JavaScript en el navegador

Juan Quemada, DIT - UPM

Índice: JavaScript de cliente

TEMARIO BÁSICO

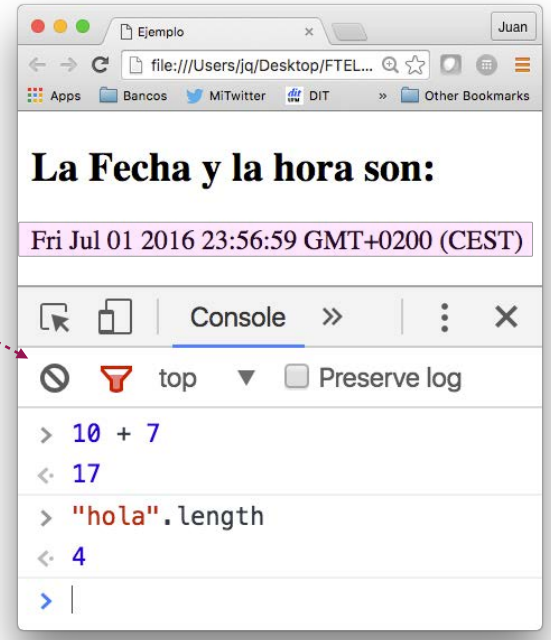
JavaScript en el navegador: objetos window, document, ...

Juan Quemada, DIT - UPM

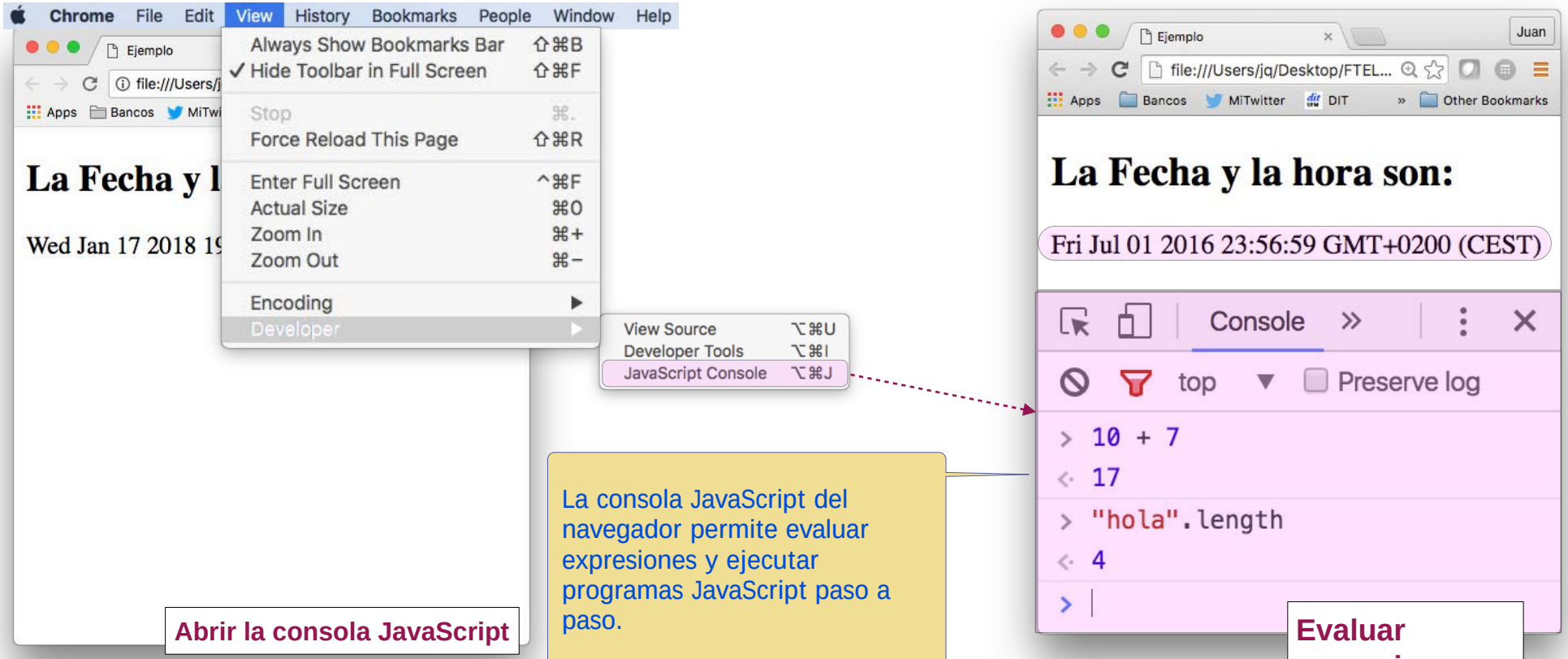
JavaScript en el Navegador

- JavaScript se extiende al navegador con
  - Objetos de interacción con el entorno y el usuario
    - **window**
      - objeto de acceso a los elementos del navegador
    - **DOM** (Document Object Model)
      - objeto **document** de acceso a elementos la página HTML
- Los navegadores ejecutan JavaScript de 2 formas
  - Ejecutan **scripts** JavaScript en una página HTML
  - Ejecutan sentencias paso a paso en la **consola**
- La adaptación de ES6 a los navegadores es lenta
  - Todos los navegadores existentes deben adaptarse
    - Ver tabla de soporte: https://kangax.github.io/compat-table/es6/
  - Existen librerías que traducen ES6 para el browser
    - Babel: https://babeljs.io
    - Traceur: https://github.com/google/traceur-compiler
    - etc.

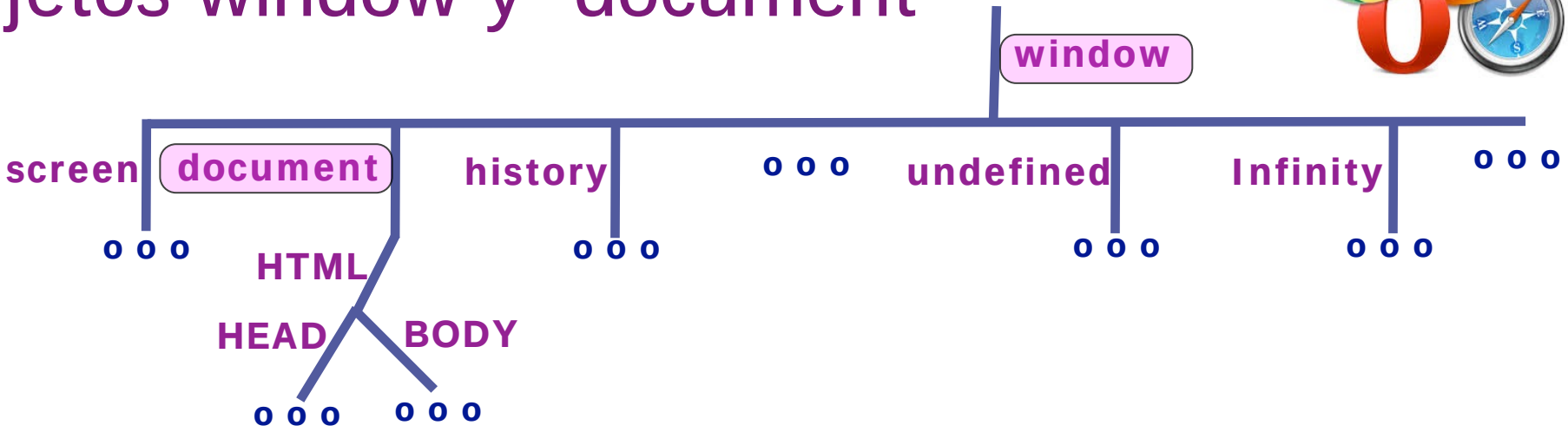
```
<!DOCTYPE html><html><head>
  <title>Ejemplo</title>
  <meta charset="UTF-8">
</head>

<body>
  <h2> La Fecha y la hora son:</h2>

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

  <script type="text/javascript">

    document.getElementById("fecha")
    .innerHTML = new Date();
  </script>
</body>
</html>
```



La consola JavaScript del navegador (Ej. Chrome)



Objetos window y document



- El objeto **window** referencia el entorno de ejecución de JavaScript
  - Es un objeto global cuyas propiedades dan acceso:
    - a nombres y objetos predefinidos de JavaScript, a los elementos del navegador, al documento HTML, ...
      - https://developer.mozilla.org/es/docs/Web/API/Window
  - Una propiedad de **window** se referencia como **window.<prop>**, **this.<prop>** o **<prop>**
- **document**
  - La propiedad **document** da acceso a la interfaz DOM de la página HTML
    - DOM (Document Object Model) es la API que permite acceder a los elementos de HTML desde
      - https://developer.mozilla.org/es/docs/DOM
  - El objeto **document** se referencia como **document**, **window.document** o **this.document**

document y sus métodos de acceso

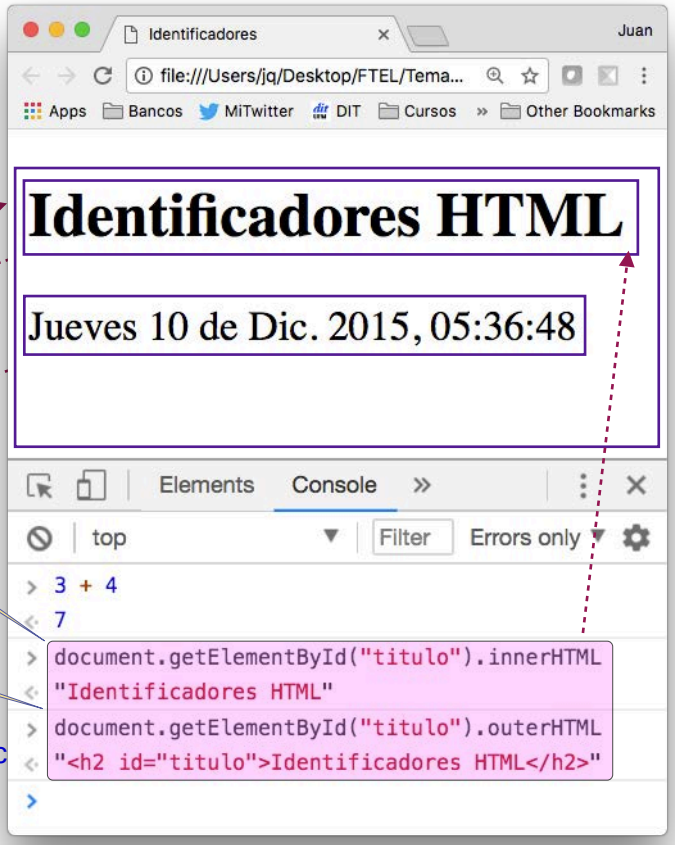
- **document**
  - Hereda métodos y propiedades para interacción con los elementos HTML de la página cargada
    - https://developer.mozilla.org/en/docs/Web/API/Document
- **getElementById("<id>")**
  - Devuelve el objeto DOM con el identificador buscado o null si no lo encuentra
    - Un identificador solo puede estar en un objeto de una página HTML
      - https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById
- **querySelectorAll("<CSS_selector>")**
  - Devuelven un array con los objetos DOM que casan con <CSS_selector>, por ejemplo
    - querySelectorAll(**"#id1"**) array con el objeto DOM del elemento con **id="id1"**
    - querySelectorAll(**".cl1"**) array con objetos de todos los elementos con **class="cl1"**
    - querySelectorAll(**"h1.cl1"**) array con objetos de todos los elementos **<h1 class="cl1">**
      - https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelectorAll

