



# Tipo Number de JavaScript

# Números: tipo number

◆ Los números se representan con literales de

- **Enteros:** 32
  - ◆ Entero máximo: **9007199254740992**
- **Decimales:** 32.23
- **Coma flotante:** 3.2e1 (3,2x10)
  - ◆ Rango real: **1,797x10<sup>308</sup> --- 5x10<sup>-324</sup>**

◆ Todos los números son del tipo **number**

◆ Todos los números se representan igual internamente

- coma flotante de doble precisión (64bits)

◆ El tipo number incluye 2 valores especiales

- **Infinity:** representa desbordamiento
- **NaN:** representa resultado no numérico

<b>10 + 4</b>	<b>=&gt; 14</b>	<b>// sumar</b>
<b>10 - 4</b>	<b>=&gt; 6</b>	<b>// restar</b>
<b>10 * 4</b>	<b>=&gt; 40</b>	<b>// multiplicar</b>
<b>10 / 4</b>	<b>=&gt; 2.5</b>	<b>// dividir</b>
<b>10 % 4</b>	<b>=&gt; 2</b>	<b>// operación resto</b>

//decimales dan error de redondeo  
**0.1 + 0.2 => 0,300000000000004**

<b>3e2</b>	<b>=&gt; 300</b>
<b>3e-2</b>	<b>=&gt; 0,03</b>

<b>+10/0</b>	<b>=&gt; Infinity</b>	<b>//desborda</b>
<b>-10/0</b>	<b>=&gt; -Infinity</b>	<b>//desborda</b>

<b>5e500</b>	<b>=&gt; Infinity</b>	<b>//desborda</b>
--------------	-----------------------	-------------------

# Conversión a enteros

- ◆ Cuando JavaScript calcula expresiones
  - **convirtiendo tipos** según necesita
    - ◆ según las prioridades de operadores
- ◆ Conversión a **entero (o real)**
  - **booleano**: true a 1, false a 0
  - **String**: Convierte número a valor o NaN
  - **null**: a 0,      **undefined**: a NaN

<b>'67' + 13</b>	<b>=&gt; 6713</b>
<b>+'67' + 13</b>	<b>=&gt; 80</b>
<b>+'6.7e1' + 13</b>	<b>=&gt; 80</b>
<b>'xx' + 13</b>	<b>=&gt; 'xx13'</b>
<b>+'xx' + 13</b>	<b>=&gt; NaN</b>
<b>13 + true</b>	<b>=&gt; 14</b>
<b>13 + false</b>	<b>=&gt; 13</b>

<b>.</b> <b>[]</b>	<b>Acceso a propiedad o invocar método; índice a array</b>
<b>new</b>	<b>Crear objeto con constructor de clase</b>
<b>()</b>	<b>Invocación de función/método o agrupar expresión</b>
<b>++ --</b>	<b>Pre o post auto-incremento; pre o post auto-decremento</b>
<b>! ~</b>	<b>Negación lógica (NOT); complemento de bits</b>
<b>+</b> <b>-</b>	<b>Operador unitario, números. signo positivo; signo negativo</b>
<b>delete</b>	<b>Borrar propiedad de un objeto</b>
<b>typeof void</b>	<b>Devolver tipo; valor indefinido</b>
<b>*</b> <b>/</b> <b>%</b>	<b>Números. Multiplicación; división; modulo (o resto)</b>
<b>+</b>	<b>Concatenación de string</b>
<b>+</b> <b>-</b>	<b>Números. Suma; resta</b>
<b>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</b>	<b>Desplazamientos de bit</b>
<b>&lt; &lt;= &gt; &gt;=</b>	<b>Menor; menor o igual; mayor; mayor o igual</b>
<b>instanceof in</b>	<b>¿objeto pertenece a clase?; ¿propiedad pertenece a objeto?</b>
<b>== != === !==</b>	<b>Igualdad; desigualdad; identidad; no identidad</b>
<b>&amp;</b>	<b>Operacion y (AND) de bits</b>
<b>^</b>	<b>Operacion ó exclusivo (XOR) de bits</b>
<b> </b>	<b>Operacion ó (OR) de bits</b>
<b>&amp;&amp;</b>	<b>Operación lógica y (AND)</b>
<b>  </b>	<b>Operación lógica o (OR)</b>
<b>?:</b>	<b>Asignación condicional</b>
<b>=</b>	<b>Asignación de valor</b>
<b>OP=</b>	<b>Asig. con operación: += -= *= /= %= &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;= &amp;= ^=  =</b>
<b>,</b>	<b>Evaluación múltiple</b>

## Operadores JavaScript

Los operadores están ordenados verticalmente por prioridades. Los más altos se evalúan antes.

