



# Tema 8.1

## Javascript: Librerías jQuery y Zepto

# Librerías Javascript



- ◆ Las librerías JavaScript actuales son **multi-navegador**
  - Funcionan en IE, Firefox, Safari, Chrome, Opera, ...
    - ◆ Ahorran mucho tiempo -> **utilizarlas siempre que existan**
  - Ejemplos: **jQuery**, **Zepto**, Prototype, Dojo, lungo.js, PhoneGap, ...
- ◆ **jQuery** es muy popular (<http://jquery.com/>)
  - Procesar DOM, gestionar eventos, animaciones y estilos CSS, Ajax, ..
- ◆ **Zepto**: subconjunto compatible de **jQuery** para móviles (<http://zeptojs.com>)
  - **zepto.min.js** ocupa solo **9,7Kbytes** (gzipped)
    - ◆ ¡OJO! Soporta browsers móviles actuales, pero no **Internet Explorer**
  - Añade **eventos táctiles** para móviles y tabletas

# Objetos y función jQuery (o Zepto)

- ◆ **Objeto jQuery**: representación más eficaz de un **objeto DOM**
  - se procesan en bloque (array) con métodos de jQuery como **html(...)**
- ◆ **Función jQuery**: **jQuery("<selector CSS>")** o **\$("<selector CSS>")**
  - devuelve el **array de objetos jQuery** que casan con **<selector CSS>**
    - ◆ **<selector CSS>** selecciona objetos DOM igual que en CSS

```
document.getElementById("fecha").innerHTML = "Hola";  
// es equivalente a:  
$("#fecha").html("Hola");
```

- ◆ La función jQuery convierte además **objetos DOM** y **HTML** a **objetos jQuery**

```
$(objetoDOM); // convierte objetoDOM a objeto jQuery  
$("<ul><li>Uno</li><li>Dos</li></ul>") // convierte HTML a objeto jQuery
```


# Fecha y hora con jQuery/Zepto

- ◆ **Libreria:** script externo identificado por un **URL** que hay que cargar con:
  - `<script type="text/javascript" src="zepto.min.js" > </script>`
- ◆ `$("#fecha")` devuelve el objeto jQuery/Zepto del elemento con id="clock"
- ◆ `$("#fecha").html(new Date())` inserta
  - **new Date()** en objeto jQuery
    - ◆ devuelto por `$("#fecha")`



# Función ready: esperar a página cargada

- ◆ Función ready(): ejecuta un script con el objeto DOM está construido
  - Invocación: `$(document).ready(function() { .. código js ... });`
    - ◆ Suele utilizarse la sintaxis abreviada: `$(function() { .. código .. });`



```
<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript" src="zepto.min.js"></script>

  <script type="text/javascript">
    $(function() { $('#fecha').html(new Date( )); });
  </script>
</head>

<body>
<h2>Fecha y hora (ready):</h2>

<div id="fecha"></div>
</body>
</html>
```

© Juan Quemada, DIT, UPM

# Cache y CDN (Content Distribution Network)

- ◆ Cache: contiene recursos accedidos anteriormente para acceso rapido
  - Caches identifican recursos por igualdad de URLs
- ◆ CDN: utilizar URL común a Google, Microsoft, jQuery, Zepto, ...
  - Para maximizar probabilidad de que recursos estén ya en la cache



The screenshot shows a web browser window titled "29-date\_zepto\_cdn.htm" with a status bar indicating "UNREGISTERED". The browser displays the source code of an HTML document. The code includes a script tag for Zepto.js from a CDN, followed by a script that updates the content of a div with the id "clock" to show the current date and time. The HTML structure includes a head section for the scripts and a body section with a heading and a div for the clock.

```
<html>
<head>
<script type="text/javascript" src="http://zeptojs.com/zepto.min.js">
</script>

<script type="text/javascript">
  $(function() { $('#clock').html(new Date( )); });
</script>
</head>
<body>
<h2>Fecha y hora, con CDN Zepto</h2>

<div id="clock"></div>
</body>
</html>
```

# Selectores tipo CSS de jQuery/Zepto

## SELECTORES DE MARCAS CON ATRIBUTO ID

`$("#h1#d83")` /\* devuelve objeto con marca **h1** e **id="d83"** \*/  
`$("#d83")` /\* devuelve objeto con **id="d83"** \*/

