JavaScript:
Introducción a la librería jQuery

Librerías JavaScript: jQuery

- Las **librerías JavaScript** facilitan la programación **multi-navegador**
  - Se diseñan para funcionar en IE, Firefox, Safari, Chrome, Opera, ...
    - **Ahorran mucho tiempo** -> se deben utilizar siempre que existan
      - Por ejemplo. **jQuery**, jQuery UI, D3, Bootstrap, Prototype, PhoneGap, …
- **jQuery** es muy popular (http://jquery.com/) -> Lema: **write less, do more**
  - Se distribuye con licencia abierta (MIT) y facilita mucho la programación JavaScript de cliente
    - Procesa objetos DOM, gestiona eventos, animaciones, estilos CSS, Ajax, ..
  - **jQuery 1.x** y **2.x** son **dos versiones** de la librería con la **misma interfaz (API)**
- **jQuery 1.x** (última versión 7-1-15: **1.11.3**)
  - Fue la primera y mantiene compatibilidad desde **Explorer 6+**
    - Esta optimizada para compatibilidad legacy y es **más pesada** que jQuery 2.x
- **jQuery 2.x** (última versión 7-1-15: **2.1.4**)
  - Creada recientemente y mantiene compatibilidad desde **Explorer 9+**
    - Está optimizada para móviles y es **mucho mas ligera** que jQuery 1.x

	Internet Explorer	Chrome, Firefox	Edge	Safari	Opera	iOS	Android
jQuery 1.x	6+	(Current - 1) or Current	Current	5.1+	12.1x, (Current - 1) or Current	6.1+	2.3, 4.0+
jQuery 2.x	9+						

La función jQuery

- **Objeto jQuery**: representación **equivalente a un objeto DOM**
  - **Mas eficaz de procesar**, tanto de forma individual, como en bloque (array)
- Función jQuery: **jQuery("<selector CSS>")** o **$("<selector CSS>")**
  - devuelve el **objeto jQuery** que **casa con <selector CSS>**
    - Si **no casa ninguno**, devuelve **null** o **undefined**
  - **<selector CSS>** selecciona objetos DOM **igual que CSS**

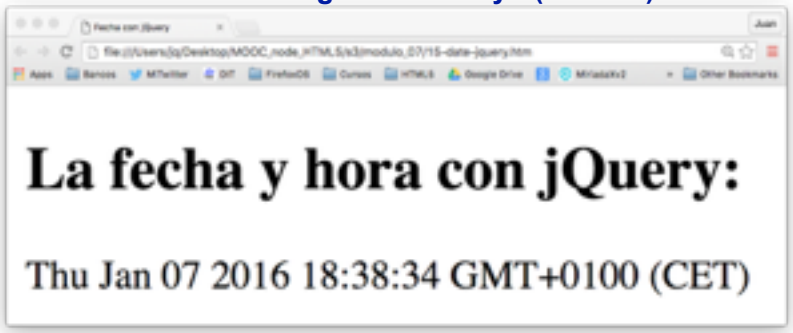
```
document.getElementById("fecha")
        // es equivalente a:
$("#fecha")
```

- Además la función jQuery **convierte 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

- 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>
```

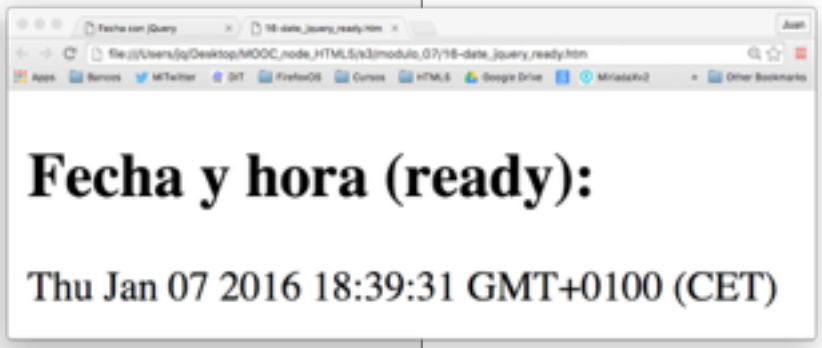
Selecciona el elemento DOM con atributo id="fecha": .

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

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>
```

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 manipulación

- Los objetos jQuery se manipulan con métodos de la librería
  - Más información en: http://api.jquery.com/category/manipulation/
- Método **html(<código html>)**
  - **$("#id3").html("Hello World!")** sustituye el **innerHTML** del elemento con **id="id3** por **"Hello World!"**
    - Es equivalente a: **document.getElementById("fecha").innerHTML = "Hello World!"**
- Método **html()**
  - **$("#id3").html()** devuelve el **innerHTML** del elemento con **"#id3"**
- 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>)**
  - **$(".lic").attr("rel", "license")** asigna **"license"** al atributo **rel** a todos los elementos con **class="lic"**
    - Una **gran ventaja de jQuery** es que **puede hacer asignaciones en bloque** sin utilizar bucles como aquí!
- Método **addClass(<valor>)**
  - **$("ul").addClass("visible")** asigna el **valor "visible"** al atributo **class** de todos los elementos **<ul>**
    - Una **gran ventaja de jQuery** es que **puede hacer asignaciones en bloque** sin utilizar bucles como aquí!