# Modulo Math

- ◆ El Modulo Math contiene
  - constantes y funciones matemáticas

- ◆ Constantes

- Números: E, PI, SQRT2, ...
- ...

- ◆ Funciones

- sin(x), cos(x), tan(x), asin(x), ....
- log(x), exp(x), pow(x, y), sqrt(x), ....
- abs(x), ceil(x), floor(x), round(x), ....
- min(x,y,z,...), max (x,y,z,...), ...
- random()

Mas info:

[http://www.w3schools.com/jsref/jsref\\_obj\\_math.asp](http://www.w3schools.com/jsref/jsref_obj_math.asp)

**Math.PI** => 3.141592653589793

**Math.E** => 2.718281828459045

// numero aleatorio entre 0 y 1

**Math.random()** => 0.7890234

**Math.pow(3,2)** => 9 // 3 al cuadrado

**Math.sqrt(9)** => 3 // raíz cuadrada de 3

**Math.min(2,1,9,3)** => 1 // número mínimo

**Math.max(2,1,9,3)** => 9 // número máximo

**Math.floor(3.2)** => 3

**Math.ceil(3.2)** => 4

**Math.round(3.2)** => 3

**Math.sin(1)** => 0.8414709848078965

**Math.asin(0.8414709848078965)** => 1



# La Clase Array de JavaScript

# 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]
```



# JSON - JavaScript Object Notation

# JSON

- ◆ JSON: formato textual de representación de tipos y objetos JavaScript
  - <http://json.org/json-es.html>
- ◆ Un **objeto JavaScript** se transforma a un **string JSON** con
  - **JSON.stringify(object)**
- ◆ Un **string JSON** se transforma en el **objeto original** con
  - **JSON.parse(string\_JSON)**

```
var x = {a:1, b:{y:[false, null, ""]}}, y, z;  
  
y = JSON.stringify(x);      => '{"a":1, "b":{"y":[false, null, ""]}}'  
  
z = JSON.parse(y);          => {a:1, b:{y:[false, null, ""]}}
```

# Serialización de datos

- ◆ Serialización:
  - transformación **reversible** de un tipo u objeto (en memoria) en un **string equivalente**
- ◆ La serialización es un formato de intercambio de datos
  - **Almacenar datos** en un fichero
  - **Enviar datos** a través de una línea de comunicación
  - **Paso de parámetros** en interfaces REST
- ◆ En JavaScript se realiza desde ECMAScript 5 con
  - **JSON.stringify(...)** y **JSON.parse(...)**
- ◆ Otros formatos de serialización: XML, HTML, XDR(C), ...
  - XML está siendo desplazados por JSON
    - ◆ Hay bibliotecas de JSON para los lenguajes más importantes

# Características de JSON

## ◆ JSON puede serializar

- objetos, arrays, strings, números finitos, true, false y null
  - ◆ NaN, Infinity y -Infinity se serializan a null
  - ◆ Objetos Date se serializan a formato ISO
    - la reconstrucción devuelve un string y no el objeto original
- No se puede serializar
  - ◆ Funciones, RegExp, errores, undefined

## ◆ Admite filtros para los elementos no soportados

- ver doc de APIs JavaScript

**JSON.stringify(new Date())**      => "2013-08-08T17:13:10.751Z"

**JSON.stringify(NaN)**                => 'null'

**JSON.stringify(Infinity)**        => 'null'



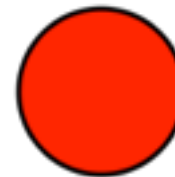
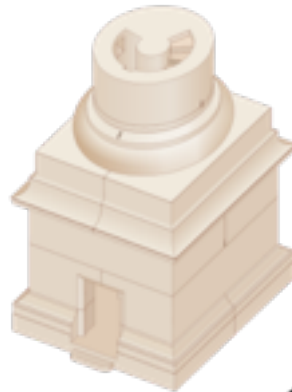
# HTML5 SVG - Scalable Vector Graphics

# SVG: Scalable Vector Graphics

- ◆ Formato de representación de gráficos vectoriales
  - Pueden cambiar de tamaño sin pérdida de calidad
- ◆ Recursos
  - Galeria Wikimedia: [http://commons.wikimedia.org/wiki/Category:SVGs\\_by\\_subject](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: <http://www.w3schools.com/svg/>



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



[http://commons.wikimedia.org/wiki/SVG\\_examples](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 “-”