## SELECTORES DE MARCAS CON ATRIBUTO CLASS

`$("#h1.princ")` /\* devuelve array de objetos con marcas **h1** y **class="princ"** \*/  
`$(".princ")` /\* devuelve array de objetos con **class="princ"** \*/

## SELECTORES DE MARCAS CON ATRIBUTOS

`$("#h1[border]")` /\* devuelve array de objetos con marcas **h1** y **atributo border** \*/  
`$("#h1[border=yes]")` /\* devuelve array de objetos con marcas **h1** y **atributo border=yes** \*/

## SELECTORES DE MARCAS

`$("#h1, h2, p")` /\* devuelve array de objetos con marcas **h1, h2 y p** \*/  
`$("#h1 h2")` /\* devuelve array de objetos con marca **h2 después de h1** en el árbol \*/  
`$("#h1 > h2")` /\* devuelve array de objetos con marca **h2 justo después de h1** en árbol \*/  
`$("#h1 + p")` /\* devuelve array de objetos con **marca p adyacente a h1** del mismo nivel \*/

.....

# jQuery/Zepto: Metodos modificadores

## ◆ \$('#id3').html('Hello World!')

- sustituye **HTML** por **'Hello World!'** en el elemento con **id='id3'**
  - ◆ \$('#id3').html() devuelve contenido **HTML** de \$('#id3')
- \$('#id3').append('Hello World!') añade **HTML** al final del existente

## ◆ \$('.expr').val('3')

- inserta atributo **value='3'** a elementos de la clase **'expr'**
  - ◆ \$('#id3').val() devuelve atributo **value** de \$('#id3')

## ◆ \$('#id3').attr('rel', 'license')

- inserta atributo **rel='license'** a elemento con **id=id3**
  - ◆ \$('#id3').attr('rel') devuelve atributo **rel** de \$('#id3')

## ◆ \$('ul').addClass('visible')

- inserta atributo **class='visible'** a elementos **<ul>** (lista)

## ◆ \$('h1').hide() y \$('h1').show()

- oculta o muestra elementos **<h1>**



# 4 Modos de invocar Zepto (o jQuery)

- ◆ **String con selector CSS:** `$("#<selector CSS>")`
  - Devuelve un array de objetos jQuery que casan con **<selector CSS>**
- ◆ **"String HTML":** `$("#<ul><li>Uno</li><li>Dos</li></ul>")`
  - Devuelve objeto jQuery equivalente a string interpretado como **HTML**
- ◆ **"Objeto DOM":** `$(document)`
  - Transforma objeto DOM en objeto jQuery equivalente
- ◆ **"Función ready":** `$(function(){..})`
  - Ejecuta la función cuando el **árbol DOM está construido**

# Ejercicio

- ◆ Si tenemos una página HTML con el siguiente contenido

```
<!DOCTYPE html><html>
<script type="text/javascript" src="zepto.min.js" > </script>
<script type="text/javascript">
  $(function){ ..... /* script con expresiones de abajo */ });
</script>
</head><body>
  <h4 id="id1" >Título</h4>

  <p id="id2">Texto</p>

  <div id="id3"></div>
</body></html>
```

- ◆ Como se evaluarán las siguientes expresiones si estuviesen en el script

<b><code>\$("#id1").html()</code></b>	<b><code>=&gt; undefined, "", "null", "Título", "Texto"</code></b>
<b><code>\$("#id2").html()</code></b>	<b><code>=&gt; undefined, "", "null", "Título", "Texto"</code></b>
<b><code>\$("#id3").html()</code></b>	<b><code>=&gt; undefined, "", "null", "Título", "Texto"</code></b>

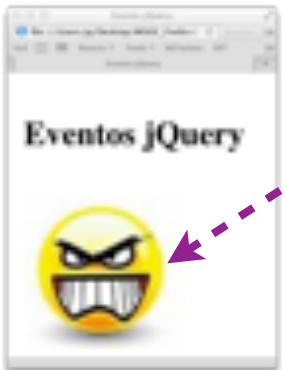
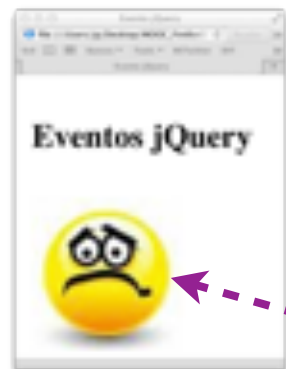