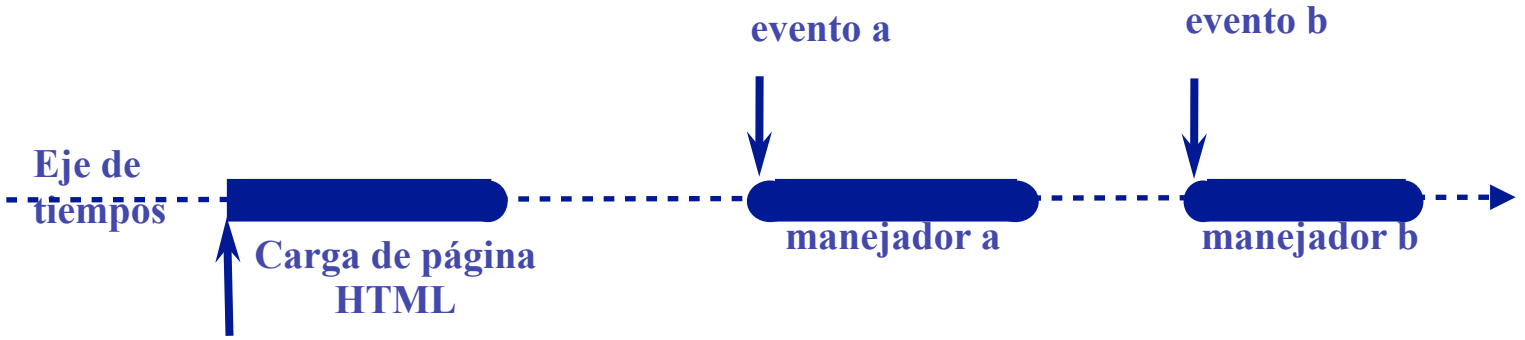
Los 4 modos de la función jQuery

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

JavaScript:
Eventos DOM y jQuery

Eventos y Manejadores

- JavaScript utiliza eventos para interaccionar con el entorno
  - Hay eventos de muchos tipos
    - Temporizadores, clicks en boton, tocar en pantalla, pulsar tecla, ...
- Un 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 HTML</h1>

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

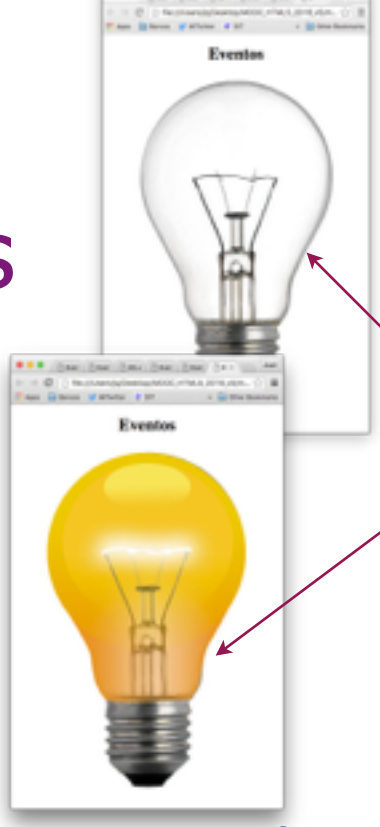
- En programas grandes es recomendable **separar HTML**, **CSS** y **JavaScript**
  - Debe estar en **ficheros diferentes** (o al menos en **partes claramente separadas)**
    - De esta forma el **cuerpo (body) solo contiene HTML** y la **cabecera** (head) incluye **CSS** y **JavaScript**
- La forma habitual de definir **eventos directamente en JavaScript** es con
  - **objetoDOM.addEventListener(evento, manejador)**
    - También existe un método **removeEventListener(..)** para **eliminar el evento**
      - https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
  - Tradicionalmente el manejador se asignaba a una propiedad con el nombre del atributo HTML
    - **objeto.evento = manejador**
- jQuery usa la función **on()** para definir eventos y **off()** para eleminarlos
  - **objetoJQuery.on(evento, manejador)**
    - http://api.jquery.com/category/events/event-handler-attachment/

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

- Los **eventos** se definen asociados a **onload** para que el **árbol DOM esté ya está construído**
  - El manejador del **evento onload** hay que invocarlo o definirlo o en HTML o en sino en un script al final
- La norma de JavaScript incluye **muchos eventos** diferentes
  - Se pueden ver en https://developer.mozilla.org/en-US/docs/Web/Events
    - Los **nombres de los eventos** son **diferentes** del de los atributos de eventos
  - La forma tradicional (objeto.evento = manejador) esta en desuso y no la ilustramos
- El **manejador** del evento es una **función** (objeto de la clase Función)
  - Puede pasarse directamente como un literal de función con el código del manejador (como aquí)
    - o cualquier otro objeto Function, como por ejemplo el nombre de una función definida en otro lugar
      - OJO! debe ser la función (su código) y no su invocación