```

<!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>



</body>
</html>

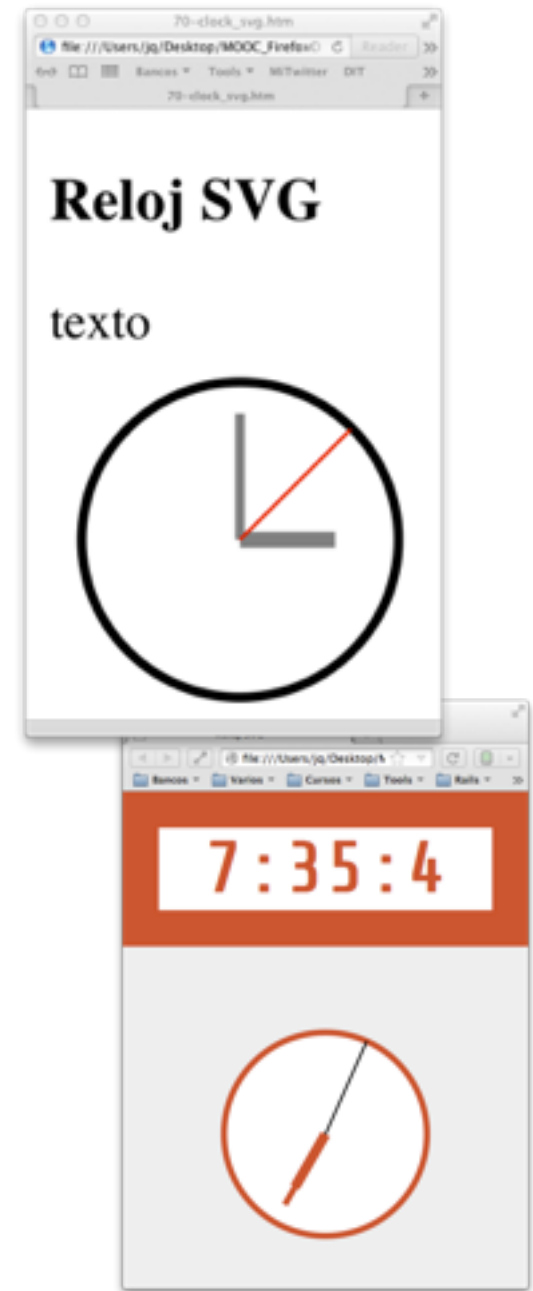
```

# Ejemplo SVG



# Ejemplo “Reloj SVG”

- ◆ “**Reloj SVG**” genera un reloj sencillo con SVG
  - El reloj se compone de
    - ◆ Un círculo negro
    - ◆ Tres líneas para las manecillas del reloj
- ◆ SVG puede animarse con javaScript
  - modificando la representación DOM del reloj
    - ◆ tal y como se ilustra en el ejemplo siguiente
- ◆ Se añade estilo CSS
  - Mejora el aspecto
  - Lo 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

# Animar las manecillas del reloj

- ◆ 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 minuterio 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
  - $x2(\text{tiempo}, \text{unidades\_por\_vuelta}, x1, \text{radio})$
  - $y2(\text{tiempo}, \text{unidades\_por\_vuelta}, y1, \text{radio})$





# SVG: Reloj animado



```
<head>
<title>Reloj SVG</title><meta charset="UTF-8">
<script type="text/javascript" src="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(){
    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));
    }

