" src="/app/tareas/controllers/tareasCtrl.js"></script>
<script type="text/javascript" src="/app/recursos/controllers/recursosCtrl.js"></script>

<script type="text/javascript" src="/app/services/toastr.js"></script>
<script type="text/javascript" src="/app/services/session.js"></script>
<script type="text/javascript" src="/app/services/tareas.js"></script>
<script type="text/javascript" src="/app/services/recursos.js"></script>
<script type="text/javascript" src="/app/services/dashboard.js"></script>

<script type="text/javascript" src="/app/directives/files.js"></script>

```

## Timeline

### Timeline en tiempo real

## Servidor.js

```

var express = require('express'),
app = express(),
server = require('http').createServer(app);

var config = {
  rootPath : __dirname
};

require('./server/config/express')(app, config);
require('./server/config/routes')(app);
require('./server/config/socket')(server);

server.listen(3000, function(){
  console.log("Servidor Corriendo en el puerto: 3000");
});

```

## dashboardCtrl.js

```

angular.module('Teamapp').controller('dashboardCtrl', function($scope, DashboardService){
  $scope.today = new Date();
  $scope.timeline = [];

  DashboardService.getTimeline()
  .success(function(response){
    $scope.unshiftTimeline(response);
  });

  $scope.unshiftTimeline = function(data){
    _.each(data, function(item,i){
      $scope.timeline.unshift(item);
    });
  }

  Socket.on('nueva:accion', function(data){
    console.log('Data Completo', data);
    $scope.unshiftTimeline(data);
    console.log($scope.timeline);
  });
});

```

Se modifica al final recursosCtrl.js

```

angular.module('Teamapp').controller('recursosCtrl', function($scope, $http, $state, ToastService, RecursosService, Soc
  $scope.filesChanged = function(elm){
    $scope.files = elm.files;
    $scope.$apply();
  }

  $scope.uploadFile = function(){
    var fd = new FormData();
    angular.forEach($scope.files, function(file){
      fd.append('file', file);
    });
    fd.append('destinatarios', $scope.destinatarios);
    fd.append('asunto', $scope.asunto);

    $http.post('/recurso', fd,{
      transformRequest: angular.identity,
      headers : {'Content-Type' : undefined}
    })
    .success(function(success){
      Socket.emit('nuevo:recurso', response);
      ToastService.success('Enviado Correctamente');
      $state.transitionTo('app.recursos');
    });
  };
});

app.controller('enviadosCtrl', function($scope, RecursosService){
  RecursosService.getRecursosEnviados()
  .success(function(response){
    $scope.enviados = response;
  });
});

```

Socket.js

```

angular.module('Teamapp').factory('Socket', function($rootScope){
  var socket = io.connect();

  return {
    on: function(eventName, callback){
      socket.on(eventName, function(){
        var args = arguments;
        $rootScope.$apply(function(){
          callback.apply(socket, args);
        });
      });
    },
    emit: function(eventName, data, callback){
      socket.emit(eventName, data, function(){
        var args = arguments;
        $rootScope.$apply(function(){
          if(callback){
            callback.apply(socket, args);
          }
        });
      });
    },
    socket: function(){
      return socket
    }
  };
});

```

## tareasCtrl.js

```
angular.module('Teamapp').controller('tareasCtrl', function($scope, TareasService, Socket){
    var checkeados = [];

    $scope.tareas = [];
    $scope.tareas_finalizadas = [];
    $scope.item_master = {descripcion : "", fecha : "", finalizada : false};
    $scope.date = new Date();

    $scope.guardar = function(){
        TareasService.guardarTareas({descripcion: $scope.item.descripcion})
        .then(function(response){
            if(response.data.success){
                $scope.tareas.push(response.data.tarea);
                angular.copy($scope.item_master, $scope.item);
            }
        });
    }

    $scope.enviar_finalizadas = function(){
        var ids = _.pluck(checkeados, '_id');
        TareasService.guardarFinalizadas({ids : ids})
        .then(function(response){
            _.each(response.data, function(item){
                var item = item;
                _.remove($scope.tareas, function(tarea){
                    return tarea._id == item._id;
                });
            });
            Socket.emit('nueva:tarea', response.data.populated);
        });
    }
})
```

## Socket.js

```
module.exports = function(server){
    var io = require('socket.io')(server);

    io.on('connection', function(socket){
        socket.on('nueva:tarea', function(data){
            io.emit('nueva:accion', data);
        });

        socket.on('nueva:recurso', function(data){
            io.emit('nueva:accion', [data]);
        });
    });
}
```

Se modifica varias partes de Recurso.js

Se modifica timeline.js

```
async.map(req.body.tareas, items, function(err, result){
    async.waterfall([
        function(callback){
            Timeline.populate(result, {path : 'usuario', model : 'Usuario'}, function(err, items){
                callback(null, items);
            });
        },
        function(items, callback){
            Timeline.populate(items, {path : 'tarea', model : 'Tarea'}, function(err, items){
                callback(null, items);
            });
        }
    ],function(err, data){
        if(!err){
            res.send({populated : data, lean : req.body.tareas});
            console.log("Accion Guardada");
            console.log("tarea");
        }else{
            console.log(err);
        }
    });
}, {});
```

## Scritps.js