# Tema 8.2

## Javascript: Eventos en jQuery y Zepto

# Eventos con jQuery/Zepto

- ◆ Manejadores de eventos: se definen sobre el objeto jQuery `i` de `<img.. id=i1>`
  - con la función `on` -> `i.on('evento', manejador)`



```
20_event_id_zepto.htm UNREGISTERED
<!DOCTYPE html>
<html><head><title>Evento jQuery</title><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js" > </script>
<script type="text/javascript">
    $(function(){
        var i = $('#i1');

        i.on('dblclick', function(){i.attr('src', 'wait.png')});

        i.on('click', function(){i.attr('src', 'scare.png')});
    });
</script>
</head>
<body>
    <h4>Eventos jQuery</h4>

    
</body>
</html>
```

```

<!DOCTYPE html>
<html><head><title>Pregunta (jQuery)</title><meta charset="UTF-8">

<script type="text/javascript" src="zepto.min.js" > </script>

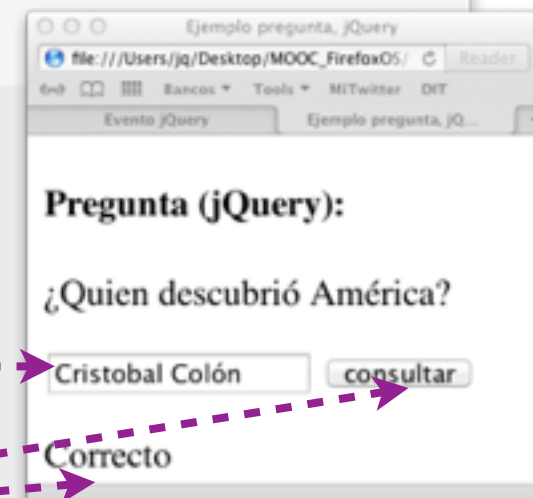
<script type="text/javascript">
$(function(){
    $("#boton").on('click', function(){
        if ($('#respuesta').val() === "Cristobal Colón")
            $('#resultado').html("Correcto");
        else
            $('#resultado').html("No es correcto");
    });
});
</script>
</head>
<body>
<h4> Pregunta (jQuery):</h4>
¿Quien descubrió América? <p>

<input type="text" id="respuesta" value="responda aquí">
<button type="button" id="boton"> consultar </button>

<p><div id="resultado" />
</body>
</html>

```

## Pregunta Zepto



# Ejercicio

- ◆ Si tenemos una página HTML con el siguiente contenido

```
<!DOCTYPE html><html>
<script type="text/javascript" src="zepto.min.js" > </script>
<script type="text/javascript">
    $(function(){ ..... /* script con expresiones de abajo */  });
</script>
</head><body>
    <h4 id="id1" >Título</h4>

    <input type="text" id="id2" name= "caja" value="7">
</body>
</html>
```

- ◆ Como se evaluarán las siguientes expresiones si estuviesen en el script

<code>\$("#id1").html()</code>	<code>=&gt; undefined, "", "caja", "Título", "7", 7</code>
<code>\$("#id1").val()</code>	<code>=&gt; undefined, "", "caja", "Título", "7", 7</code>
<code>\$("#id2").html()</code>	<code>=&gt; undefined, "", "caja", "Título", "7", 7</code>
<code>\$("#id2").val()</code>	<code>=&gt; undefined, "", "caja", "Título", "7", 7</code>

# Ejercicio

- ◆ Modificar el ejemplo anterior para que si la respuesta es incorrecta
  - además de indicar “No es correcto”
    - ♦ inicie el cajetín con el texto “pruebe otra vez”
      - Pero ahora haciendolo con Zepto
- ◆ Información sobre funciones Zepto para modificar atributos HTML:
  - <http://zeptajs.com/>
  - <http://zeptajs.com/#val>
  - <http://zeptajs.com/#attr>