Objetos DOM (Document Object Model)

- Los **elementos HTML** tienen una **caja visual** asociada en el navegador
  - Las cajas son similares a las utilizadas en CSS
- **Objeto DOM**
  - Da acceso a los elementos de HTML cargados en el navegador desde JavaScript
    - El **atributo id="…"** permite obtener el objeto DOM del elemento HTML asociado
      - con el método: **getElementById("identificador")**
- Propiedad **innerHTML** de DOM
  - Contiene el HTML entre las marcas
- Propiedad **outerHTML** de DOM
  - Contiene el HTML completo del elemento

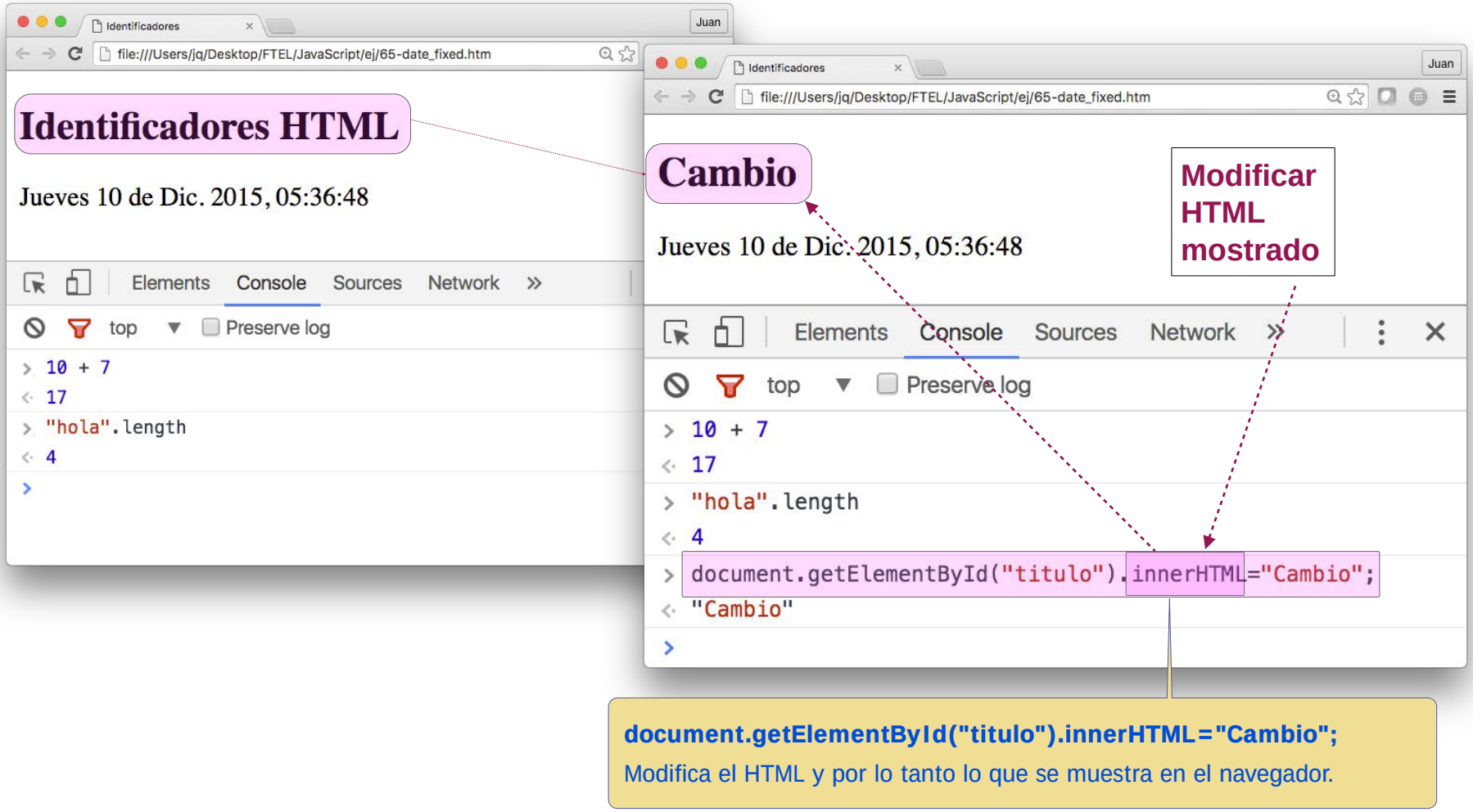
```
<!DOCTYPE html>
<html>
<head>
  <title>Identificadores</title>
  <meta charset="UTF-8">
</head>
<body id="cuerpo">

  <h2 id="titulo">Identificadores HTML</h2>

  <div id="fecha">Jueves 10 de Dic. 2015, 05:36:48</div>
</body>
</html>
```

Elemento HTML **<h2 id="titulo">.... **
Objeto DOM asociado con: **document.getElementById("titulo")**

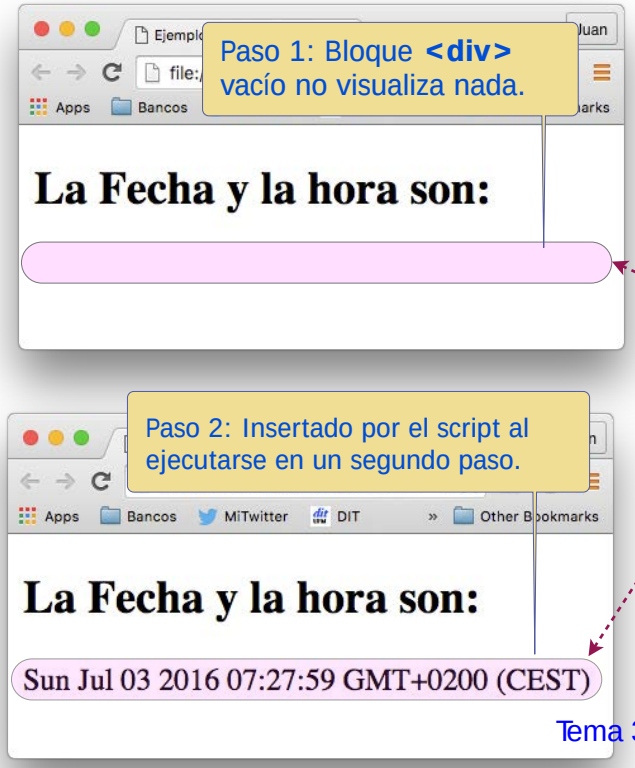


cionamiento de proc

Modificar la página HTML cargada desde la consola JavaScript



Scripts: ejemplo fecha y hora

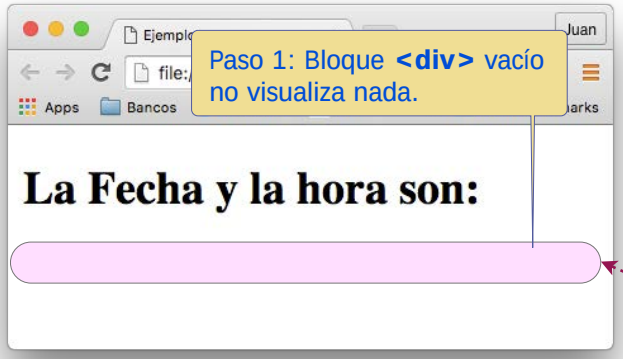


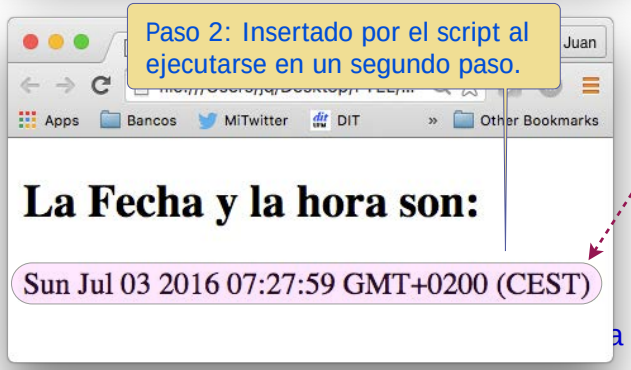

```
<!DOCTYPE html><html>
<head><title>Ejemplo</title><meta charset="UTF-8"></head>
<body>

  <h2> La Fecha y la hora son:</h2>

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

  <script type="text/javascript">
    document.getElementById("fecha").innerHTML = new Date( );
  </script>
</body>
</html>
```

** ** define un bloque HTML genérico y vacío, identificado por "**fecha".**

Scripts: ejemplo fecha y hora

- Un **script** es un **elemento HTML** delimitado por la **marca <script>** que contiene un programa
  - **<script type="text/javascript">** ….. programa JavaScript ….. **</script>**
- **document.getElementById("fecha").innerHTML = new Date();**
  - muestra en el bloque vacío ** ** la fecha y la hora obtenida con **new Date()**
- El navegador ejecuta el **script** de la página Web al cargarla
  - Al visualizar la página HTML no muestra nada en el elemento ** ** (está vacío)
    - El script JavaScript **inserta la hora y la fecha al ejecutarse** (en un segundo paso)






```
<!DOCTYPE html><html>
<head><title>Ejemplo</title><meta charset="UTF-8"></head>
<body>

  <h2> La Fecha y la hora son:</h2>

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

  <script type="text/javascript">
    document.getElementById("fecha").innerHTML = new Date( );
  </script>
</body>
</html>
```

** ** define un bloque HTML genérico y vacío, identificado por "**fecha".**

document: objeto del entorno de ejecución de JavaScript que da acceso a la página Web cargada en el navegador

getElementById("fecha"): Método del objeto document que retorna el objeto DOM del elemento HTML con id="fecha".

innerHTML: Propiedad de un objeto DOM que contiene el HTML interno asociado a ese objeto.