```
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
<script type="text/javascript" src="/vendor/angular-animate/angular-animate.js"></script>
<script type="text/javascript" src="/vendor/angular-toastr/dist/angular-toastr.js"></script>
<script type="text/javascript" src="/vendor/lodash/lodash.js"></script>
<script type="text/javascript" src="/vendor/async/lib/async.js"></script>

<script type="text/javascript" src="/socket.io/socket.io.js"></script>
<script type="text/javascript" src="/app/app.js"></script>

<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/registroCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/loginCtrl.js"></script>
<script type="text/javascript" src="/app/dashboard/controllers/dashboardCtrl.js"></script>
<script type="text/javascript" src="/app/tareas/controllers/tareasCtrl.js"></script>
<script type="text/javascript" src="/app/recursos/controllers/recursosCtrl.js"></script>

<script type="text/javascript" src="/app/services/toastr.js"></script>
<script type="text/javascript" src="/app/services/session.js"></script>
<script type="text/javascript" src="/app/services/tareas.js"></script>
<script type="text/javascript" src="/app/services/recursos.js"></script>
<script type="text/javascript" src="/app/services/dashboard.js"></script>
<script type="text/javascript" src="/app/services/socket.js"></script>

<script type="text/javascript" src="/app/directives/files.js"></script>
```

```
{
  "name": "Modulacion",
  "version": "1.0.0",
  "description": "",
  "main": "Servidor.js",
  "scripts": {
    "test": "null"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "async": "^0.9.0",
    "body-parser": "^1.12.3",
    "connect-multiparty": "^1.2.5",
    "connect-redis": "^2.3.0",
    "cookie-parser": "1.3.4",
    "express": "4.12.3",
    "express-session": "1.11.1",
    "lodash": "3.8.0",
    "mongoose": "4.0.2",
    "morgan": "1.5.2",
    "passport": "0.2.1",
    "passport-local": "1.0.0",
    "passport-twitter": "1.0.3",
    "socket.io": "1.3.5",
    "swig": "1.4.2"
  }
}
```

## Chat I

### Crear las rutas del sistema de chat

- ✓ Hacer uso de “Socket.io” para el envío de mensajes entre todos los usuarios en tiempo real.
- ✓ Mostrar lista de usuarios conectados al iniciar sesión.
- ✓ Filtrar el envío de mensajes entre dos usuarios.
- ✓ Implementar el “Socket.io” para diferenciar el envío entre dos y múltiples usuarios.

## Socket.js

```

angular.module('Teamapp').factory('Socket', function($rootScope, Session){
  var socket = io.connect();

  socket.on('connect', function(){
    Session.getUsuario()
      .then(function(response){
        var user = response.data.user;
        socket.emit('nuevo:usuario', user);
      });
  });

  return {
    on: function(eventName, callback){
      socket.on(eventName, function(){
        var args = arguments;
        $rootScope.$apply(function(){
          callback.apply(socket, args);
        });
      });
    },
    emit: function(eventName, data, callback){
      socket.emit(eventName, data, function(){
        var args = arguments;
        $rootScope.$apply(function(){
          if(callback){
            callback.apply(socket, args);
          }
        });
      });
    },
    socket: function(){
      return socket
    }
  };
});

```

## Chat.html

```

<div class="box box-solid">
  <div class="box-body">
    <div class="row">
      <div class="col-md-3 col-sm-4">
        <div class="box-header">
          <i class="fa fa-inbox"></i>
          <h3 class="box-title">Chat</h3>
        </div>
        <div style="margin-top: 15px;">
          <ul class="nav nav-pills nav-stacked" style="overflow:auto;height:300px;">
            <li class="header">Equipo</li>
            <li><a href="#"><i class="fa fa-group zz"></i>General</a></li>

            <li ui-sref-active="active" ng-repeat="usuario in usuarios_conectados">
              <a ng-click="goToChat(usuario._id)">
                <i class="fa fa-circle text-sucess"></i>{{usuario.nick}}
              </a>
            </li>
          </ul>
        </div>
      </div>
      <div class="col-md-9 col-sm-8 ng-scope" ui-view="">
        <div class="callout callout-info">
          <h4>Bienvenido al Chat del Equipo</h4>
          <p>Selecciona a la Izquierda con quien Chatear.</p>
        </div>
      </div>
    </div>
  </div>
</div>

```

## chatCtrl.js

```

angular.module('Teamapp').controller('chatCtrl', function($scope, socket, Session){
  $scope.usuarios_conectados = [];

  if($scope.usuarios_conectados.length <= 0){
    Socket.emit('usuario');
  }

  Socket.on('usuarios:lista', function(usuarios){
    Session.getUsuario()
      .then(function(response){
        var user = response.data.user;
        var conectados = _.reject(usuarios, {_id : user._id});
        angular.copy(conectados, $scope.usuarios_conectados);
      });
  });
});

```

## Socket.js(config)