# Tema 8.3

## Javascript: Ejemplo de un cronómetro



# Cronómetro

- ◆ WebApp similar a un cronómetro digital
- ◆ Cuenta décimas de segundo (100 miliseg.)
  - El contador se inicializa con **0,0** segundos
    - ◆ **n.toFixed(1)** formatea con 1 decimal
- ◆ Tiene 2 botones
  - **arrancar/parar**: arranca o para la cuenta
    - ◆ a partir del valor en que quedo
      - arranca si cronómetro parado
      - para si cronómetro contando
  - **inicializar**: pone el contador a 0,0



```

<!DOCTYPE html>
<html>
<head><title>Event Example</title><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js" > </script>
<script type="text/javascript">
    $(function(){
        var t, cl = $("#crono");

        function mostrar() { cl.html((+cl.html() + 0.1).toFixed(1)); };
        function arrancar() { t=setInterval(mostrar, 100);};
        function parar() { clearInterval(t); t=undefined; };
        function cambiar() { if (!t) arrancar(); else parar(); };

        $("#cambiar").on('click', cambiar);
        $("#inicializar").on('click', function(){ cl.html("0.0"); });
    });
</script>
</head>
<body>
<h2>Cronómetro</h2>

<h2><span id="crono"> 0.0 </span> segundos </h2>

<button type="button" id="cambiar"> arrancar/parar </button>
<button type="button" id="inicializar"> inicializar </button>
</body>
</html>

```

## Cronómetro



# DOM como almacén de datos

- ◆ El navegador guarda en **document** la página HTML que está mostrando
  - **document** es un objeto JavaScript con propiedades
    - ◆ que contienen todos los elementos de la página
- ◆ Las propiedades DOM son variables: **src**, **value**, **innerHTML**, ....
  - donde la información se puede guardar y recuperar
    - ◆ DOM solo contiene strings y todo debe convertirse a/de string
- ◆ Los elementos de DOM se pueden utilizar como variables
  - Hemos utilizado el elemento **<span id="crono">**
    - ◆ para almacenar el contador de decimas de segundo

# Ejercicio

- ◆ Modificar el cronómetro para que cuente centésimas de segundo
  - mostrando los segundos con dos decimales
- ◆ Añadir además una lista debajo de los botones
  - con los instantes en que el cronómetro ha parado de contar
    - ◆ La lista deberá vaciarse cuando se inicialice el cronómetro
  - Sugerencia: guardar los instantes como líneas HTML separadas por `<br>` en un bloque genérico debajo de los botones
- ◆ La respuesta al botón de inicializar hay que modificarla para que
  - no inicialice si se pulsa cuando el cronómetro está contando