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-2.1.4.min.js" > </script>
<script type="text/javascript">
  $(function(){
    var img = $('#i1');

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

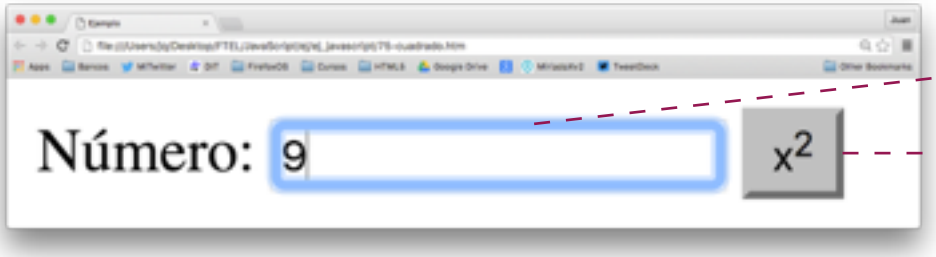
- jQuery también permite definir eventos en objetos jQuery con el método **on()**
  - **objetoJQuery.on(evento, manejador)**
    - jQuery conserva métodos asociados a eventos individuales de versiones anteriores, pero está recomendado usar solo **on()** y **off()**
      - http://api.jquery.com/category/events/
- Los **nombres y tipos de eventos** utilizados por los métodos **on(..)** y **off()**
  - son los mismos que los utilizados con el método addEventListener(..)
    - se pueden ver en https://developer.mozilla.org/en-US/docs/Web/Events
- El **manejador** del evento es una **función** (objeto de la clase Función) ejecutado al ocurrir el evento
  - Puede pasarse directamente como un literal de función con el código del manejador (como aquí)
    - o cualquier otro objeto Function, como por ejemplo el nombre de una función definida en otro lugar
      - OJO! debe ser el nombre de la función (su código) y no su invocación

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(..)**
      - 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 char set="utf-8">
  <script type="text/javascript"
          src="jquery-2.1.4.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

Evento click

Vaciar el cajetín

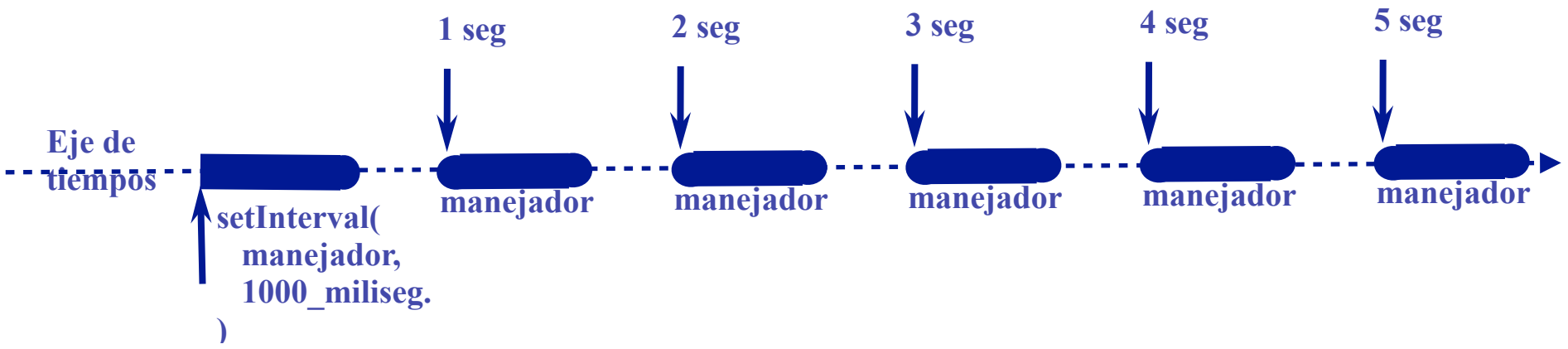
Evento click

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

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-2.1.4.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>
```

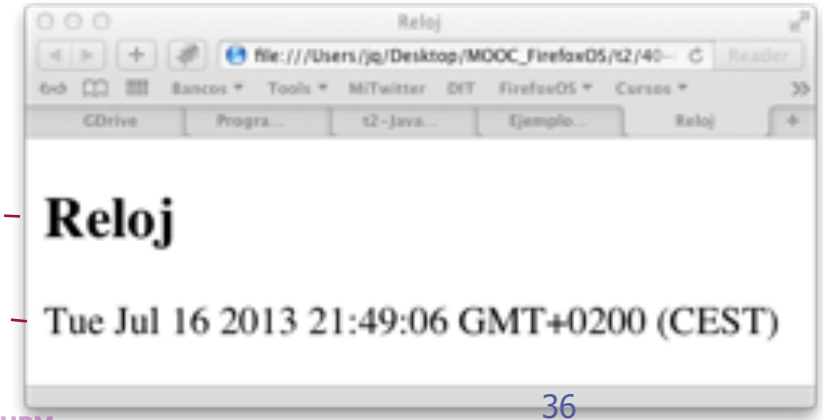
Importar librería jQuery

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



Final del tema

Muchas gracias!