```
var _ = require('lodash');
var usuarios = [];

module.exports = function(server){
  var io = require('socket.io')(server);

  io.on('connection', function(socket){
    socket.on('nueva:tarea', function(data){
      io.emit('nueva:accion', data);
    });

    socket.on('nueva:recurso', function(data){
      io.emit('nueva:accion', [data]);
    });

    socket.on('disconnect', function(){
      var list = _.reject(usuarios, function(user){
        return user.socket === socket.id;
      });
      socket.emit('usuarios:lista', usuarios);
    });

    socket.on('nuevo:usuario', function(data){
      var index = _.findIndex(usuarios, {_id : data.user._id});
      if(index > -1){
        usuarios[index].socket = socket.id;
      }else{
        usuarios.push({_id : data.user._id, socket : socket.id, nombre : data.user.nombre, nick : data.user.nom });
      }
      console.log(usuarios);
      socket.broadcast.emit('usuarios:lista', usuarios);
    });

    socket.on('usuarios', function(data){
      socket.emit('usuarios:lista', usuarios);
    });
  });
};
```

## Script.html

```
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script type="text/javascript" src="/js/AdminLTE/app.js"></script>
<script type="text/javascript" src="/vendor/angular/angular.js"></script>
<script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
<script type="text/javascript" src="/vendor/angular-animate/angular-animate.js"></script>
<script type="text/javascript" src="/vendor/angular-toastr/dist/angular-toastr.js"></script>
<script type="text/javascript" src="/vendor/lodash/lodash.js"></script>
<script type="text/javascript" src="/vendor/async/lib/async.js"></script>

<script type="text/javascript" src="/socket.io/socket.io.js"></script>
<script type="text/javascript" src="/app/app.js"></script>

<script type="text/javascript" src="/app/index/controllers/indexCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/registroCtrl.js"></script>
<script type="text/javascript" src="/app/sign/controllers/loginCtrl.js"></script>
<script type="text/javascript" src="/app/dashboard/controllers/dashboardCtrl.js"></script>
<script type="text/javascript" src="/app/tareas/controllers/tareasCtrl.js"></script>
<script type="text/javascript" src="/app/recursos/controllers/recursosCtrl.js"></script>
<script type="text/javascript" src="/app/chat/controllers/chatCtrl.js"></script>

<script type="text/javascript" src="/app/services/toastr.js"></script>
<script type="text/javascript" src="/app/services/session.js"></script>
<script type="text/javascript" src="/app/services/tareas.js"></script>
<script type="text/javascript" src="/app/services/recursos.js"></script>
<script type="text/javascript" src="/app/services/dashboard.js"></script>
<script type="text/javascript" src="/app/services/socket.js"></script>

<script type="text/javascript" src="/app/directives/files.js"></script>
```

## Se agrega al final App.js

```
.state('app.chat',{
  url : '/chat',
  templateUrl : 'partials/chat/templates/chat.html',
  controller: 'chatCtrl'
})
```

## Chat II

### Mensajes de chat sin filtro de envío

Layout.html

```
<!DOCTYPE html>
<html ng-app="Teamapp">
<head>
  <base href="/">
  {% include 'links.html' %}
</head>
<body class="skin-blue">
  {% block content %}
  {% endblock %}
  {% include 'scripts.html' %}
</body>
</html>
```

Se modifica socket.js

```
socket.on('nuevo:mensaje:general', function(mensaje){
  io.emit('mensaje:general', mensaje);
});

socket.on('usuarios', function(data){
  socket.emit('usuarios:lista', usuarios);
});
};

});
```

chatCtrl.js

```
var app = angular.module('Teamapp');

app.controller('chatCtrl', function($scope, $stateParams, $state, Socket, Session){
  $scope.usuarios_conectados = [];
  $scope.chat = "general";
  $scope.messagesList = new Object();
  $scope.messagesList[$scope.chat] = [];

  if($scope.usuarios_conectados.length <= 0){
    Socket.emit('usuarios');
  }

  $scope.enviarMensajeGeneral = function(){
    Session.getUsuario()
    .then(function(response){
      var data = {};
      var nombre = response.data.user.user.nombre;
      data = {contenido : $scope.mensaje, tipo : 'general', nombre : nombre};
      Socket.emit('nuevo:mensaje:general', data);
      $scope.mensaje = "";
    });
  }

  $scope.goToChat = function(){
    $state.go('app.chat.general');
  }

  Socket.on('usuarios:lista', function(usuarios){
    Session.getUsuario()
    .then(function(response){
      var user = response.data.user.user;
      var conectados = _.reject(usuarios, {_id : user._id});
      angular.copy(conectados, $scope.usuarios_conectados);
    });
  });

  Socket.on('mensaje:general', function(mensaje){
    if($scope.messagesList && $scope.chat){
      $scope.messagesList[$scope.chat].push(mensaje);
    }
  });
});
```

## General.html