# Tema 8.4

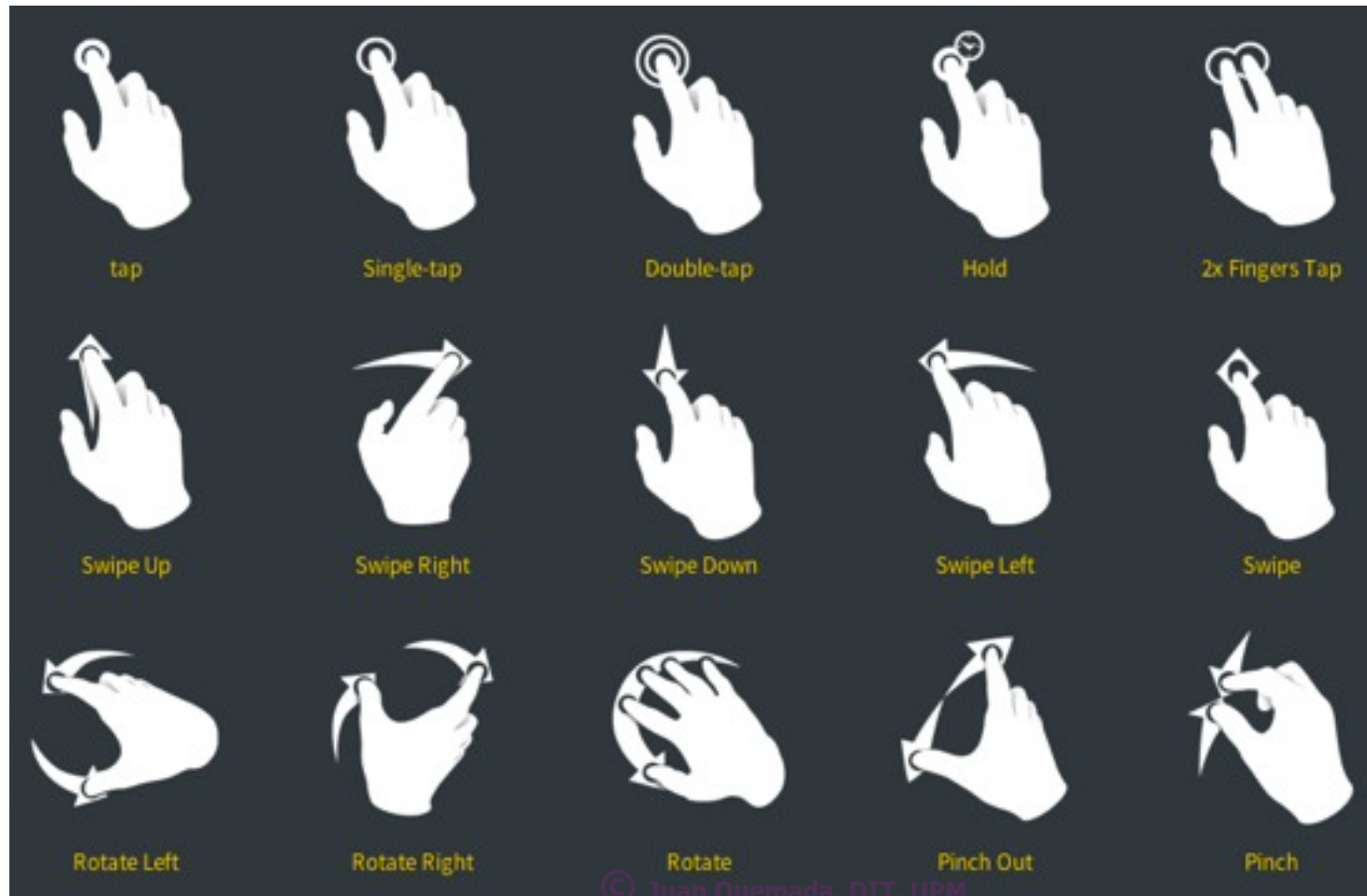
## Javascript: Eventos tactiles

# Eventos táctiles

- ◆ iPhone (2007): dispara el uso de pantallas táctiles
  - Empiezan a incluirse eventos “touch” en navegadores (JavaScript)
- ◆ W3C está normalizando eventos táctiles básicos
  - **touchstart, touchmove, touchend**
    - ◆ [https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Events/Touch\\_events](https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Events/Touch_events)
- ◆ Además están los gestos (gestures)
  - se soportan con librerías JavaScript
    - ◆ TapQUO, Zepto (touch, gesture), jQuery Mobile, Hammer, ...

# Gestos (gestures)

- ◆ La tendencia es utilizar gestos complejos soportados por librerías
  - La figura muestra eventos táctiles de la librería TapQUO

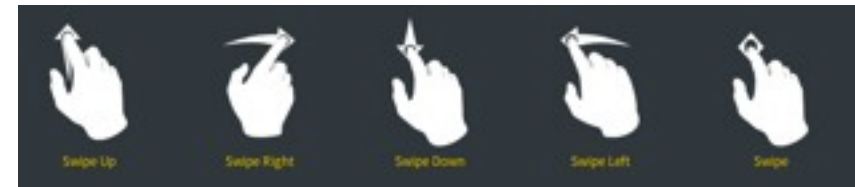


# Eventos táctiles y Zepto

- ◆ El único evento reutilizable en pantallas táctiles es: **click**
  - Suele estar enlazado al evento **tap** y funciona con pantallas táctiles

- ◆ Zepto incluye 2 librerías de gestos táctiles

- touch.js que añade los eventos
  - ◆ tap, singleTap, doubleTap, swipe, swipeUp, swipeDown, swipeLeft, swipeRight
- gesture.js que añade los eventos
  - ◆ pinch, pinchIn, pinchOut



- ◆ Los S.O. de los dispositivos táctiles como iOS o Android
  - Llevan eventos predefinidos asociados a gestos
    - ◆ Por ejemplo, iOS (Apple) predefine **double\_tab** (ampliar) y **pinch** (ampliar)
  - La configuración por defecto se quita incluyendo en el manejador
    - ◆ **evento.preventDefault()**



# Ejemplo: Eventos para pantalla tactil

- ◆ El ejemplo adapta el programa **Eventos** a una pantalla tactil
  - Cambio de icono se hace con: **swipeRight** (click), **swipeLeft** (dblclick)
- ◆ Eventos táctiles nativos multi-toque (multi-touch) de W3C
  - devuelven un array de toques (TouchList)
    - ◆ uno por cada dedo que toque
  - Cada toque genera 3 eventos
    - ◆ **touchstart**: evento disparado al tocar la pantalla
    - ◆ **touchmove**: evento disparado al mover el toque
    - ◆ **touchend**: evento disparado al final del toque
- ◆ Se mide la diferencia de pageX (coordenada X en pantalla)
  - entre **touchstart** y **touchmove** (en manejadores)



```

<!DOCTYPE html>
<html><head><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js" ></script>
<script type="text/javascript">