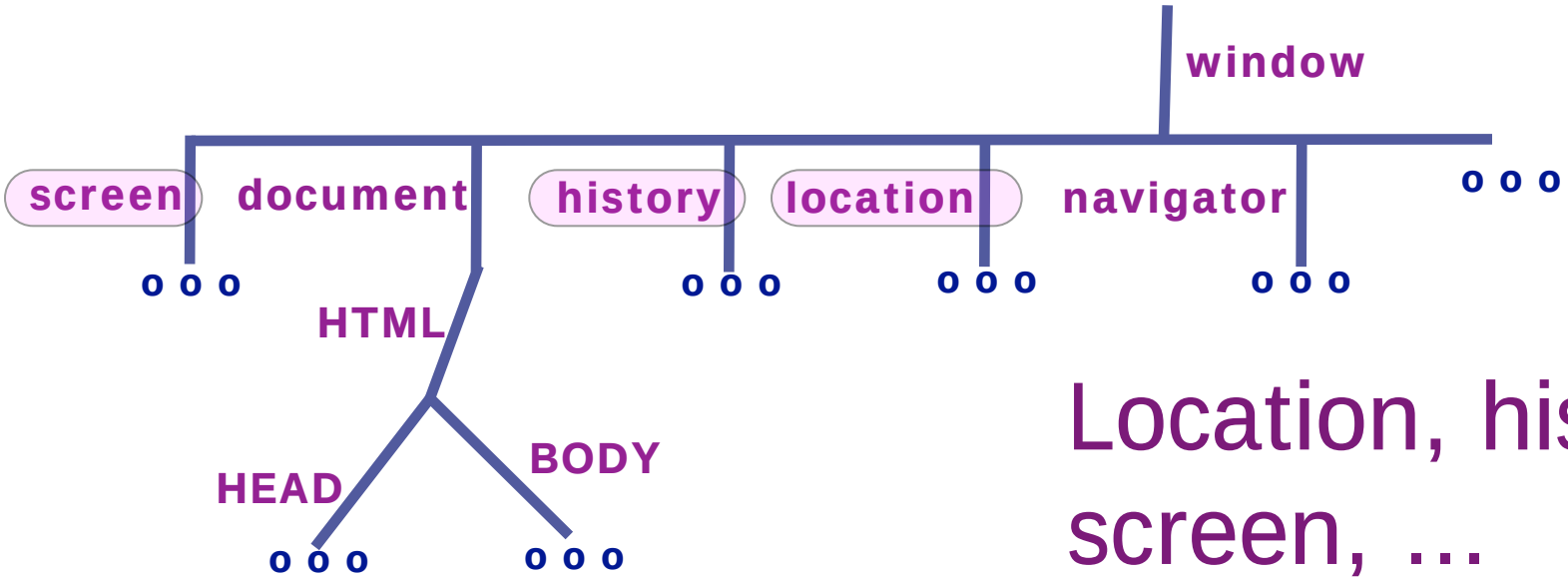
Varios scripts

- **Varios scripts** en una página forman un **único programa JavaScript**
  - Las definiciones (variables, funciones, …) son visibles entre scripts
- Los **scripts se ejecutan** siguiendo el **orden de definición** en la página
  - **Instrucciones adicionales** ejecutadas en la consola del navegador, **se ejecutan después del último script**
- Este ejemplo también es equivalente al anterior
  - Define la **función que inserta fecha** y hora en un script en la cabecera y **la invoca en el script del final**
    - La invocación debe realizarse al final, para que el árbol DOM esté ya construido


```
<!DOCTYPE html><html>
<head>
<title>Ejemplo de función</title>
<meta charset="UTF-8">

<script type="text/javascript">

function mostrar_fecha( ) {
  var cl = document.getElemtById("fecha");
  cl.innerHTML = new Date( );
}
</script>

</head>

<body>
<h2>La fecha y la hora son:</h2>

<div id="fecha"><div>

<script type="text/javascript">
  mostrar_fecha( ); // Invocar función
</script>

</body>
</html>
```

JavaScript en el navegador: objetos location, history, screen, ...

Juan Quemada, DIT - UPM



Location, history, screen, ...

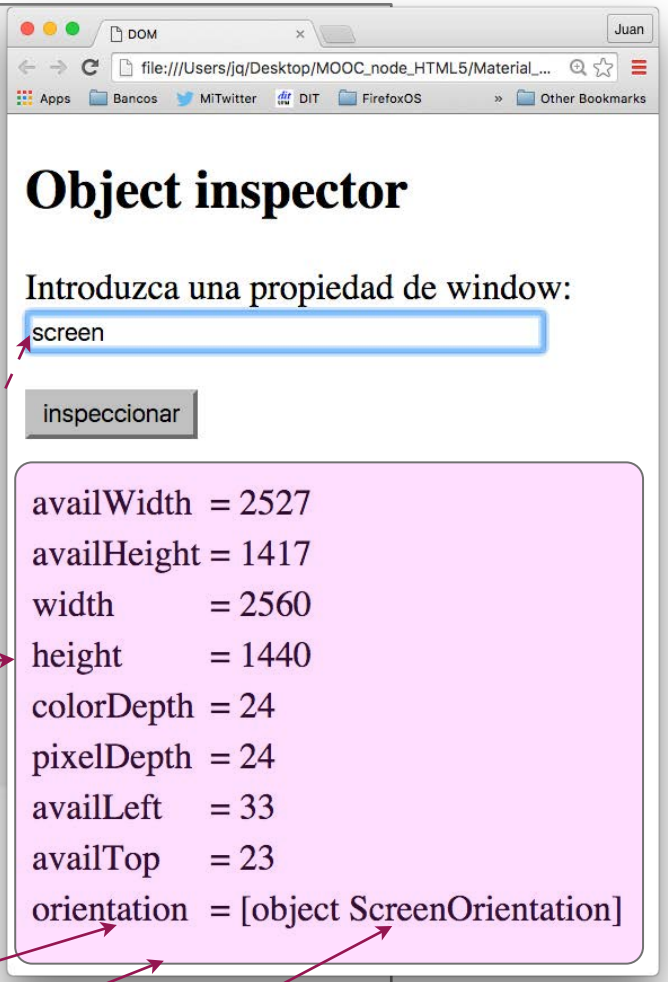
- **location:** propiedad que contiene el URL a la página en curso
  - **location = "http://www.upm.es"** Carga la página en el navegador
  - **location.reload()** re-carga la página en curso
    - Doc: https://developer.mozilla.org/en-US/docs/Web/API/Window/location
- **history**: propiedad con la historia de navegación
  - Métodos para navegar por la historia: **history.back()**, **history.forward()**, ...
    - Doc: https://developer.mozilla.org/en-US/docs/Web/API/Window/history
- **screen**: dimensiones de la pantalla
  - **width, height, availWidth, availHeight**: para adaptar apps a pantallas móviles
    - Doc: https://developer.mozilla.org/en-US/docs/Web/API/Window/screen

window.screen

```
<!DOCTYPE html>
<html><head>
<title>DOM</title>
<meta charset="UTF-8">
<style>
  span {font-weight: bold}
</style>
</head><body>
<h1>Propiedades de window</h1>

La propiedad location.href contiene el URL:
<br>
<span id="i1"></span>
<p>
Los pixels de mi pantalla (screen.width y screen.height) son:
<span id="i2"></span>

<script>
document.getElementById("i1").innerHTML = location.href;

var p = document.getElementById("i2")
p.innerHTML = screen.width + " x " + screen.height;
</script>
</body>
</html>
```

Object inspector

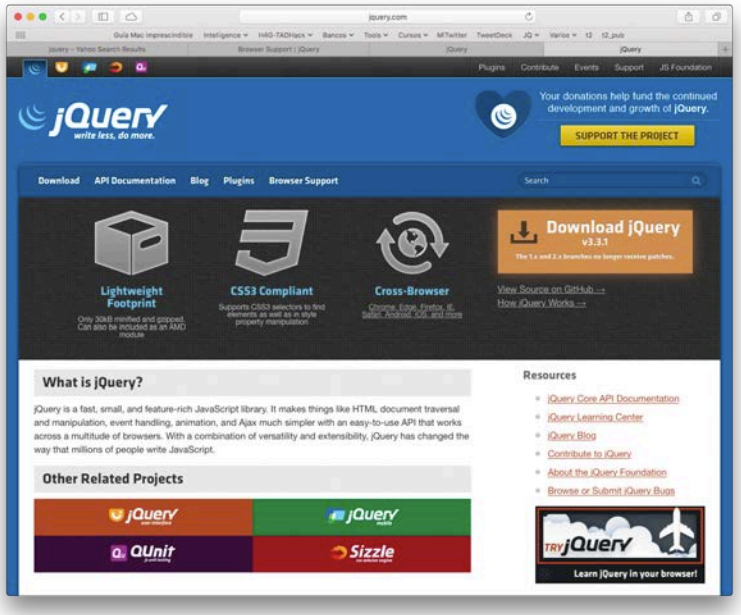
```
<!DOCTYPE html>
<html>
<head>
<title>DOM</title>
<meta charset="UTF-8">
</head>
<body>

<h2> Object inspector </h2>

Introduzca una propiedad de window: <br>
<input type="text" size="40" id="prop"><p>
<button onClick="inspeccionar()">inspeccionar</button>

    <!-- tabla con propiedades de screen -->
<table id="tabla"></table>

<script>
function inspeccionar (){
  var obj = window[document.getElementById("prop").value];
  var tabla = document.getElementById("tabla");
  tabla.innerHTML = "";

  for (var i in obj) {  //cada iteración genera una fila de la tabla
    tabla.innerHTML += "<tr><td>" + i + "</td><td> = "+ obj[i]+"</td></tr>";
  }
}
</script>
</body>
</html>
```



Introducción a jQuery

Juan Quemada, DIT - UPM

Librería jQuery



- **jQuery**
  - Lema: **write less, do more**
    - **Multi-navegador:** Ejecuta en Chrome, Firefox, Safari, Edge, IE, Opera, ...
      - Gestiona: búsqueda y proceso de objetos DOM, eventos, estilos CSS, AJAXs, ...
    - Documentación y descarga: http://jquery.com/
- Aquí se ven dos de sus componentes principales:
  - **La función jQuery(..)** o **$(..)**: busca elementos del árbol DOM
  - **Métodos de proceso**: procesan elementos DOM
- **jQuery 3.3.1** (versión utilizada aquí, de septiembre de 2017)
  - jQuery evoluciona: los proyectos deben utilizar la última versión

Objetos y función jQuery: $(..)

