Librerías JavaScript jQuery y Zepto

Librerias Javascript

- Las librerias 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 tactiles** para móviles y tabletas
  - Es equivalente a **jQuery 2.0** aparecida recientemente

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>**
    - Si no casa ninguno, devuelve null o undefined
  - **<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")

27-date-jquery.htm — UNREGISTERED

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

<body>
<h2>La fecha y hora con Zepto:</h2>

<div id="fecha"></div>

<script type="text/javascript">
  $('#fecha').html(new Date( ));
</script>
</body>
</html>
```

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 .. });**

28-date_jquery_ready.htm — UNREGISTERED

```
<!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>
```

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

29-date_zepto_cdn.htm — UNREGISTERED

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

SELECTORES DE MARCAS CON ATRIBUTO CLASS
$("h1.princ")    /* devuelve array de objectos con marcas h1 y class="princ" */
$(".princ")      /* devuelve array de objectos con class="princ" */

SELECTORES DE MARCAS CON ATRIBUTOS
$("h1[border]")        /* devuelve array de objectos con marcas h1 y atributo border */
$("h1[border=yes]")    /* devuelve array de objectos con marcas h1 y atributo border=yes */

SELECTORES DE MARCAS
$("h1, h2, p")   /* devuelve array de objectos con marcas h1, h2 y p */
$("h1 h2")       /* devuelve array de objectos con marca h2 después de h1 en el árbol */
$("h1 > h2")     /* devuelve array de objectos con marca h2 justo después de h1 en arbol */
$("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>**

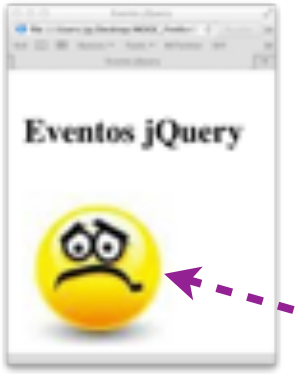
4 Modos de invocar Zepto (o jQuery)

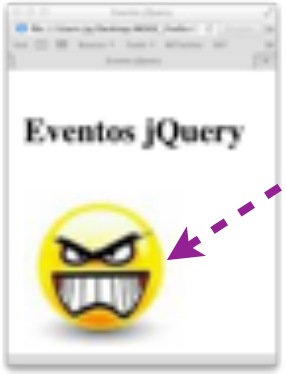
- **String con selector CSS**: "$("**<selector CSS>**")"
  - Devuelve un array de objetos jQuery que casan con "**<selector CSS>**"
- **"String HTML"**: "$("**
Uno
Dos
**")"
  - Devuelve objeto jQuery equivalente a string interpretado como **HTML**
- **"Objeto DOM"**: "$(document)"
  - Transforma objeto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

Eventos con jQuery y Zepto

Eventos con jQuery/Zepto

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



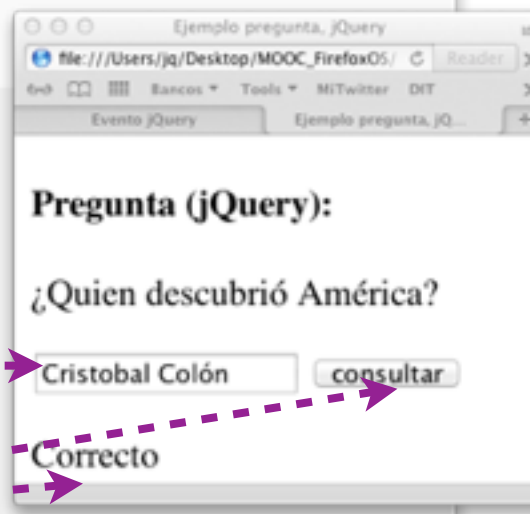


20_event_id_zepto.htm — UNREGISTERED

```
<!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>

  <img src="wait.png" id="i1">
</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



La Clase Array

Arrays

- **Array:** lista ordenada de
  - elementos **heterogéneos**
    - accesibles a través de un índice
      - de **0** a **length-1**
- **Tamaño máximo:** 2^32-2 elementos
- **Elementos**
  - **a[0]** es el primer elemento
  - **.....**
  - **a[a.length-1]** último elemento

```
var x = [1, 2, 3];


a[0]          => 1
a[1]          => 2
a[2]          => 3


a.length      => 3
```

Elementos de un Array

- Elementos del array pueden contener cualquier valor u objeto
  - **undefined**
  - otro **array**
  - **objetos**
  - ...
- Indexar elementos que no existen
  - devuelve **undefined**
    - por ejemplo con índices mayores que **length**

```
var a = [1, undefined, 'a', , [1, 2]];

a[3];          => undefined
a[4];          => [1, 2]

a[9];          => undefined

a[4][1];       => 2
```