```
<div class="box box-primary">
  <div class="box-header">
    <i class="fa fa-comments-o"></i>
    <h3 class="box-title">Chat General</h3>
  </div>
  <div class="box-body chat" id="chat-box" style="overflow:auto; height:300px;" scroll-glue>
    <div class="item" ng-repeat="message in messagesList[chat]">
      
      <p class="message">
        <a href="#" class="name">
          <small class="text-muted pull-right"><i class="fa fa-clock-o"></i>2:15</small>
          {{message.nombre}}
        </a>
        {{message.contenido}}
      </p>
    </div>
  </div>
  <div class="box-footer">
    <form ng-submit="enviarMensajeGeneral()">
      <div class="input-group">
        <input name="mensaje" ng-model="mensaje" class="form-control" placeholder="Escribir Mensaje..." />
        <div class="input-group-btn">
          <button type="submit" class="btn btn-success"><i class="fa fa-plus"></i></button>
        </div>
      </div>
    </form>
  </div>
</div>
```

## Chat.html

```
<div class="col-xs-12">
  <div class="box box-solid">
    <div class="box-body">
      <div class="row">
        <div class="col-md-3 col-sm-4">
          <div class="box-header">
            <i class="fa fa-inbox"></i>
            <h3 class="box-title">Chat</h3>
          </div>
          <div style="margin-top: 15px;">
            <ul class="nav nav-pills nav-stacked" style="overflow:auto;height:300px;">
              <li class="header">Equipo</li>
              <li><a href="#"><i class="fa fa-group zz"></i>General</a></li>

              <li ui-sref-active="active" ng-repeat="usuario in usuarios_conectados" href="javascript:void(0);">
                <a ng-click="goToChat(usuario._id)">
                  <i class="fa fa-circle text-sucess"></i>{{usuario.nick}}
                </a>
              </li>
            </ul>
          </div>
        </div>
        <div class="col-md-9 col-sm-8 ng-scope" ui-view="">
          <div class="callout callout-info">
            <h4>Bienvenido al Chat del Equipo</h4>
            <p>Selecciona a la Izquierda con quien Chatear.</p>
          </div>
        </div>
      </div>
    </div>
  </div>
```

## Se modifica app.js

```
var app = angular.module('Teamapp',['ui.router', 'ngAnimate', 'toastr']);

app.config(['$stateProvider','$urlRouterProvider", "$locationProvider", function($stateProvider, $urlRouterProvider, $locationProvider){
  $urlRouterProvider.otherwise('/app/dashboard');
  $locationProvider.html5Mode(true);

  $stateProvider
    .state('app',{
      url : '/app',
      templateUrl : 'partials/index/templates/index.html',
      controller : 'indexCtrl'
    })
    .state('app.dashboard',{
      url : '/dashboard',
      templateUrl : 'partials/dashboard/templates/dashboard.html',
      controller : 'dashboardCtrl'
    })
    .state('app.chat',{
      url : '/chat',
      templateUrl : 'partials/chat/templates/chat.html',
      controller: 'chatCtrl'
    })
    .state('app.chat.general',{
      url : '/general',
      templateUrl : 'partials/chat/templates/general.html',
      controller: 'chatCtrl'
    })
}])
```

## Chat III

### crear sistema de chat individual

se modifica chatCtrl.js

```
var app = angular.module('Teamapp');

app.controller('chatCtrl', function($scope, $stateParams, $state, Socket, Session, ChatService){
    $scope.usuarios_conectados = [];
    $scope.chat = null;
    $scope.messagesList = [];
    $scope.message = {};

    $scope.messagesG = [];
    $scope.messages = [];

    if($scope.usuarios_conectados.length <= 0){
        Socket.emit('usuarios');
    }

    $scope.enviarMensajeIndividual = function(){
        Session.getUsuario()
        .then(function(response){
            var data = {};
            var nombre = response.data.user.user.nombre;
            data = {contenido : $scope.mensaje, tipo : 'individual', destinatario : {_id : $scope.otro._id.toString()}, remitente : {nombre:nombre}, chat : $scope.chat};
            $scope.messages.push(data);
            $scope.messagesList[$scope.chat] = $scope.messages;
            Socket.emit('nuevo:mensaje:individual', data);
            $scope.mensaje = "";
        });
    }

    $scope.goToChat = function(destino){
        ChatService.crearDarConversacion({destinatario : destino})
        .success(function(response){
            if(response.chat.tipo == "individual"){
                $state.go('app.chat.individual', {id_chat : response.chat._id});
                $scope.yo = response.yo;
                $scope.otro = response.otro;
            }else{
                $state.go('app.chat.general');
            }
        });
    }

    $scope.getTipoChat = function(callback){
        var id = $stateParams.hasOwnProperty('id_chat');
        if(id){
            callback($stateParams.id_chat);
        }
    };

    $scope.whereIAm = function(){
        $scope.getTipoChat(function(tipo){
            $scope.chat = tipo;
        });
    }();