- **jQuery** representa los **objetos DOM** como **objetos jQuery**
  - Los **objetos jQuery** permiten **procesar** el árbol DOM de forma **más eficaz**
    - **Arrays de objetos jQuery** se procesan con **un solo método,** sin necesidad de bucles
- Función jQuery: **jQuery("<selector:CSS>")** o **$("<selector_CSS>")**
  - devuelve la colección de **objetos jQuery** que casan con el **<selector CSS>**
    - Si **no casa ninguno**, devuelve un **objeto jQuery vacío**
  - **<selector CSS>** utiliza la **sintaxis CSS** para seleccionar objetos DOM

```
document.getElementById("fecha")
        // es equivalente a:
$("#fecha")
```

- Además la función jQuery **convierte** objetos **DOM** y **HTML** a **objetos jQuery**

```
$("<ul><li>Uno</li><li>Dos</li></ul>")   // convierte HTML a objeto jQuery
$(document.getElementById("fecha"))   // convierte objeto DOM a objeto jQuery
```

Selectores tipo CSS de jQuery

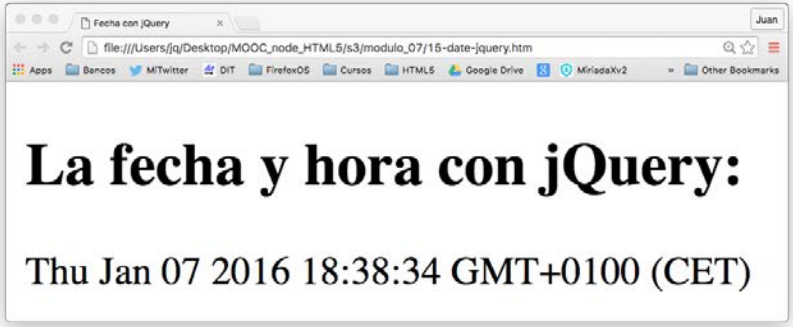
```
SELECTORES DE MARCAS CON ATRIBUTO ID
$("h1#d83")      /* devuelve objeto con marca h1 e id="d83" */
$("#d83")        /* devuelve objeto con con id="d83" */

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 arbol */
$("h1 + p")      /* devuelve array de objetos con marca p adyacente a h1 del mismo nivel */
.................
```

Métodos de jQuery: ejemplos

- Método **html(<código html>)**
  - **$("h1").html("Hello World!")** sustituye por **Hello World!** el **innerHTML** de todos los elementos **h1**
- Método **html()**
  - **$("h1").html()** devuelve una colección con todos los **innerHTML** de todos los elementos **h1**
- Método **append("Hello World!")**
  - **$("#id3").append("Hello World!")** añade "Hello World!" al **innerHTML** del elemento con id=**"id3"**
- Método **val(<valor>)**
  - **$("#id3").val("3")** asigna el **valor "3"** al atributo **value** del elemento con id=**"id3"**
- Método **attr(<atributo>, <valor>)**
  - **$(".lic").attr("rel", "license")** asigna **"license"** al atributo **rel** a todos los elementos con **class="lic"**
- Método **addClass(<valor>)**
  - **$("ul").addClass("visible")** asigna el **valor "visible"** al atributo **class** de todos los elementos **<ul>**
- Métodos **hide()** y **show()**
  - **Oculta** o **muestra** objetos DOM de la página HTML cargada en el navegador
- Ejemplo de composición serie: **$("h1").html("Title").addClass("view").show()**
  - Asigna el HTML**"Title"** a todos los <h1>, les añade el atributo **class="view"** y los hace visibles
- La API de jQuery es grande y esta bien estructurada (es util conocer sus detalles)
  - Más información en: http://api.jquery.com/category/manipulation/

Fecha y hora con jQuery

- Una **librería JavaScript** externa se identifica por su **URL**:
  - <script type="text/javascript" **src="jquery-2.1.4.min.js.js"** > </script>
- **$("#fecha")** obtiene el objeto jQuery
  - del **elemento HTML** con **id="fecha"**
- **$("#fecha").html(new Date())**
  - inserta **new Date()** como **HTML interno**
    - del **objeto jQuery** devuelto por **$("#fecha")**
  - es equivalente a
    - **document.getElementById("fecha").innerHTML = new Date();**




```
<!DOCTYPE html>
<html>
<head>
<title>Fecha con jQuery</title>
<script type="text/javascript"
        src="jquery-2.1.4.min.js">
</script>
</head>

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

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

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

Selecciona el elemento DOM con atributo id="fecha": .

Asigna la fecha y hora a innerHTML del objeto DOM seleccionado.

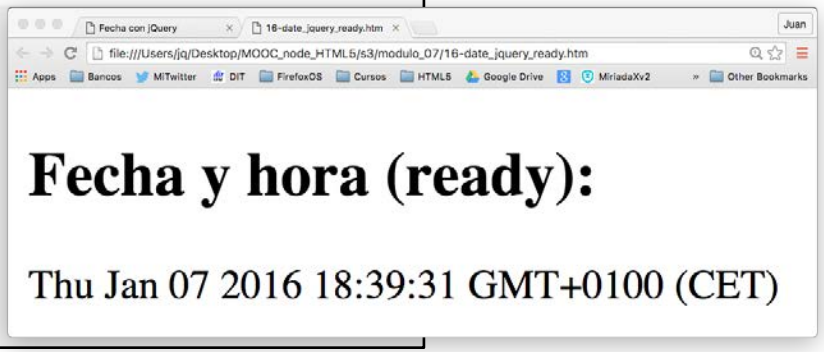
Los 4 usos de la función jQuery: $(..)

- **Acceso a DOM**: **$("selector CSS")**
  - Devuelve un **array** con los **objetos jQuery** que casan con **<selector CSS>**
    - Programas mas **cortos**, **eficaces** y **multi-navegador** que con JavaScript directamente
- **Convierte HTML a objeto jQuery**: **$("
Uno
Dos
")**
  - Devuelve **objeto jQuery** equivalente al **HTML**
    - Mecanismo simple para convertir **HTML** en **jQuery**
- **Convierte objeto DOM a jQuery**: **$(document.getElementById("fecha"))**
  - Transforma objeto DOM en objeto jQuery equivalente
    - Tiene compatibilidad total con DOM y con otras librerías basadas en DOM
- **Esperar a DOM-construido**: **$(function(){..código..})**
  - **Ejecuta el código** de la función cuando el **árbol DOM está construido**
    - Equivalente a **ejecutar el código** asociado al evento **onload**

Función ready: árbol DOM construido

- **$(document).ready(function() { ..código..)}**
  - Ejecuta el código (bloque) de la función cuando el **árbol DOM está construido**
    - Es decir, dicho bloque se ejecuta cuando ocurre el evento **onload** de <body>
  - Se recomienda utilizar la invocación abreviada: **$(function() { ..código.. })**

```
<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript" src="jquery-2.1.4.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 cargados anteriormente durante la navegación
  - La cache identifica los recursos por igualdad de URLs
    - Un nuevo recurso se carga de alguna cache (navegador, ..) si tiene el mismo URL que otro ya guardado
      - Cargarlo de la cache es más rápido que bajarlo del servidor, especialmente de la del navegador
- CDNs Web: utilizan el mismo URL (a Google, jQuery, …) en muchas páginas
  - Así se maximiza la probabilidad de que los recursos estén ya en la cache

```
<html>
<head>
<script type="text/javascript"
        src="https://code.jquery.com/jquery-2.1.4.min.js" >
</script>

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

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

Ejercicio

- Dada una página HTML que utiliza la librería jQuery con el siguiente contenido