Tamaño del Array

- Los arrays son dinámicos
  - pueden crecer y encoger
- Asignar un elemento fuera de rango
  - incrementa el tamaño del array
- El tamaño del array se puede modificar
  - con la propiedad **a.length**
    - **a.length = 3;**
      - modifica el tamaño del array
        - Que pasa a ser 4

```
var a = [1, 3, 1];

a;                    => [1, 3, 1]

a[4] = 2;             => 2
a;                    => [1, 3, 1, , 2]

    // el array se reduce
a.length = 2      =>  2

a                     =>  [1, 3]
```

Métodos de Array

Array hereda métodos de su clase

- **sort()**: devuelve array ordenado
- **reverse()**: devuelve array invertido
- **push(e1, .., en)**
  - añade **e1, ..,en** al final del array
- **pop()**
  - extrae último elemento del array

```
var a = [1, 5, 3];

a.sort()          =>  [1, 3, 5]

a.reverse()       =>  [5, 3, 1]

a.push(false)     =>  4
a                 =>  [5, 3, 1, false]

a.pop()           =>  false
a                 =>  [5, 3, 1]
```

Más métodos

- **join(<separador>):**
  - devuelve string uniendo elementos
    - introduce <separador> entre elementos
- **slice(i,j):** devuelve una rodaja
  - Indice negativo (j) es relativo al final
    - índice “-1” es igual a a.length-2
- **splice(i, j, e1, e2, .., en)**
  - sustituye j elementos desde i en array
    - por e1, e2, ..,en
  - Devuelve elementos eliminados

```
var a = [1, 5, 3, 7];

a.join(';')            =>  '1;5;3;7'

a.slice(1, -1)         =>  [5, 3]

a.splice(1,2,true)  =>  [5, 3]
a                      => [1, true, 7]
```

Bucles: sentencias while, for y do/ while

Bucle

- Un bucle es una **secuencia o bloque de instrucciones**
  - que **se repite** mientras se cumple una **condición** de permanencia
- Un bucle se controla con 3 elementos,
  - normalmente asociados a una variable(s) de control del bucle
    - **Inicialización**: fija los valores de arranque del bucle
    - **Permanencia en bucle**: indica si se debe volver a ejecutar el bloque
    - **Acciones de final bloque**: actualiza en cada repetición las variables de control
- Ilustraremos los bucles (**while**, **for** y **do/while**) con la **función veces**
- **veces(..)** acepta **letra** y **frase** como primer y segundo parámetros
  - Y devuelve el número de veces que la frase contiene la letra


```html
<!DOCTYPE html><html>
<head><meta charset="UTF-8">
</head>
<body>
<h3>Función veces con while</h3>

<script type="text/javascript">

function veces (letra, frase) {
  var i = 0, n = 0;              // inicialización del bucle
  while ( i < frase.length ) {// condición de permanencia
    if ( letra === frase[i]) { ++n; }; //acción del bucle
    ++i;                         // acción de final del bucle
  }
  return n;
};

var l='a', f='la casa roja';
document.write('La frase "' + f + '" tiene '
              + veces(l,f) + ' veces la letra ' + l);
</script>
</body>
</html>
```

veces(..): bucle while


```
<!DOCTYPE html><html>
<head><meta charset="UTF-8">
</head>
<body>
<h3>Función veces con for</h3>

<script type="text/javascript">

function veces (letra, frase) {
  var i, n;  // inicialización, condición y final
  for ( i=0, n=0; i < frase.length; ++i ) {
    if ( letra === frase[i]) { ++n; };    // acción
  }
  return n;
};

var l='a', f='la casa roja';
document.write('La frase "' + f + '" tiene '
             + veces(l,f) + ' veces la letra ' + l);
</script>
</body>
</html>
```

veces(..): bucle for

Sintaxis de la sentencia for

```
39-for_bloque.js                                    UNREGISTERED
// Bloque de acciones del bucle:
//   -> se delimita con {...}

for (i=0; i < arguments.length; ++i) {
    x += " " + arguments[i];
}

// El bloque tiene solo una sentencia
// y los parentesis se pueden omitir,
// pero es menos legible

for (i=0; i < arguments.length; ++i)
    x += " " + arguments[i];
```