    $scope.getMessagesChat = function(list){
        var messages = list[$scope.chat];
        return messages;
    };

    Socket.on('mensaje:individual', function(mensaje){
        if($scope.messagesList){
            if(mensaje.chat && $scope.chat){
                if($scope.messagesList.hasOwnProperty(mensaje.chat)){
                    $scope.messagesList[mensaje.chat].push(mensaje);
                }else{
                    $scope.messagesList[mensaje.chat] = new Array();
                    $scope.messagesList[mensaje.chat].push(mensaje);
                }
                console.log($scope.messagesList, $scope.chat, mensaje);
            }
        }
    });

    $scope.$on('$destroy', function(event){
        Socket.init();
    });
});
```

## Chat.html

```
<div class="col-xs-12">
  <div class="box box-solid">
    <div class="box-body">
      <div class="row">
        <div class="col-md-3 col-sm-4">
          <div class="box-header">
            <i class="fa fa-inbox"></i>
            <h3 class="box-title">Chat</h3>
          </div>
          <div style="margin-top: 15px;">
            <ul class="nav nav-pills nav-stacked" style="overflow:auto;height:300px;">
              <li class="header">Equipo</li>
              <li ui-sref-active="active"><a ng-click="goToChat('general')" href="javascript:void(0);"><i class="fa fa-group zz"></i>General</a></li>

              <li ui-sref-active="active" ng-repeat="usuario in usuarios_conectados" href="javascript:void(0);">
                <a ng-click="goToChat(usuario._id)">
                  <i class="fa fa-circle text-sucess"></i>{{usuario.nick}}
                </a>
              </li>
            </ul>
          </div>
        </div>
        <div class="col-md-9 col-sm-8 ng-scope" ui-view="">
          <div class="callout callout-info">
            <h4>Bienvenido al Chat del Equipo</h4>
            <p>Selecciona a la Izquierda con quien Chatear.</p>
          </div>
        </div>
      </div>
    </div>
```

## General.html

```
<div class="box box-primary">
  <div class="box-header">
    <i class="fa fa-comments-o"></i>
    <h3 class="box-title">Chat General</h3>
  </div>
  <div class="box-body chat" id="chat-box" style="overflow:auto; height:300px;" scroll-glue>
    <div class="item" ng-repeat="message in messagesG">
      
      <p class="message">
        <a href="#" class="name">
          <small class="text-muted pull-right"><i class="fa fa-clock-o"></i>2:15</small>
          {{message.nombre}}
        </a>
        {{message.contenido}}
      </p>
    </div>
  </div>
  <div class="box-footer">
    <form ng-submit="enviarMensajeGeneral()">
      <div class="input-group">
        <input name="mensaje" ng-model="mensaje" class="form-control" placeholder="Escribir Mensaje..." />
        <div class="input-group-btn">
          <button type="submit" class="btn btn-success"><i class="fa fa-plus"></i></button>
        </div>
      </div>
    </form>
  </div>
</div>
```

## Individual.html

```
<div class="box box-primary">
  <div class="box-header">
    <i class="fa fa-comments-o"></i>
    <h3 class="box-title">{{otro.nombre}}</h3>
  </div>
  <div class="box-body chat" id="chat-box" style="overflow:auto; height:300px;" scroll-glue>
    <div class="item" ng-repeat="message in getMessagesChat(messagesList)">
      
      <p class="message">
        <a href="#" class="name">
          <small class="text-muted pull-right"><i class="fa fa-clock-o"></i>2:15</small>
          {{message.remitente.nombre}}
        </a>
        {{message.contenido}}
      </p>
    </div>
  </div>
  <div class="box-footer">
    <form ng-submit="enviarMensajeGeneral()">
      <div class="input-group">
        <input name="mensaje" ng-model="mensaje" class="form-control" placeholder="Escribir Mensaje..." />
        <div class="input-group-btn">
          <button type="submit" class="btn btn-success"><i class="fa fa-plus"></i></button>
        </div>
      </div>
    </form>
  </div>
</div>
```

Se modifica Socket.js

```
init : function(){
    socket.removeAllListeners();
```

Chat.js

```
angular.module('Teamapp').factory('ChatService', function($http){
    return {
        crearDarConversacion : function(destinatario){
            return $http.post('/conversacion', destinatario);
        }
    };
});
```

Se modifica app.js

```
.state('app.chat.individual',{
    url : '/id_chat',
    templateUrl : 'partials/chat/templates/individual.html',
    controller: 'chatCtrl'
})
```

Se modifica routes.js

```
var chat = require('../controllers/chat');

app.post('/conversacion', chat.crear_dar_conversacion);
```

Se modifica Socket.js(config)

```
socket.on('nuevo:mensaje:individual', function(mensaje){
    var index = _.findIndex(usuarios, {_id : mensaje.destinatario._id});
    if(index > -1){
        socket.broadcast.in(usuarios[index].socket).emit('mensaje:individual', mensaje);
    }
});
```

Chat.js(servicio)