```
<!DOCTYPE html>
<html> .... <body>
   <h4 id="id1" >Título</h4>
   <p id="id2">Coche</p>
   <div id="id3"></div>
   <div id="id4" class="Casa">Casa</div>
   <script type="text/javascript"> ...script con expresiones de abajo ...  </script>
</body>
</html>
```

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

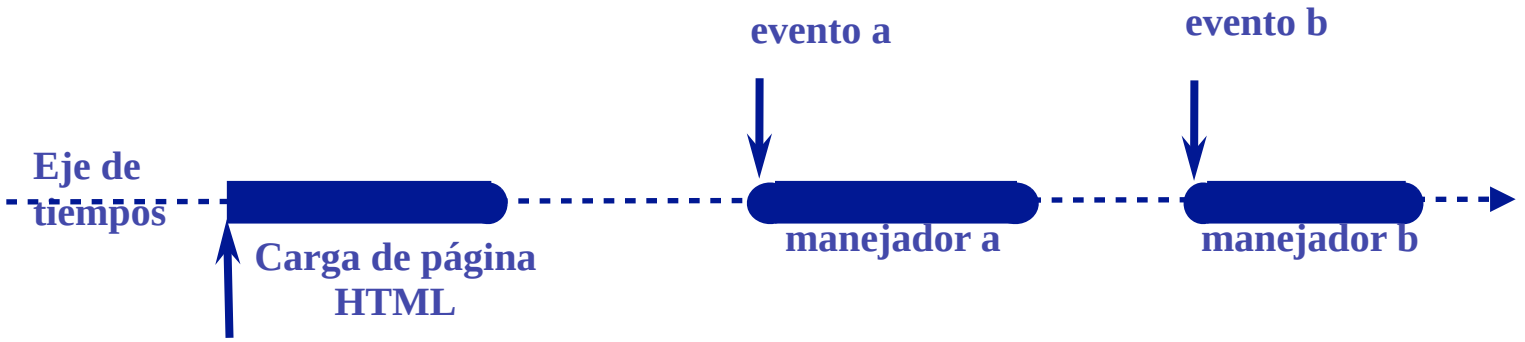
$("#id1").html() => undefined o null, error_de_ejecución, "", "Título**,** "Coche", "Casa"
$("#id2").html() => undefined o null, error_de_ejecución, "", "Título", "Coche**,** "Casa"
$("#id3").attr("innerHTML") => undefined o null, error_de_ejecución, ""**,** "Título", "Coche", "Casa"
$("#id4").html() => undefined o null, error_de_ejecución, "", "Título", "Coche", "Casa"
$(".Casa").html() => undefined o null, error_de_ejecución, "", "Título", "Coche", "Casa"
$(".Casa").attr("innerHTML") => undefined o null, error_de_ejecución, "", "Título", "Coche", "Casa"
$("#id5").html() => undefined o null**,** error_de_ejecución, "", "Título", "Coche", "Casa"
$("#id6").attr("innerHTML") => undefined***** o null, error_de_ejecución, "", "Título", "Coche", "Casa"
$("#id4").attr("innerHTML") => undefined o null, error_de_ejecución, "", "Título", "Coche", "Casa"

Eventos, DOM e interacción

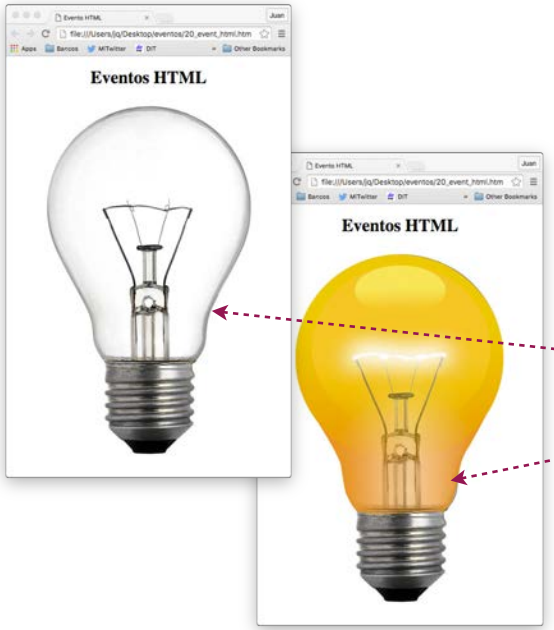
Juan Quemada, DIT - UPM

Eventos y Manejadores

- JavaScript utiliza eventos para interaccionar con el entorno
  - La norma de JavaScript incluye muchos eventos diferentes
    - Temporizadores, clicks en boton, tocar en pantalla, pulsar tecla, ...
      - Ver: https://developer.mozilla.org/en-US/docs/Web/Events
- **Manejador** (callback) de evento
  - Es una **función** que se ejecuta al ocurrir el evento
- El script inicial debe configurar los manejadores (callbacks)
  - a ejecutar cuando ocurra cada evento que deba ser atendido



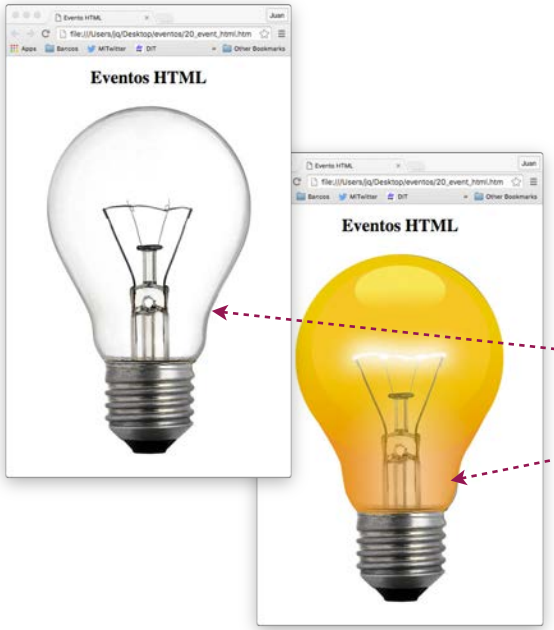
Eventos en HTML




```
<!DOCTYPE html>
<html><head><title>Evento HTML</title>
<style> body{text-align:center;} </style>
</head><body>
  <h1>Eventos</h1>

  <img src="lamp_off.jpg"
       ondblclick="this.src='lamp_off.jpg';"
       onclick="this.src='lamp_on.jpg';"
       />
</body>
</html>
```

- HTML permite definir **eventos** de interacción con el usuario
  - Tradicionalmente los **eventos** se definían con **atributos** de elementos HTML
    - **onclick** (hacer clic), **ondbclick** (hacer doble clic), **onload** (página cargada), …
      - http://librosweb.es/libro/javascript/capitulo_6/modelo_basico_de_eventos_2.html
- El **valor asignado** al atributo es **código JavaScript** (string) ejecutado al ocurrir el evento
  - **this** referencia el **objeto DOM** del elemento HTML asociado al evento
    - **onclick="this.src ='lamp_on.jpeg'"** asigna al **atributo src,** de la **imagen,** el URL al icono **lamp_on.jpeg**
      - **this.src** se refiere a la **propiedad** asociada al **atributo src** del objeto DOM de la **imagen**
    - **ondbclick="this.src ='lamp_off.jpeg'"** asigna al **atributo src,** de la **imagen,** el URL al icono **lamp_off.jpeg**
- El ejemplo asocia 2 **eventos** a la imagen (elemento **<img .... >**)
  - **onclick="this.src ='lamp_on.jpeg'"** muestra el icono **lamp_on.jpeg** al hacer **clic** en la **imagen**
  - **ondbclick="this.src ='lamp_off.jpeg'"** muestra el icono **lamp_off.jpeg** al hacer **doble clic** en la **imagen**

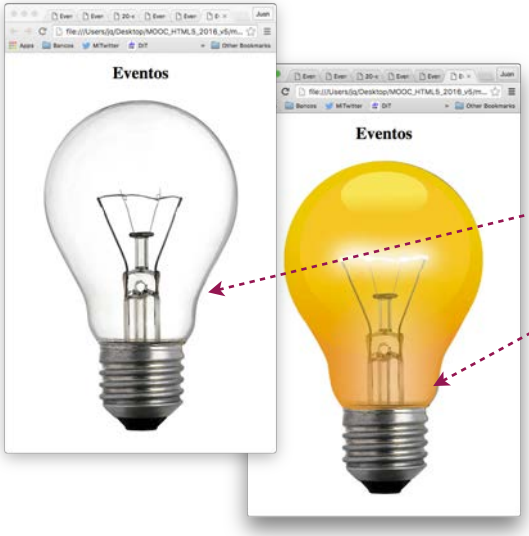
Eventos en HTML




```
<!DOCTYPE html>
<html><head><title>Evento HTML</title>
<style> body{text-align:center;} </style>
</head><body>
  <h1>Eventos</h1>

  <img src="lamp_off.jpg"
       ondblclick="this.src='lamp_off.jpg';"
       onclick="this.src='lamp_on.jpg';"
       />
</body>
</html>
```

- Las apps Web son muy grandes y hoy se recomienda
  - separar HTML, CSS y JavaScript en partes o ficheros diferentes
    - Así cada parte puede generarse por un equipo diferente
- Normalmente
  - HTML se incluye en el cuerpo de la página HTML
  - CSS se incluye en la cabecera o en un fichero separado
  - JavaScript se incluye en la cabecera o en un fichero separado

Eventos en JavaScript




```
<!DOCTYPE html>
<html><head><title>Evento</title><meta charset="UTF-8">
<style> body{text-align:center;} </style>

  <script type="text/javascript">

    function inicializar() {
       var img = document.getElementById('i1');

       img.addEventListener("dblclick", (event) => img.src='lamp_off.jpg');
       img.addEventListener("click",    (event) => img.src='lamp_on.jpg');
    }
  </script>
</head>   <!-- El arbol DOM está ya construido al ocurrir onload -->
<body onload="inicializar()">
  <h1>Eventos</h1>
  <img src="lamp_off.jpg" id="i1"/>
</body></html>
```

- **objeto.addEventListener(tipo_ev, manejador)** asocia manejadores a **eventos**
  - Además existe el método **removeEventListener(..)** para **eliminar el manejador**
    - https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
    - https://developer.mozilla.org/en-US/docs/Web/Events
  - Un elemento DOM puede tener varios manejadores asociados, que se ejecutan en orden de instalación
- El **manejador** es un literal de función**: (event) => { .. código.. }** o **function (event) { .. código.. }**
  - **event** es un objeto con información del evento ocurrido: https://developer.mozilla.org/en-US/docs/Web/API/Event
    - El parámetro **event** no es necesario en este ejemplo, pero se ha incluido para hacerlo explícito
- La función **inicializar()** define los eventos y se invoca al ocurrir el **evento onload**
  - El **evento onload** (<body>) ocurre con el **árbol DOM** construido -> los manejadores se instalan bien

Eventos en jQuery


```
<!DOCTYPE html>
<html><head><title>Evento jQuery</title><meta charset="UTF-8">
<style> body{text-align:center;} </style>

<script type="text/javascript" src="jquery-3.3.1.min.js" > </script>
<script type="text/javascript">
  $(function(){
    var img = $('#i1');

    img.on('dblclick', (event) => img.attr('src', 'lamp_off.jpg'));
    img.on('click',    (event) => img.attr('src', 'lamp_on.jpg'));
  });
</script>
</head>
<body>
  <h1>Eventos</h1>
  <img src="lamp_off.jpg" id="i1">
</body>
</html>
```

- jQuery permite también la definición de eventos en objetos con el método **on()**
  - objetoJQuery.**on(tipo_ev, manejador)**
    - Mas información en: https://api.jquery.com/on/ o http://api.jquery.com/category/events/
- El **manejador** es un literal de función**: (event) => { .. código.. }** o **function (event) { .. código.. }**
  - **event:** objeto con información del evento ocurrido: http://api.jquery.com/category/events/event-object/
    - El parámetro **event** no necesita utilizarse en este ejemplo, pero se ha incluido para hacerlo explícito
- Los **tipos de eventos** (**tipo_ev**) utilizados por **on(..)** y **off()** son los de **addEventListener(..)**
  - Más información en https://developer.mozilla.org/en-US/docs/Web/Events

Ejemplo Mini-calculadora

- La propiedad **value** da acceso al contenido de un cajetín: **<input type="text"..>**
  - Sirve para acceder al contenido tecleado (leer), como para mostrar nuevo contenido (asignar)
- La calculadora usa 2 eventos
  - Evento 1: **clicar** en el **cajetín**
  - Evento 2: **clicar** en el **botón x²**
- El atributo **onclick="f1()"**
  - Asocia código **"f1()"** (invocar función f1()...)
    - A hacer clic en el elemento HTML con el atributo
- Los eventos se atienden con:
  - Evento 1: función -> **vaciar()**
  - Evento 2: función -> **cuadrado()**

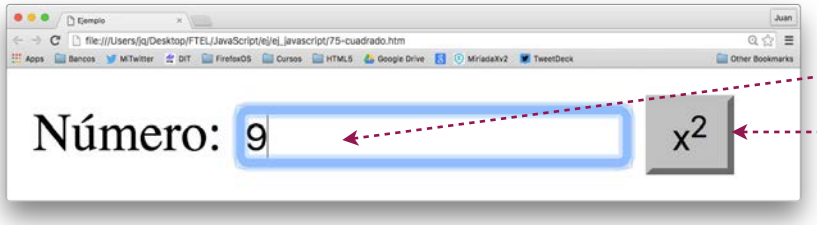