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

```

<!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%;
}

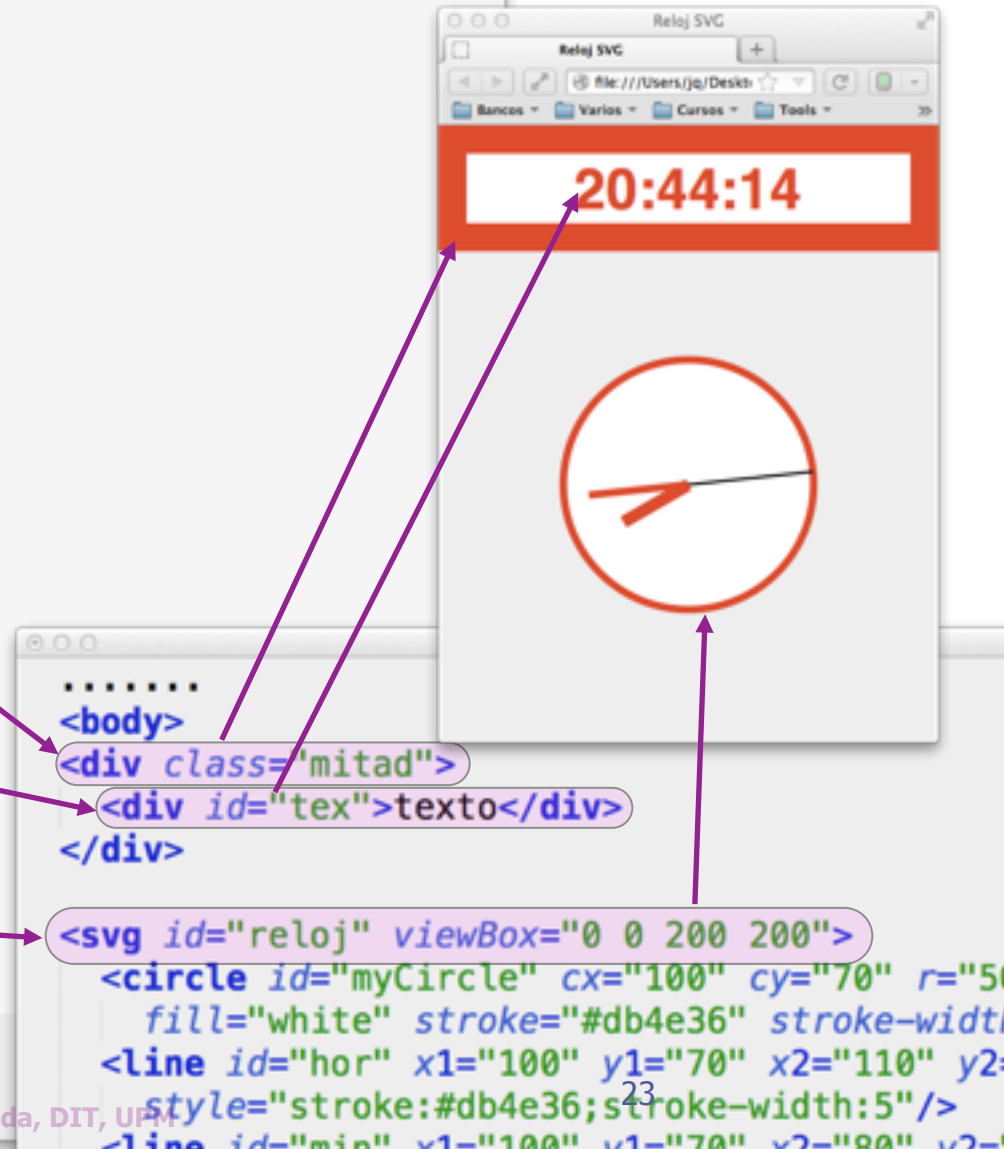
```

```

</style>
....
</html>