```
var Chat = require('../models/chat');
var Usuario = require('../models/usuarios');
var ObjectId = require('mongoose').Types.ObjectId;
var async = require('async');
var _ = require('lodash');

exports.crear_dar_conversacion = function(req, res, next){
    if(req.body.destinatario != "general"){
        async.waterfall([
            function(callback){
                Chat.findOne({$or : [{$_and : [{destinatario : req.session.passport.user._id.toString()}, {destinatario : req.body.destinatario}], populate: 'remitente'}, {$_and : [{destinatario : req.session.passport.user._id.toString()}, {remitente : req.body.destinatario}], populate: 'destinatario'}]}, exec(function(err, chat){
                    callback(null, chat);
                });
            },
            function(chat, callback){
                if(chat){
                    var data = whoIsMe(req.session.passport.user, chat);
                    callback(null, data);
                }else{
                    var chat = new Chat({
                        remitente : req.session.passport.user._id.toString(),
                        destinatario : req.body.destinatario,
                        tipo : 'individual'
                    });
                    chat.save(function(err, chat){
                        if(!err){
                            async.waterfall([
                                function(cb){
                                    Chat.populate(chat, {path : 'destinatario', model : 'Usuario'}, function(err, r1){
  cb(null, r1);
                                    });
                                },
                                function(r1, cb){
                                    Chat.populate(r1, {path : 'remitente', model : 'Usuario'}, function(err, r2){
  cb(null, r2);
                                    });
                                },
                                function(err, results){
                                    var data = whoIsMe(req.session.passport.user, results);
                                    callback(null, data);
                                }
                            ]);
                        }
                    });
                }
            },
            function(results, callback){
                res.send(results);
            }
        ], function(err, results){
            res.send(results);
        });
    }
};
```

```

    async.waterfall([
      function(callback){
        Chat.findOne({tipo : 'general'})
          .exec(function(err, chat){
            callback(null, chat);
          });
      },
      function(chat, callback){
        if(chat){
          callback(null, chat);
        }else{
          var chat = new Chat({
            tipo : 'general'
          });
          chat.save(function(err, chat){
            callback(null, chat);
          });
        }
      },
      function(err, results){
        res.send({chat : results});
      });
    });

    function whoIsMe(usuario, chat){
      var data = {chat : chat, yo : {}, otro : {}};
      if(chat.destinatario._id == usuario._id){
        data.yo = chat.destinatario;
        data.otro = chat.remitente;
      }else{
        data.yo = chat.remitente;
        data.otro = chat.destinatario;
      }
      return data;
    }
  }
}

```

Se modifica script.js

```

<script type="text/javascript" src="/app/services/toastr.js"></script>
<script type="text/javascript" src="/app/services/session.js"></script>
<script type="text/javascript" src="/app/services/tareas.js"></script>
<script type="text/javascript" src="/app/services/recursos.js"></script>
<script type="text/javascript" src="/app/services/dashboard.js"></script>
<script type="text/javascript" src="/app/services/chat.js"></script>
<script type="text/javascript" src="/app/services/socket.js"></script>

```

## Chat IV

### Filtro de mensajes para chats individuales

Se modifica chat.js

```

exports.enviar_mensaje = function(req, res, next){
  if(req.body.tipo == "individual"){
    Chat.findOne({_id : req.body.chat}, {mensajes : {$slice : 0}})
      .exec(function(err,chat){
        if(!err){
          var datos = {contenido : req.body.contenido, destinatario : req.body.destinatario._id, remitente : req
          chat.mensajes.push(datos);
          chat.save(function(err, chat){
            if(!err){
              async.waterfall([
                function(callback){
                  Usuario.populate(chat, {path : 'mensajes.remitente'}),
                    function(err, r1){
                      if(err){
                        console.log("Error populate remitente: "+err);
                      }else{
                        console.log(r1);
                      }
                      callback(null, r1);
                    }
              ]);
            }
          });
        }
      });
    }
  }
}

```

```

        function(r1, callback){
            Usuario.populate(r1, {path : 'mensajes.destinatario'},
                function(err, r2){
                    if(err){
                        console.log("Error populate destinatario: "+err);
                    }else{
                        console.log(r1);
                    }
                    callback(null, r2);
                });
        }
    ], function(err, mensaje){
        if(!err){
            res.send(mensaje);
        }else{
            res.send({success : false, message : err});
        });
    });
}else{
    res.send({success : false, message : err});
}
});

}else if(req.body.tipo == "general"){
    Chat.findOne({tipo : "general"})
    .exec(function(err, chat){
        if(!err){
            var datos = {remitente : req.body.remitente, destinatario : req.body.destinatario, contenido : req.body.contenido};
            chat.mensajes.push(datos);
            chat.save(function(err, chat){
                if(!err){
                    Chat.populate(chat, {path : 'remitente', model : 'Usuario'}, function(err, chat){
                        res.send(chat);
                    });
                };
            });
        };
    });
}
};

exports.get_mensajes_individuales = function(req, res, next){
    Chat.findOne({_id : req.params.id_chat})
    .populate('remitente')
    .populate('destinatario')
    .populate('mensajes.destinatario')
    .exec(function(err, chat){
        if(!err){
            var data = whoIsMe(req.session.passport.user, chat);
            res.send(data);
        }else{
            console.log(err);
        };
    });
};

```

Se modifica routes.js