```
<!DOCTYPE html><html><head>
<title>Ejemplo</title><meta charset="utf-8">
<script type="text/javascript">

function vaciar () {
  document.getElementById("n1").value = "";
}
function cuadrado() {
  var num = document.getElementById("n1");
  num.value = num.value * num.value;
}
</script>
</head><body>
  Número:
  <input type="text" id="n1" onclick="vaciar()">

  <button onclick="cuadrado()">
    x<sup>2</sup>
  </button>
</body></html>
```

Calculadora jQuery

Obtener objeto jQuery (DOM) deL cajetín: **$("#n1")**

Obtener objeto jQuery (DOM) del botón: **$("#b1")**

- jQuery simplifica la calculadora
- Modificaciones
  - Debemos **importar** la **librería jQuery**
  - Definir eventos en **función ready**
    - con **método on(..)**
      - y con **árbol DOM ya construido**
  - **Obtener** objetos jQuery con **$("#…")**
  - **Obtener texto** de cajetín con **val()**
  - **Asignar texto** en cajetín con **val(texto)**




```
<!DOCTYPE html>
<html><head><title>Calculadora</title>
            <meta charset="utf-8">
  <script type="text/javascript"
          src="jquery-3.3.1.min.js">
  </script>

  <script type="text/javascript">
  $(function() {
    $("#n1").on("click",
      function(){ $("#n1").val("");}
    );

    $("#b1").on("click",
      function() {
        var num = $("#n1");
        num.val(num.val() * num.val());
      }
    );
  });
  </script>
</head>

<body>
  Número:
  <input type="text" id="n1">

  <button id="b1"> x<sup>2</sup> </button>
</body>
</html>
```

Importar librería jQuery del mismo directorio de la app

Evento click en cajetín

Vaciar el cajetín

Evento click en botón x²

Calcular resultado obteniendo el string tecleado en cajetín con **num.val()** y guardando el resultado con **num.val(..resultado..)**.

"Bubbling" de eventos

- Los eventos burbujean ("bubbling") hacía arriba en el árbol DOM
  - Primero se ejecutan los manejadores (si existen) del elemento DOM sobre el que se ha hecho clic
  - Luego se ejecutan los manejadores (si existen) del elemento DOM que **contiene al anterior**
  - Y así hasta llegar a la raíz
- **elemento**.on(event, **selector**, manejador)
  - **selector**: restringe **los descendientes** de **elemento** que disparan el evento, por ejemplo
    - **".class_a"** restringe a los elementos de esta clase
    - **"#n1"** restringe al elemento con este identificador
  - Doc: https://api.jquery.com/on/
- **this** referencia el elemento DOM asociado al manejador en ejecución
- **event.target** es una referencia al elemento que provocó el evento
  - **event** es el parámetro del manejador en
    - on(tipo_ev, selector, (**event**) => {.. código ..})


```
<!DOCTYPE html>
<html><head><title>Calculadora</title>
            <meta charset="utf-8">
  <script type="text/javascript"
          src="jquery-3.3.1.min.js">
  </script>

  <script type="text/javascript">
  $(function() {
     $("body").on("click", "#n1",
      function(){ $("#n1").val("");}
     );

     $("body").on("click", "#b1",
       function() {
         var num = $("#n1");
         num.val(num.val() * num.val());
       }
     );
  });
  </script>
</head>

<body>
  Número:
  <input type="text" id="n1">

  <button id="b1"> x<sup>2</sup> </button>
</body>
</html>
```

Evento click en cajetín: click en elemento contenido en **<body>** con **id="n1"**

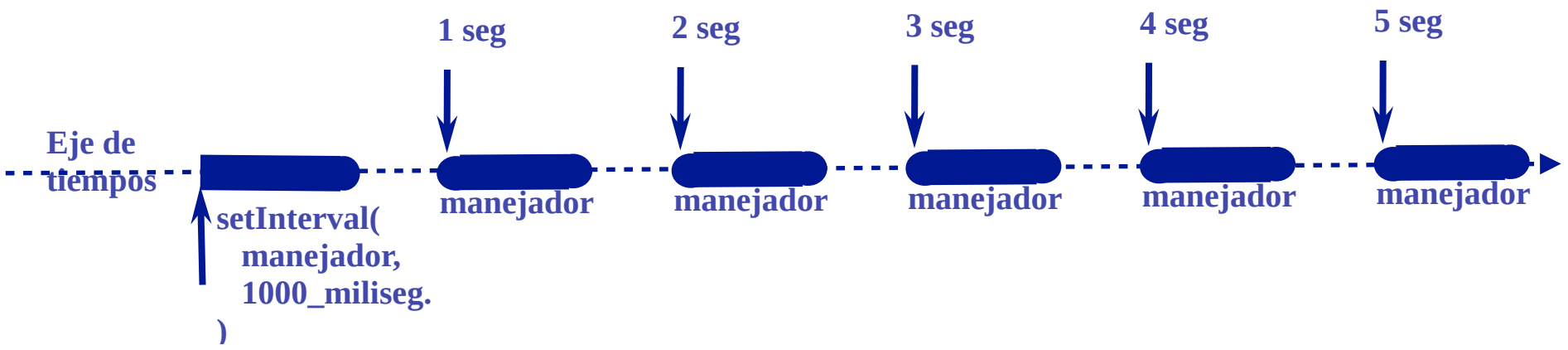
Evento click en botón x^2: click en elemento contenido en **<body>** con **id="b1"**.

Los dos **manejadores** de eventos se asocian a **<body>**, pero la activación del evento se filtra en función del identificador del evento clicado ("target").

Eventos periódicos con setInterval(….)

- JavaScript tiene una función **setInterval (..)**
  - para programar eventos periódicos

- **setInterval (manejador, periodo_en_milisegundos)**
  - tiene 2 parámetros
    - **manejador:** función que se ejecuta al ocurrir el evento
    - **periodo_en_milisegundos:** tiempo entre eventos periódicos



Reloj

```
<!DOCTYPE html>
<html><head><title>Reloj</title>
    <meta charset="UTF-8">
<script type="text/javascript"
        src="jquery-3.3.1.min.js" >
</script>

<script type="text/javascript">

  function mostrar_fecha( ) {
    $("#fecha").html(new Date( ));
  }

  $(function(){
      // Define evento periodico, ocurre
      // cada segundo (1000 miliseg)
    setInterval(mostrar_fecha, 1000);

          // muestra fecha al cargar
    mostrar_fecha();
  });
</script>
</head>
<body>

  <h2>Reloj</h2>

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

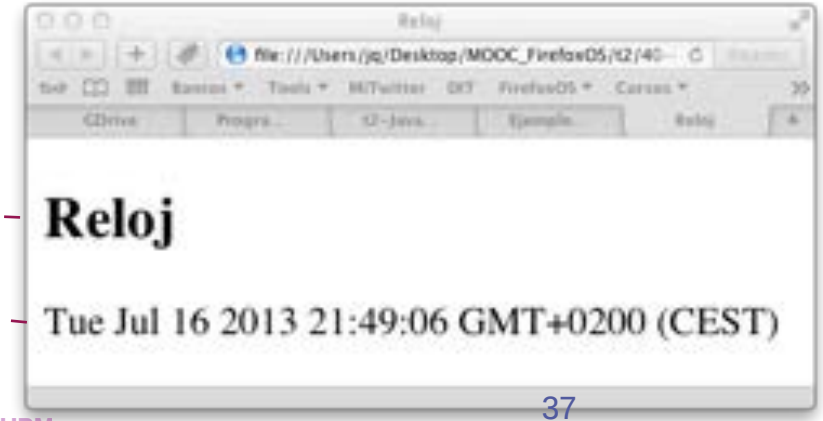
Importar librería jQuery del mismo directorio de la app

Mostrar fecha en bloque

Define un evento que actualiza la hora cada segundo

Muestra la hora al cargar la página Web

- El reloj utiliza un evento periódico
  - para actualizar cada segundo
    - la fecha y la hora mostrada en el bloque
- El evento periódico se programa con
  - **setInterval(..manejador.., ..periodo..)**
    - El manejador es una función
    - El periodo se da en milisegundos
      - con **árbol DOM ya construido**
  - **setInterval(mostrar_fecha, 1000)**
    - Ejecuta la función mostrar_fecha()
      - cada segundo (1000 milisegundos)
- Más información en
  - https://developer.mozilla.org/en-US/Add-ons/Code_snippets/Timers



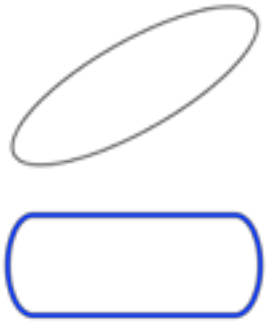
SVG: Scalable Vector Graphics

Juan Quemada, DIT - UPM

SVG: Scalable Vector Graphics

- Formato de representación de gráficos vectoriales
  - Pueden cambiar de tamaño sin perdida de calidad
- Recursos
  - Galeria Wikimedia: http://commons.wikimedia.org/wiki/Category:SVGs_by_subject
  - Editor SVG: http://svg-edit.googlecode.com/svn/branches/2.5.1/editor/svg-editor.html
  - Tutorial: https://developer.mozilla.org/en-US/docs/Web/SVG
  - Tutorial: http://www.w3schools.com/svg/

http://commons.wikimedia.org/wiki/File:Compass.svg



http://commons.wikimedia.org/wiki/SVG_examples

Ejemplo “Ajuste SVG”