```

# SVG: Reloj con estilo CSS



# 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

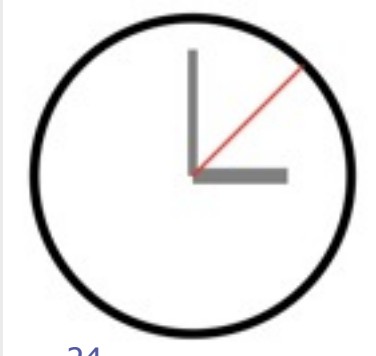
UNREGISTERED

```
<?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>
```



24





# HTML5 CANVAS

# Mapas de bits con CANVAS

- ◆ **CANVAS** define un **mapa de bits**
  - Se define en HTML con la marca **<canvas>**
    - ◆ Permite programar en Javascript
      - aplicaciones interactivas, juegos, 2D, 3D, ....
  - Esta soportado en los principales navegadores
- ◆ Características
  - Tiene resolución fija y no reescala con calidad
    - ◆ **<canvas id="c1" width="150" height="350"> Texto alternativo</canvas>**
- ◆ Tutoriales
  - [https://developer.mozilla.org/En/Canvas\\_tutorial](https://developer.mozilla.org/En/Canvas_tutorial)
  - <http://www.desarrolloweb.com/manuales/manual-canvas-html5.html>
  - <http://www.html5canvastutorials.com/>



# Ejemplo “Reloj CANVAS”

- ◆ “**Reloj CANVAS**” es similar al reloj programado con SVG
  - Pero programado en el canvas
    - ◆ Tiene el mismo círculo y manecillas del de SVG
- ◆ SVG puede animarse con JavaScript
  - modificando la representación DOM del reloj
    - ◆ tal y como se ilustra en el ejemplo siguiente



```

<!DOCTYPE html><html>
<head><title>Reloj CANVAS</title><meta charset="UTF-8">
<script type="text/javascript" src="zepto.min.js"></script>
<script type="application/javascript">
function myLine(ctx,x1,y1,x2,y2,width,color) {
  ctx.beginPath();           // comenzar nueva linea
  ctx.moveTo(x1,y1);         // Comienzo de linea
  ctx.lineTo(x2,y2);         // Final de linea

  ctx.strokeStyle = color;   // color de linea
  ctx.lineWidth = width;     // anchura de linea: 5 puntos
  ctx.stroke();              // dibujar linea
}

function myCircle(ctx,x,y,r,width,color) {
  ctx.beginPath();           // comenzar figura
                               // añadir arco (circulo entero):
  ctx.arc(x,y,r,0,2*Math.PI); // ctx.arc(x, y, r, start, stop)

  ctx.strokeStyle = color;   // color de la linea del circulo
  ctx.lineWidth = width;     // anchura de la linea del circulo
  ctx.stroke();              // dibujar circulo
}

$(function() {
  var c=document.getElementById("myCanvas"); // obtiene CANVAS
  if (c.getContext) { // CANVAS soportado?
    var ctx = c.getContext("2d"); // define contexto 2D

    myCircle(ctx,80,80,50,3,"black"); // esfera del reloj

    myLine(ctx,80,80,110,80,5,"grey"); // manecilla de horas
    myLine(ctx,80,80,80,40,3,"grey"); // manecilla de minutos
    myLine(ctx,80,80,115,45,1,"red"); // manecilla de segundos
  }
})
</script>
</head><body>
<h4> Reloj CANVAS</h4>
<div id="tex">texto</div>
<canvas id="myCanvas" width="140" height="140"></canvas>
</body></html>