```

app.post('/mensaje', chat.enviar_mensaje);

app.get('/mensajes/general', chat.get_mensajes_generales);

app.get('/mensajes/:id_chat', chat.get_mensajes_individuales);

```

Se modifica app.js

Chat.js

```

angular.module('Teamapp').factory('ChatService', function($http){
    return {
        crearDarConversacion : function(destinatario){
            return $http.post('/conversacion', destinatario);
        },
        enviarMensaje : function(data){
            return $http.post('/mensaje', data)
        },
        getMensajesGenerales : function(){
            return $http.get('/mensajes/general');
        },
        getMensajesIndividuales : function(id){
            return $http.get('/mensajes/'+id.chat);
        }
    }
});

```

## Individual.html

```

<div class="box box-primary">
    <div class="box-header">
        <i class="fa fa-comments-o"></i>
        <h3 class="box-title">{{otro.nombre}}</h3>
    </div>
    <div class="box-body chat" id="chat-box" style="overflow:auto; height:300px;" scroll-glue>
        <div class="item" ng-repeat="message in messagesList[chat]">
            
            <p class="message">
                <a href="#" class="name">
                    <small class="text-muted pull-right"><i class="fa fa-clock-o"></i>2:15</small>
                    {{message.remitente.nombre}}
                </a>
                {{message.contenido}}
            </p>
        </div>
    </div>
    <div class="box-footer">
        <form ng-submit="enviarMensajeIndividual()">
            <div class="input-group">
                <input name="mensaje" ng-model="mensaje" class="form-control" placeholder="Escribir Mensaje..." />
                <div class="input-group-btn">
                    <button type="submit" class="btn btn-success"><i class="fa fa-plus"></i></button>
                </div>
            </div>
        </form>
    </div>
</div>

```

## Se modifica varias chatCtrl.js

```

$scope.enviarMensajeGeneral = function(){
    var data = {};
    data = {contenido : $scope.mensaje, tipo : 'general'};
    ChatService.enviarMensaje(data)
    .then(function(response){
        data.remitente = response.data.remitente;
        Socket.emit('nuevo:mensaje:general', data);
        $scope.mensaje = "";
    });
}

$scope.enviarMensajeIndividual = function(){
    var data = {};
    data = {contenido : $scope.mensaje, tipo : 'individual', destinatario : {_id : $scope.otro._id.toString()})
    ChatService.enviarMensaje(data)
    .success(function(response){
        console.log(response);
        data.remitente = response.mensajes[0].remitente;
        $scope.setChat(data);
        Socket.emit('nuevo:mensaje:individual', data);
        $scope.mensaje = "";
    })
    .error(function(response){
        console.error(response);
    });
}

```

```
$scope.getTipoChat = function(callback){
  var id = $state.params.hasOwnProperty('id_chat');
  if(id){
    callback($state.params.id_chat);
  }else{
    callback("general");
  }
};

$scope.whereIAm = function(){
  $scope.getTipoChat(function(tipo){
    $scope.chat = tipo;
    if(typeof callback == "function"){
      callback($scope.chat);
    }
  });
};

$scope.getMensajes = function(){
  $scope.whereIAm(function(chat){
    if(chat == "general"){
      $scope.goToChat("general");
      ChatService.getMensajesGenerales()
      .success(function(response){
        console.log(response);
        $scope.messagesG = response[0].mensajes;
      });
    }else{
      ChatService.getMensajesIndividuales({chat : $scope.chat})
      .success(function(response){
        $scope.goToChat(response.otro._id);
        _.each(response.chat.mensajes, function(mensajes){
          mensaje.chat = response.chat._id;
        });
        $scope.messagesList[$scope.chat] = response.chat.mensajes;
      });
    }
  });
};

$scope.setChat = function(mensaje){
  if($scope.messagesList){
    if(mensaje.chat && $scope.chat){
      if($scope.messagesList.hasOwnProperty(mensaje.chat)){
        $scope.messagesList[mensaje.chat].push(mensaje);
      }else{
        $scope.messagesList[mensaje.chat] = new Array();
        $scope.messagesList[mensaje.chat].push(mensaje);
      }
    }
  }
}
```