- **“Ajuste SVG”** ilustra como reescalar una imagen SVG
  - Las imagenes en SVG reescalan sin perder calidad
    - porque son gráficos vectoriales
    - tutorial: http://www.w3schools.com/svg/
  - Las imágenes GIT, JPEG o PNG no reescalan bien
    - porque tienen una resolución determinada
- Esta WebApp tiene 2 botones: **“+”** y **“-”**
- Cada vez que pulsamos uno de estos botones
  - el tamaño de la imagen debe aumentar o disminuir un 10%
    - según pulsemos “+” y “-”

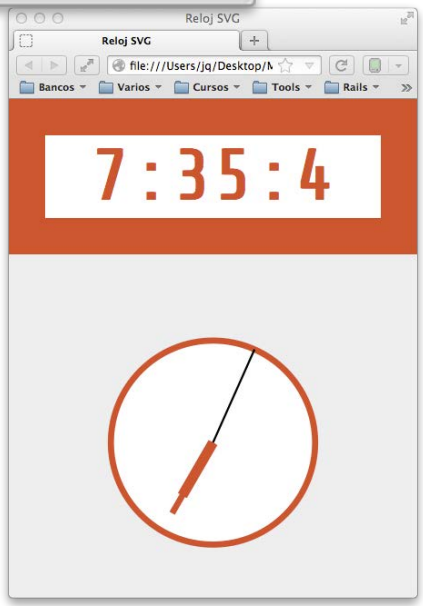
55-image_adjust_svg_zepto.htm UNREGISTERED

```
<!DOCTYPE html>
<html><head><title>Ejemplo SVG</title>
<script type="text/javascript"
        src="zepto.min.js" > </script>
<script type="text/javascript">
$(function(){
  var img  = $('#img');

  $('#incr').on('click', function(){
    img.width(img.width()*1.1);
    img.height(img.height()*1.1);
  });

  $('#decr').on('click', function(){
    img.width(img.width()/1.1);
    img.height(img.height()/1.1);
  });
});
</script>
</head>
<body>
<h4> Ejemplo SVG </h4>
<button  type="button"  id="decr">-</button>
<button  type="button"  id="incr">+</button><p>

<img src="55-smiley.svg" id="img"
     width="70"  height="70" />
</body>
</html>
```

Ejemplo SVG

Ejemplo “Reloj SVG”

- **“Reloj SVG”** genera un reloj sencillo con SVG
  - El reloj se compone de
    - Un círculo negro
    - Tres lineas para las manecillas del reloj
- SVG puede animarse con javaScript
  - modificando la representación DOM del reloj
    - Versión 1: las manecillas se mueven con transform
      - https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/transform
    - Version 2: Calcula las coordenadas de las manecillas
- Se añade estilo CSS
  - Mejora el aspecto y adapta al tamaño de la pantalla



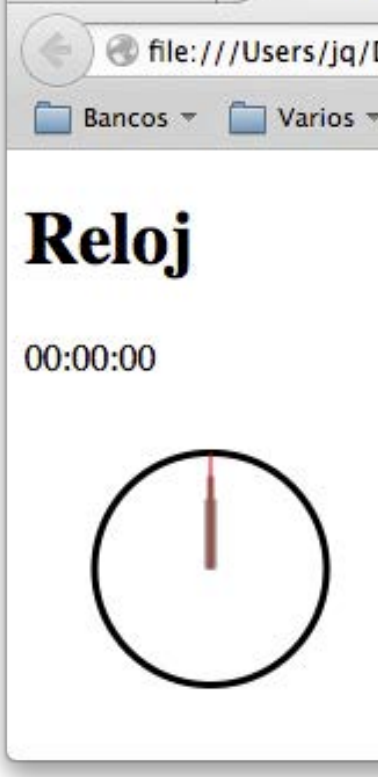
```
<!DOCTYPE html>
<html>
<head><title>Reloj SVG</title>
      <meta charset="UTF-8"></head>
<body>
<h3>Reloj SVG</h3>
<div id="tex">texto</div>

<svg>
  <circle id="myCircle"
    cx="80" cy="80" r="50"
    fill="white" stroke="black" stroke-width="3"/>
  <line id="hor"
    x1="80" y1="80" x2="110" y2="80"
    style="stroke:grey;stroke-width:5"/>
  <line id="min"
    x1="80" y1="80" x2="80" y2="40"
    style="stroke:grey;stroke-width:3"/>
  <line id="seg"
    x1="80" y1="80" x2="115" y2="45"
    style="stroke:red;stroke-width:1"/>
</svg>

</body>
</html>
```

Reloj SVG

05-clock_transform.htm UNREGISTERED

```
<!DOCTYPE html><html>
<head><title>Galería</title><meta charset="UTF-8">
<script type="text/javascript" src="http://zeptojs.com/zepto.min.js" ></script>
<script>
  function animar() {
    var d = new Date();
    var s = d.getSeconds();    // grados = segundos * 6
    var m = d.getMinutes();    // grados = minutos * 6
    var h = d.getHours();
    var hh = h*30 + m/2;  // grados de la manecilla de horas
    $("#tex").html(h + ":" + m + ":" + s);
    $("#hor").attr("transform", "rotate(" + hh + " 80 80)");
    $("#min").attr("transform", "rotate(" + m*6 + " 80 80)");
    $("#seg").attr("transform", "rotate(" + s*6 + " 80 80)");
  }
  $(function(){
    setInterval(animar, 1000);
    animar();
  })
</script>
</head>
<body>
<h1>Reloj</h1>

<div id="tex">texto</div>

<svg>
  <circle id='myCircle'  cx='80' cy='80' r='50'
          fill='white' stroke='black'  stroke-width='3' />
  <line id="hor" x1='80' y1='80' x2='80' y2='50'
        style='stroke:grey;stroke-width:5'/>
  <line id="min" x1='80' y1='80' x2='80' y2='40'
        style='stroke:grey;stroke-width:3'/>
  <line id="seg" x1='80' y1='80' x2='80' y2='30'
        style='stroke:red;stroke-width:1'/>
</svg></body></html>
```

SVG: Reloj animado con “transform”



Animar manecillas con coordenadas

- Para animar las manecillas del reloj
  - se incluye un script que cada segundo
    - recalcula las coordenadas exteriores
      - de las manecillas del reloj
  - El secundero tiene una longitud de 50 pixels
  - El minutero tiene una longitud de 40 pixels
  - La manecilla horaria de 30 pixels

- Las coordenadas x2, y2 de las manecillas de horas, minutos y segundos se calculan con las funciones
  - x2(tiempo, unidades_por_vuelta, x1, radio)
  - y2(tiempo, unidades_por_vuelta, y1, radio)

SVG: Reloj animado con coordenadas

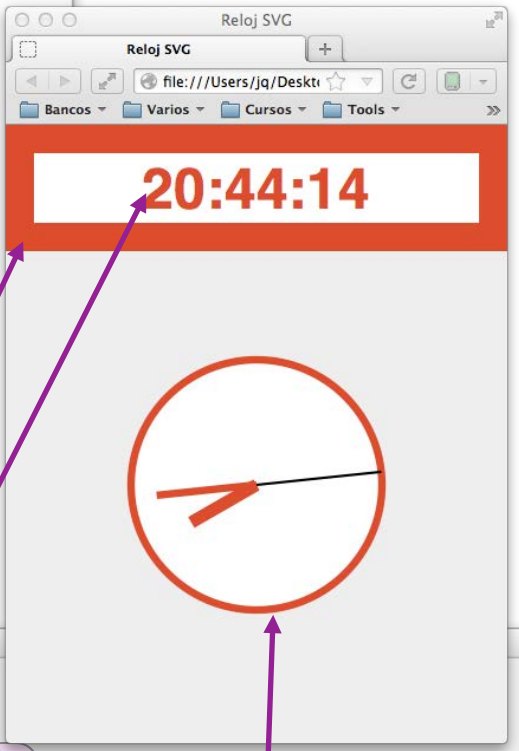
```
<!DOCTYPE html>
<html>
<head>
  <title>Reloj SVG</title>
  <meta charset="UTF-8">
  <script type="text/javascript" src="http://zeptojs.com/zepto.min.js" >
  </script>

<script type="text/javascript">
  function x2(n,i,x1,r) {return x1 + r*Math.sin(2*Math.PI*n/i);};
  function y2(n,i,y1,r) {return y1 - r*Math.cos(2*Math.PI*n/i);};

  function mostrar_hora( ) {
    var d = new Date();
    var h = d.getHours();
    var m = d.getMinutes();
    var s = d.getSeconds();
    $('#tex').html(h + ":" + m + ":" + s);
    $('#seg').attr('x2', x2(s,60,80,50)).attr('y2', y2(s,60,80,50));
    $('#min').attr('x2', x2(m,60,80,40)).attr('y2', y2(m,60,80,40));
    $('#hor').attr('x2', x2(h,12,80,30)).attr('y2', y2(h,12,80,30));
  }

  $(function(){
    setInterval(mostrar_hora, 1000);
    mostrar_hora();
  })
</script>
```

Relojes con “estilo”

- Usando CSS e imágenes se pueden diseñar
  - Las capturas muestran pequeños cambios de diseño
    - que cambian muy significativamente la apariencia del reloj
  - Hacer clic en estos URLs para verlos
    - https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/09-clock_CSS.htm
    - https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/10_clock_CSS_a.html
    - https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/10_clock_CSS_b.htm
    - https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/10_clock_CSS_c.htm
    - https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/10_clock_CSS_d.htm

SVG: Reloj con estilo CSS

```
<!DOCTYPE html>
<html><head><title>Reloj SVG</title><meta charset="UTF-8">
<style type="text/css">

body, html {
    margin: 0px;
    padding: 0px;
    height: 100%;
    width: 100%;
    background-color: #eee;
}

.mitad {
    background-color: #db4e36;
    color: #db4e36;
    font-family: sans-serif;
    font-size: 5em;
    font-weight: bold;
    text-align: center;
    padding: 0.5em;
}

#tex {
  padding-top: 0.2em;
  background-color: #FFF;
}
#reloj {
    height: 100%;
    width: 100%;
}
</style>
....
</html>
```