```

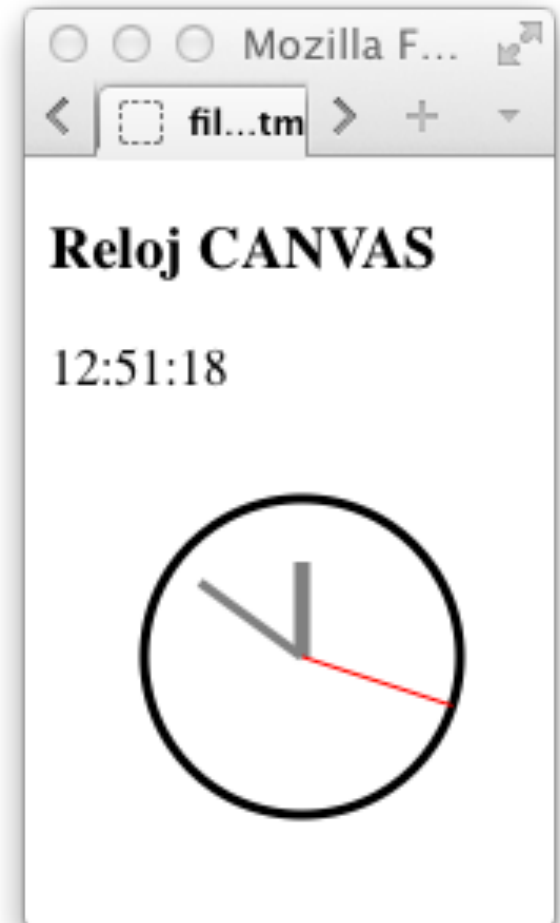
© Juan Quemada, DIT, UPM



# Reloj CANVAS

# Animar las manecillas del reloj

- ◆ El script calcula cada segundo las manecillas
  - una vez calculadas borra el canvas
    - ◆ y vuelve a dibujar el reloj completo
  - Secundero (50 px), minuterero (40 px), hora (30 px)
- ◆ 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)`
- ◆ `myLine(...)` y `myCircle(...)`
  - dibujan lineas y circulos



```

<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="zepto.min.js" ></script>
<script type="application/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)};

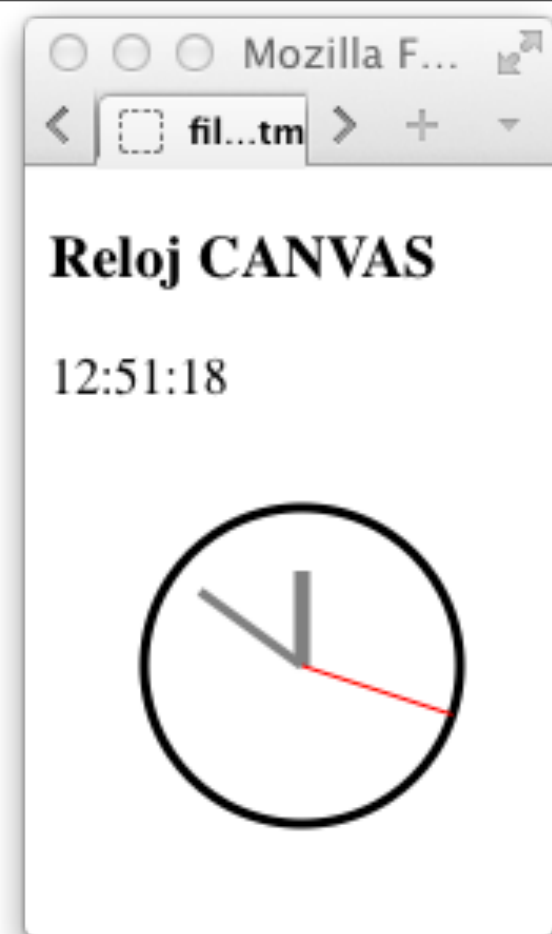
function myLine(ctx,x1,y1,x2,y2,width,color) { ... }
function myCircle(ctx,x,y,r,width,color) { ... }

function mostrar_hora(ctx) {
var d = new Date();
var h = d.getHours();
var m = d.getMinutes();
var s = d.getSeconds();
$('#tex').html(h + ":" + m + ":" + s);
ctx.clearRect(0,0,140,140) // borrar CANVAS
myCircle(ctx,80,80,50,3,"black"); // dibujar esfera del reloj
myLine(ctx,80,80,x2(h,12,80,30),y2(h,12,80,30),5,"grey"); // horas
myLine(ctx,80,80,x2(m,60,80,40),y2(m,60,80,40),3,"grey"); // min.
myLine(ctx,80,80,x2(s,60,80,50),y2(s,60,80,50),1,"red"); // seg.
}

$(function() {
var c=document.getElementById("myCanvas"); // obtiene CANVAS
if (c.getContext) { // CANVAS soportado?
var ctx = c.getContext("2d"); // define contexto 2D
mostrar_hora(ctx);

setInterval(function(){mostrar_hora(ctx);}, 1000)
}
})
</script>
</head>
<body>
<h3> Reloj CANVAS</h3>
<div id="tex">texto</div>
<canvas id="myCanvas" width="140" height="140"></canvas>
</body>
</html>