```

## Evento tactil JavaScript

```

$(function(){
  var i=$('#i1');
  var xIni, yIni;

```

```

i.on('touchstart', function(e){
  xIni = e.targetTouches[0].pageX;
  yIni = e.targetTouches[0].pageY;
});

```

```

i.on('touchmove', function(e){
  if (e.targetTouches[0].pageX > xIni+10) i.attr('src', 'scare.png');
  if (e.targetTouches[0].pageX < xIni-10) i.attr('src', 'wait.png');
});

```

```

</script>

```

```

</head><body>

```

```

<h4>Evento Touch</h4>

```

```



```

```

</body>

```

```

</html>

```



- ◆ La librería **touch.js** de **Zepto** detecta y dispara eventos táctiles automáticamente
  - Si cargamos la librería podemos definir directamente manejadores de
    - ◆ **swipeRight** y **swipeLeft** sobre el icono

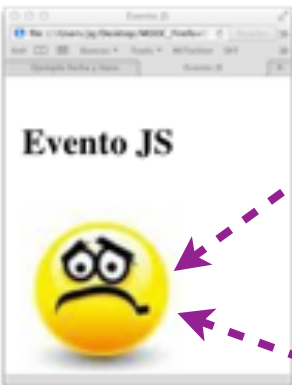
## Eventos touch.js

```
<!DOCTYPE html>
<html>
<head><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js" ></script>
<script type="text/javascript" src="touch.js" ></script>
<script type="text/javascript">
  $(function(){
    var i = $('#i1');

    i.on('swipeRight', function(){ i.attr('src', 'scare.png'); });

    i.on('swipeLeft', function() { i.attr('src', 'wait.png'); });
  });
</script>
</head>
<body>
  <h4>Evento Touch</h4>

  
</body>
</html>
```



# Ejercicio

- ◆ Modificar el ejemplo del cronómetro
  - quitar los 2 botones: **arrancar/parar** e **inicializar**
- ◆ Añadir los siguiente eventos tactiles sobre el body del cronómetro
  - Evento **tap**: sustituir a **arrancar/parar**
  - Evento **swipe**: sustituir a **inicializar**
- ◆ Utilizar la libreria **touch.js** para implementar los eventos tactiles



# Tema 8.5

## Javascript: Memoria local

# Almacenamiento de datos en cliente

- ◆ HTML5 implementa nuevos tipos de almacenamiento de variables
  - Sencillas y eficientes de utilizar desde Javascript
    - ◆ Definición: <http://dev.w3.org/html5/webstorage/>
- ◆ **Variables locales**
  - los datos se guardan permanentemente, hasta que se borran
- ◆ **Variables de sesión**
  - Los datos solo se guardan solo **durante la sesión**
    - ◆ **Comienzo de sesión:** apertura de navegador o pestaña
    - ◆ **Final de sesión:** cierre de navegador o pestaña

# Variables locales y de sesión

- ◆ Son **propiedades** de los **objetos localStorage y sessionStorage**
  - solo pueden contener **strings**, como por ejemplo
    - ◆ **localStorage.usuario = “Pedro Pérez”;**
    - ◆ **sessionStorage.apellido = “Pérez”;**
- ◆ Las variables locales están asociadas a **protocolo, dominio y puerto**
  - un programa solo puede acceder a propiedades de local/sessionStorage
    - ◆ creadas por otros programas cargados del mismo servidor
- ◆ **Same origin policy**
  - **Seguridad:** un programa solo confía en programas del mismo servidor
  - **Modularidad:** cada servidor tiene un espacio de nombres diferente

# Ejemplo de localStorage

- ◆ Cada usuario que acceda a esta página tendrá una cuenta diferente
  - La variable está en su navegador

```
65-visitCount.html UNREGISTERED
<!DOCTYPE html>
<html><head><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js"></script>
<script type="text/javascript">
  $(function() {
    // si variable no existe se crea (primera visita)
    localStorage.cuenta = (localStorage.cuenta || 0);

    localStorage.cuenta++; // incrementamos cuenta de visitas

    $('#cuenta').html(localStorage.cuenta);
  });
</script>
</head><body>
  <h3>Ejemplo de localStorage</h3>

  Ha visitado esta página <span id='cuenta'></span> veces!
</body>
</html>
```