```

express.js      routes.js      usuarios.js      passport.js      registro.js
1 var logger = require('morgan');
2 var bodyParser = require('body-parser');
3 var cookieParser = require('cookie-parser');
4 var swig = require('swig');
5 var express = require("express");
6 var passport = require('./passport');
7 var session = require("express-session");
8 var redisStore = require('connect-redis')(sess
9
10
11 module.exports = function(app, config){
12     app.engine('html', swig.renderFile);
13     app.set('view engine', 'html');
14     app.set('views', config.rootPath + '/server/views');
15
16     app.set('view cache', false);
17     swig.setDefaults({cache: false, varControls: ['{\^', '}^']}});
18
19     app.use(cookieParser());
20     app.use(logger('dev'));
21     app.use(bodyParser());
22
23
24     app.use(session({store : new redisStore
25     app.use(passport.initialize());
26     app.use(passport.session());
27     app.use(express.static(config.rootPath+
28 });

```

redis-server terminal output:

```

Redis 3.0.0 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 5362
http://redis.io

5362:M 30 Apr 12:42:09.594 # Server started, Redis version 3.0.0
5362:M 30 Apr 12:42:09.594 * DB loaded from disk: 0.000 seconds
5362:M 30 Apr 12:42:09.594 # The server is now ready to accept connections on port 6379

```

Passport.js terminal output:

```

pause@0.0.1
└── passport-strategy@1.0.0
iMac-de-Desarrollo01:Passport i Desarrollo01$ supervisor -e html.js Servidor.js
Running node-supervisor with
program 'Servidor.js'
--watch '.'
--extensions 'html.js'
--exec 'node'

Starting child process with 'node Servidor.js'
Watching directory '/Users/Desarrollo01/Documents/DesarrolloNextUniversity/NodeJS II/Passport' for changes.
body-parser deprecated bodyParsers use individual json/urlencoded middlewares see
server/config/express.js#22:18
body-parser deprecated undefined extended; provide extended option node_modules/
body-parser/index.js:85:29
express-session deprecated undefined resave option; provide resave option server
/config/express.js:24:18
express-session deprecated undefined saveUninitialized option; provide saveUnini
tialized option server/config/express.js#24:18
Servidor Corriendo en el puerto: 3000
Base de Datos Teamapp abierta

```

```

package.json      Servidor.js      index.html
72
73         <i class="fa fa-check"></i><span>Tareas</span>
74     </a>
75 </li>
76
77     <li class="">
78         <a ui-sref="app.recursos">
79             <i class="fa fa-envelope"></i><span>Recursos</span>
80         </a>
81     </ul>
82 </section>
83 </aside>
84 </div>
85
86 <aside class="right-side">
87     <section class="content-header">
88         <h1>Dashboard<small>Control P
89     </section>
90
91     <section class="content">
92
93         </section>
94     </aside>
95
96     <script type="text/javascript" src="/js/app.js"></script>
97     <script type="text/javascript" src="/js/controllers.js"></script>
98     <script type="text/javascript" src="/js/services.js"></script>
99     <script type="text/javascript" src="/vendor/angular/angular.js"></script>
100    <script type="text/javascript" src="/vendor/angular-ui-router/release/angular-ui-router.js"></script>
101 </body>

```

Terminal output for the MEAN application:

```

pause@0.0.1
└── supervisor@0.6.0
iMac-de-Desarrollo01:Modulacion MEAN i Desarrollo01$ supervisor -e html.js Servidor.js
Running node-supervisor with
program 'Servidor.js'
--watch '.'
--extensions 'html.js'
--exec 'node'

Starting child process with 'node Servidor.js'
Watching directory '/Users/Desarrollo01/Documents/DesarrolloNextUniversity/NodeJS II/Modulacion MEAN' for changes.
body-parser deprecated bodyParsers use individual json/urlencoded middlewares see
server/config/express.js#22:18
body-parser deprecated undefined extended; provide extended option node_modules/
body-parser/index.js:85:29
express-session deprecated undefined resave option; provide resave option server
/config/express.js:24:18
express-session deprecated undefined saveUninitialized option; provide saveUnini
tialized option server/config/express.js#24:18
Servidor Corriendo en el puerto: 3000
Base de Datos Teamapp abierta

```

# Knockout JS

- ✓ Biblioteca de JavaScript, que nos permite crear aplicaciones con una interfaz dinámica.
- ✓ Fácil mantenimiento a nuestro aplicativo web.
- ✓ Utiliza enlaces declarativos y comportamientos personalizados.
- ✓ Compatibilidad con cualquier navegador actual.
- ✓ Desarrollo BDD o desarrollo guiado por comportamiento.
- ✓ Proporciona una manera complementaria de alto nivel para vincular un modelo de datos para una interfaz de usuario.
- ✓ No depende de JQuery, pero es compatible con esta famosa librería.
- ✓ Muy útil para aplicativos web con transacciones animadas.

## Observables

- ✓ Son objetos especiales de JavaScript que pueden notificar a los suscriptores acerca de los cambios, y puede detectar automáticamente las dependencias.
- ✓ Cambian de manera dinámica nuestra interfaz de usuario.

## Lectura y escritura de Observables

- ✓ Lectura: Permite capturar o llamar a un observable sin ningún parámetro.
- ✓ Escritura: Consiste en asignarle un valor directo a nuestro observable.

```
this.n = ko.observable(n); → escritura  
Mymodel.personaNombre(); → lectura
```