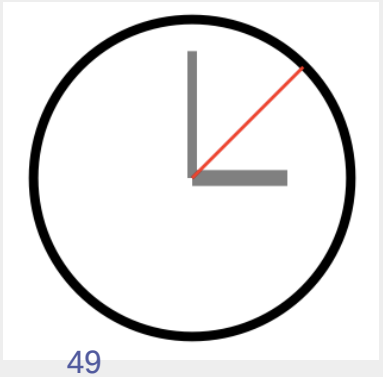

```
.......
<body>
<div class="mitad">
  <div id="tex">texto</div>
</div>

<svg id="reloj" viewBox="0 0 200 200">
  <circle id="myCircle" cx="100" cy="70" r="50
    fill="white" stroke="#db4e36" stroke-width
  <line id="hor" x1="100" y1="70" x2="110" y2=
    style="stroke:#db4e36;stroke-width:5"/>
```

Objetos SVG

- Los objetos SVG se pueden definir también como objetos externos en XML
  - Para importarlos con:
    - <img>, <object>, <embed>, <iframe>
  - Tutorial: http://tavmjong.free.fr/INKSCAPE/MANUAL/html/Web-Use.html

73-clock.svg — UNREGISTERED

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
          "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="120" height="120">

  <circle id='myCircle'  cx='60' cy='60' r='50'
          fill='white' stroke='black'  stroke-width='3' />

  <line x1='60' y1='60' x2='90' y2='60'
        style='stroke:grey;stroke-width:5'/>
  <line x1='60' y1='60' x2='60' y2='20'
        style='stroke:grey;stroke-width:3'/>
  <line x1='60' y1='60' x2='95' y2='25'
        style='stroke:red;stroke-width:1'/>
</svg>
```



Memoria local: localStorage y sessionStorage

Juan Quemada, DIT - UPM

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 confia 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

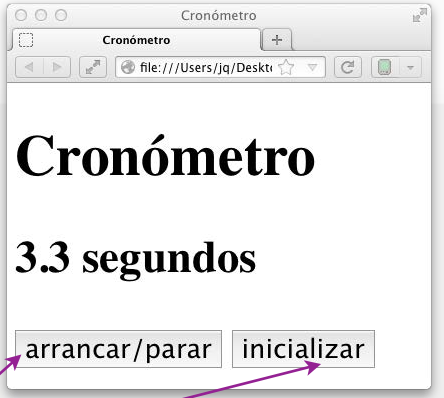
- El cronómetro utiliza ahora la variable
  - **localStorage.c**
    - para guardar la cuenta de segundos

- 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>
```



Javascript: Notepad

Juan Quemada, DIT - UPM

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

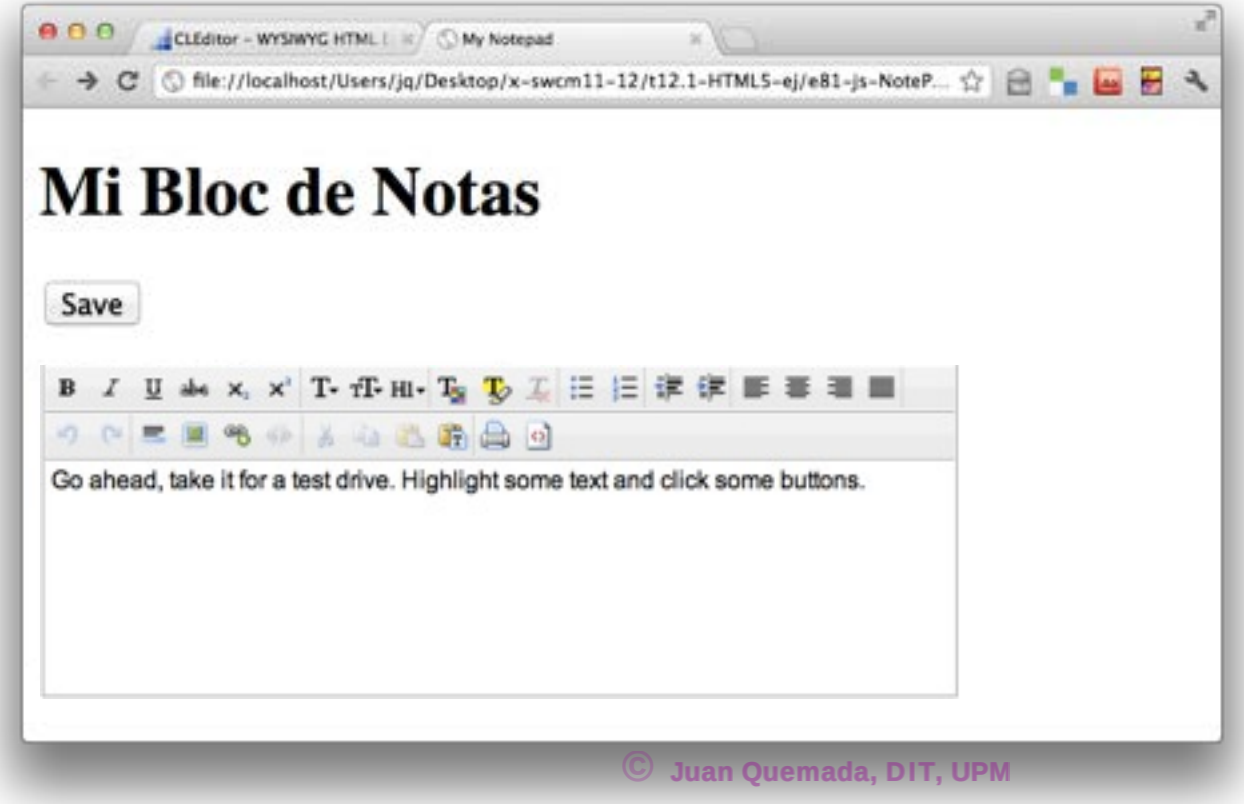
- Al pulsar los botones (Editar o Salvar)
  - Una página se activa con el método **show()** de Zepto
    - cuando la otra se desactiva con el método **hide()**

```html
<!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
    - Codifcado en javascript/jQuery
  - http://premiumsoftware.net/cleditor/



JavaScript y HTML5:
iframes y origin policy

Juan Quemada, DIT - UPM

iFrame

- **iFrame**
  - Importa un recurso Web
    - en un 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 introduzcan virus
    - 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 á**rbol 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 estén en nuestra cadena de confianza

Ejemplo con iFrames

Flexbox

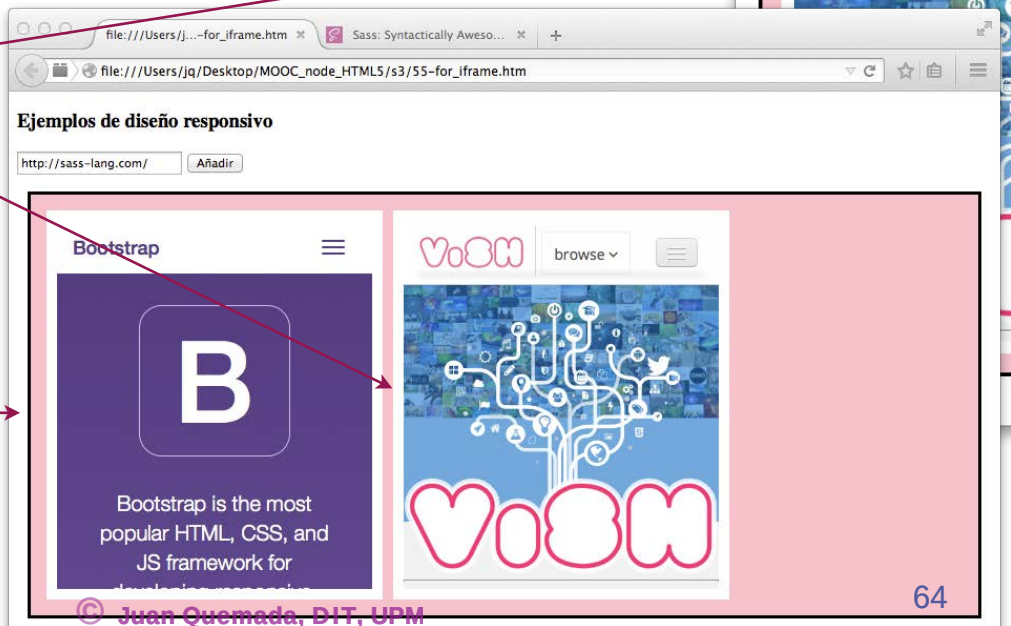
Flexbox permite un diseño responsivo fácil y flexible:

display: flex;
flex-wrap: wrap;

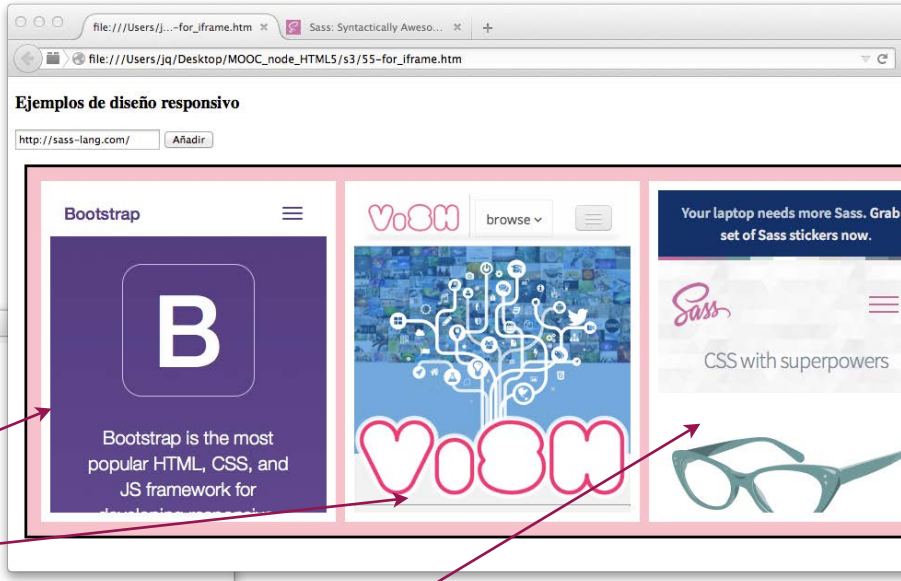
coloca cada iframe a la derecha del anterior. Al llegar al limite pasa a la línea siguiente.

55-for_iframe.htm UNREGISTERED

```
<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>
```



Código



55-for_iframe.htm

```
<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>
```

55-for_iframe.htm UNREGISTERED

```
<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>
```

Final del tema

Muchas gracias!