# Cronómetro con memoria

- ◆ Nueva versión del cronómetro con **localStorage**
  - así mantiene la cuenta de décimas de segundos
    - ◆ entre usos sucesivos de la aplicación
- ◆ El cronómetro utiliza ahora la variable
  - **localStorage.c**
    - ◆ para guardar la cuenta de segundos
- ◆ Debemos inicializar localStorage.c
  - con parámetro por defecto para cuando se ejecute por primera vez
- ◆ Como la información se guarda ahora en localStorage y no en DOM
  - hay que actualizar primero localStorage y luego mostrar en DOM



# Cronómetro: localStorage

```
<!DOCTYPE html>
<html>
<head><title>Cronómetro</title><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js" > </script>
<script type="text/javascript">
  $(function(){
    localStorage.c = (localStorage.c || "0.0");

    var t, cl = $("#crono");

    function incr() { localStorage.c = +localStorage.c + 0.1; }
    function mostrar() { cl.html((+localStorage.c).toFixed(1)); };
    function arrancar() { t=setInterval(function(){incr(); mostrar()}, 100);};
    function parar() { clearInterval(t); t=undefined; };
    function cambiar() { if (!t) arrancar(); else parar(); };

    $("#cambiar").on('click', cambiar);
    $("#inicializar").on('click', function(){ localStorage.c="0.0"; mostrar();});
    mostrar();
  });
</script>
</head>
<body>
<h2>Cronómetro</h2>

<h3><span id="crono"> 0.0 </span> segundos </h3>

<button type="button" id="cambiar"> arrancar/parar </button>
<button type="button" id="inicializar"> inicializar </button>
</body>
</html>
```



# Ejercicio

- ◆ Modificar el cronómetro con memoria para que
  - cuente en centésimas de segundo
- ◆ Añadir una lista debajo de los botones
  - con los instantes en que ha parado de contar
    - ♦ la lista se deberá guardar en una variable de localStorage
      - para que los valores se guarden entre invocaciones sucesivas
    - ♦ La lista deberá vaciarse con el botón de inicializar del cronómetro
  - Sugerencia: añadir variable localStorage que guarde los instantes como líneas HTML separadas por <br>
- ◆ La respuesta al botón de inicializar hay que modificarla para que
  - no inicialice si se pulsa cuando el cronómetro está contando
- ◆ Añadir los eventos táctiles **tap** y **swipe** sobre el body del cronómetro
  - para **parar/arrancar** e **inicializar** respectivamente



# Tema 8.6

## Javascript: Notepad

# Ejemplo Notepad

- ◆ Notepad es una aplicación Web
  - que crea una agenda donde guardar notas de texto
    - ◆ que se almacenan en localStorage
- ◆ La página Web que la implementa
  - tiene en realidad 2 paginas en una
    - ◆ La página que visualiza las notas
    - ◆ La página que permite editar las notas
- ◆ Al pulsar los botones (Editar o Salvar)
  - Una página se activa con el método **show()** de Zepto
    - ◆ cuando la otra se desactiva con el método **hide()**



```

<!DOCTYPE html> <html>
<head><title>My Notepad</title><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js"></script>
<script >
function salvar(){ // guarda contenido de textarea y lo muestra
    localStorage.texto = $("#textarea").val();
    visor();
}
function visor(){ // ver contenido en bloque <pre>
    $("#texto").html(localStorage.texto);
    $(".editor").hide();
    $(".visor").show();
}
function editor(){ // ver textarea con localStorage.texto
    $("#textarea").val(localStorage.texto);
    $(".editor").show();
    $(".visor").hide();
}

$(function(){
    $("#salvar").on('click', salvar);
    $("#editar").on('click', editor);

    visor();
});
</script>
</head><body>
<h2>Mi Bloc de Notas</h2>

<input type=button value="Edita" id="editar" class="visor">
<pre id="texto" class="visor"> </pre>

<input type=button value="Salva" id="salvar" class="editor"><p>
<textarea rows=10 cols=20 id="textarea" class="editor" ></textarea>
</body>
</html>

```

# NotePad



# Ejercicio almacenamiento

- ◆ Añadir a NotePad utilizando jQuery
  - El editor WYSIWYG Cleditor
    - ◆ Codificado en javascript/jQuery
  - <http://premiumsoftware.net/cleditor/>