- La sentencia comienza conr **for**
- sigue la condición (con 3 partes)
  - **(i=0; i < arguments[i]; i++)**
    - **Inicialización**: i=0, ...
    - **Permanencia en bucle**: i < arguments.length
    - **Acción final bloque**: ++i, ...
- La sentencia termina con un bloque que debe delimitarse con {....}
- Bloques que contengan solo 1 sentencia
  - pueden omitir {..}, pero se mejora la legibilidad delimitandolos con {..}


```
<!DOCTYPE html><html>
<head><meta charset="UTF-8">
</head>
<body>
<h3>Función veces con do</h3>

<script type="text/javascript">

function veces (letra, frase) {
  var i = 0, n = 0;              // inicialización del bucle
  do {
    if ( letra === frase[i]) { ++n; }; // acción del bucle
    ++i;                         //acción de final del bucle
  } while ( i < frase.length )  // condición de permanencia
  return n;
};

var l='a', f='la casa roja';
document.write('La frase "' + f + '" tiene '
            + veces(l,f) + ' veces la letra ' + l);
</script>
</body>
</html>
```

veces(..): bucle do

Ejemplos con iframe, array y for

iFrame

- **iFrame**
  - marco de navegación independiente
- Un iFrame crea una caja de arena segura
  - donde poder importar objetos externos
- Ejemplo: enlaza un juego en otro servidor
  - El iFrame evita que se introduzca malvare
    - Acceso JavaScript limitado a caja de arena

e64-iFrame_Pixie.htm

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" >
    <title> Ejemplo iFrame</title>
  </head>
  <body>
    <h1>Canvas en iFrame</h1>

    iFrame con animación en canvas de
    <a href="http://www.chromeexperiments.com/mobile/">
    Google Chrome Mobile Experiments</a> <p><p>

    <iframe src="http://www.goodboydigital.com/runpixierun/"
            width="600" height="400"> </iframe>
  </body>
</html>
```

Seguridad Web: "Same Origin Policy"

- La seguridad se controla en las aplicaciones JavaScript
  - Permitiendo que un **programa JavaScript** en un **iframe solo acceda**
    - Al **arbol DOM** de la **página principal** si **proviene del mismo origen**
  - Esto **evita** en el ejemplo anterior que el juego
    - **robe o modifique información o datos** del usuario en la página externa

- Origen
  - **protocolo**, **servidor** y **puerto** del URL

- La restricción de pertenecer al "mismo origen"
  - Solo afecta al recurso principal: página Web, recurso, …
    - Los scripts o los estilos no están afectados y pueden venir de otro servidor

- Así es posible hacer "**mash-ups**" seguros
  - de contenidos que no esten en nuestra cadena de confianza

Ejemplo con iFrames

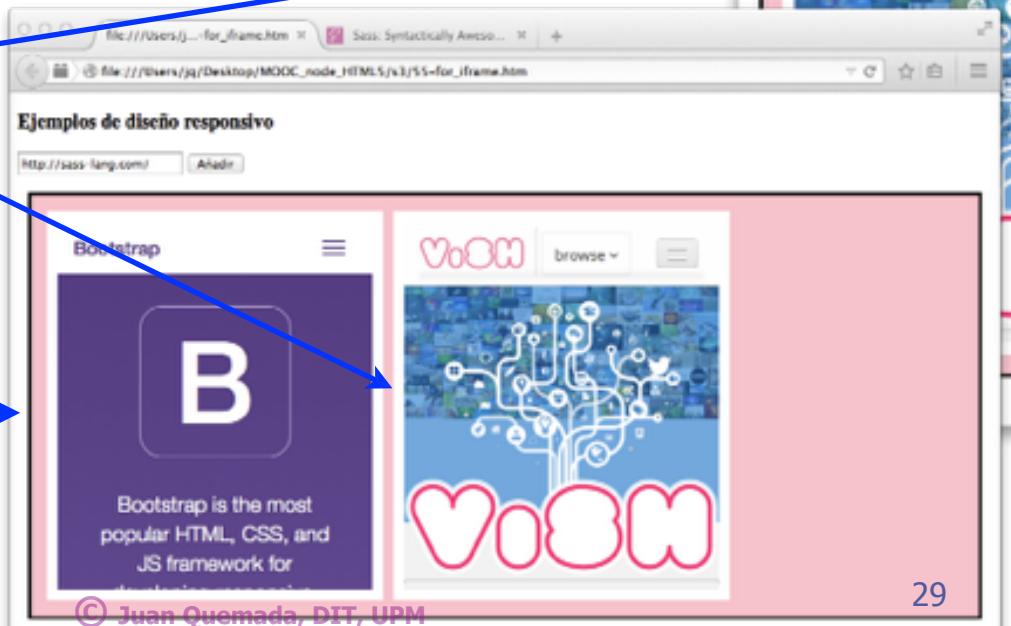
Flexbox

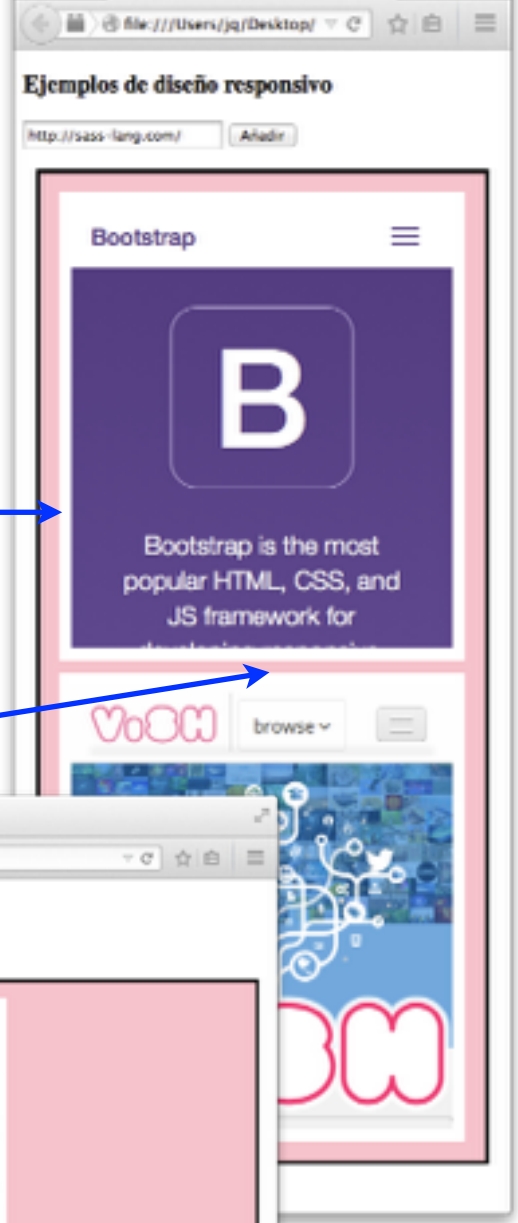
```
<style>

