



Ejemplo Javascript: 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 ****
 - ◆ para almacenar el contador de decimas de segundo



Eventos táctiles en JavaScript

Eventos táctiles



- ◆ iPhone (2007): dispara el uso de pantallas táctiles
 - Empiezan a incluirse eventos “touch” en navegadores (JavaScript)
- ◆ W3C está normalizando eventos táctiles básicos o toques
 - **touchstart, touchmove, touchend**
 - ◆ https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Events/Touch_events
- ◆ Los tipos de toque en pantalla se denominan gestos o gestures
 - Se están generando librerías de eventos, que incluyen
 - por ejemplo, toque (tap), desplazamiento (swipe), pellizco (pinch), ..

Gestos (gestures)

- ◆ La tendencia es utilizar gestos complejos soportados por librerías
 - TapQUO, Zepto (touch, gesture), jQuery Mobile, Hammer, ...
 - ◆ La figura muestra eventos táctiles de la librería TapQUO



Eventos básicos para pantalla táctil (W3C)

- ◆ Eventos táctiles básicos devuelven un array de toques (TouchList)
 - Un elemento por cada dedo que toque, generando 3 eventos
 - ◆ **touchstart**: evento disparado al tocar la pantalla
 - ◆ **touchmove**: evento disparado al finalizar el movimiento del toque
 - ◆ **touchend**: evento disparado al acabar del toque
 - Cada elemento de TouchList lleva las coordenadas y otros datos del toque
- ◆ El ejemplo de la página siguiente modifica los controladores de eventos
 - clic sencillo y doble de raton por los eventos swipe-right y swipe-left
 - ◆ Los gestos swipe-right y swipe-left se detectan midiendo
 - la diferencia de la coordenada X entre **touchstart** y **touchmove**




```

<!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;
        });

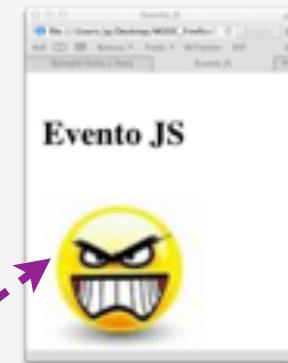
```

```

        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>

```

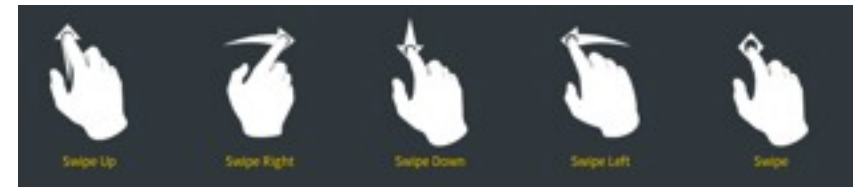


Eventos táctiles y Zepto

- ◆ 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()**

- ◆ 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

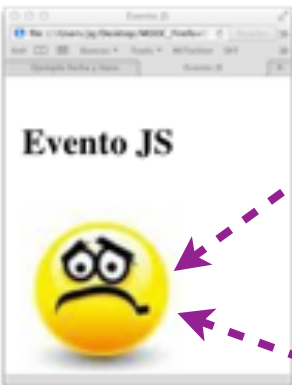
Eventos touch.js

```
<!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>
```





Memoria local en HTML5

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>

```





Final del tema
Muchas gracias!