iframe {
  padding:10px;
  border:5px solid pink;
  background-color:white;
  width:300px;
  height:350px;
}

#iframes {
    display: -webkit-box;
    display: -moz-box;
    display: -ms-flexbox;
    display: -webkit-flex;
    display: flex;

    -webkit-flex-wrap: wrap;
    flex-wrap:wrap;
}

#marco {
    background-color: pink;
    padding: 10px;
    margin: 10px;
    border: 3px solid black;
}
</style>
```

Flexbox permite un diseño responsivo fácil y flexible:

display: flex;
flex-wrap: wrap;

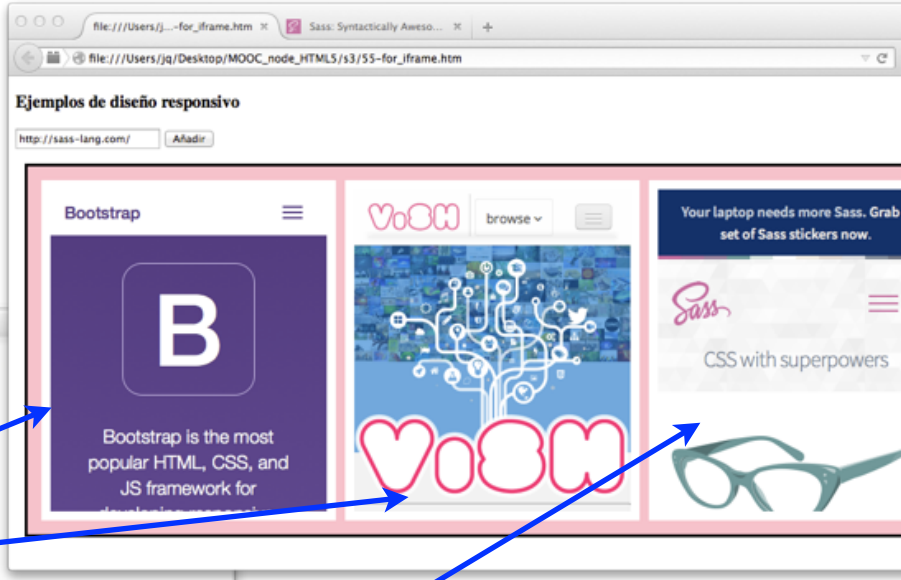
coloca cada iframe a la derecha del anterior. Al llegar al limite pasa a la línea siguiente.





Código

```
<script type="text/javascript" src="zepto.min.js" >
</script>
<script type="text/javascript">
$(function(){
  var urls = ["http://getbootstrap.com",
              "http://vishub.org"];

  function mostrar(urls) {
    var i, iframes="";
    for (i=0;  i < urls.length;  ++i){
      iframes += "<iframe src='" + urls[i] + "'></iframe>";
    }
    $('#iframes').html(iframes);
  };

  $("#boton").on('click', function(){
    urls.push($('#nuevo').val());
    mostrar(urls);
  });

  mostrar(urls);
});
</script>
```




```
<body>
<h3>Ejemplos de diseño responsivo</h3>

<input type="text" id="nuevo" value="Nuevo URL" />
<button type="button" id="boton"> Añadir </button>
<p>
<div  id='marco'><div id="iframes" /></div>

</body>
```