```



## CANVAS: Reloj animado

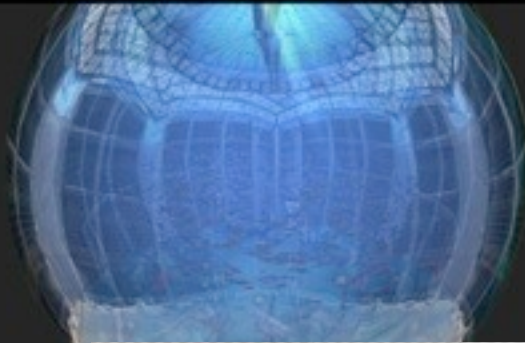


# WebGL: Web en 3D

<http://www.chromeexperiments.com/webgl>

## WebGL Experiments

WebGL is a new web technology that brings hardware-accelerated 3D graphics to the browser without installing additional software.



### WebGL Experiments

Most Recent | Highest Rated | 3



[\(B\) WebGL test](#)

Saku Tiainen

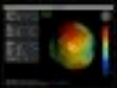
★★★★



[T04](#)

Paulo Falcao

★★★★



[Genod Viewer](#)

Andrea Gatti

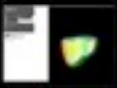
★★★★



[The Wobble Dance](#)

thierry franchina

★★★★



[Heart Browser](#)

Jay Amin

★★★★★



[Nebula](#)

Felix Turner

★★★★





# Audio, Video e iFrames en HTML5

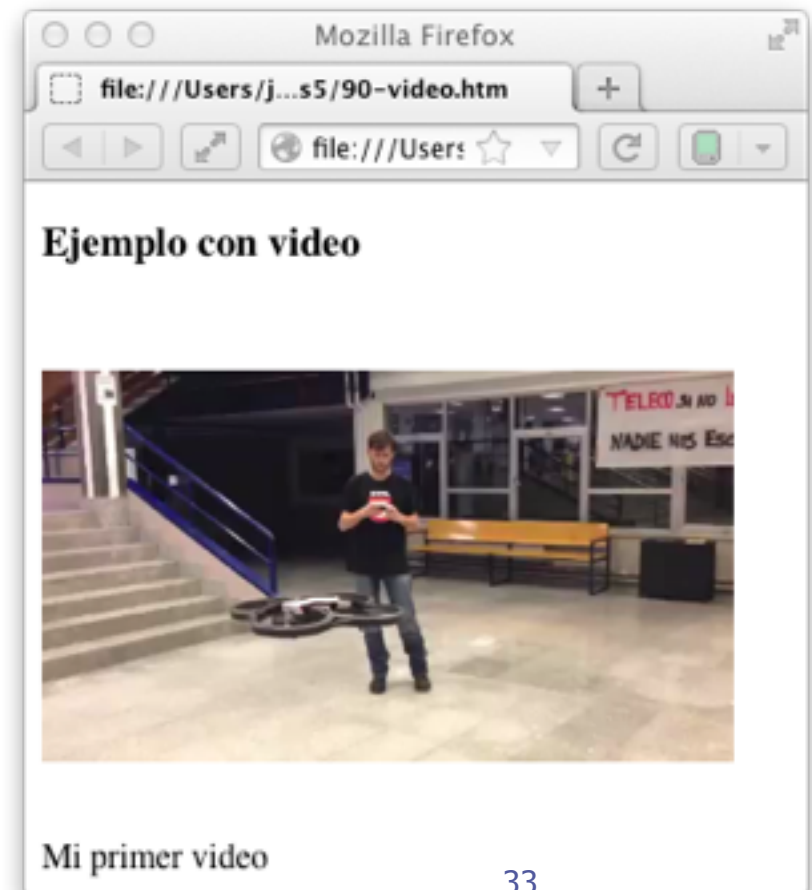


# Audio y Video en HTML5

- ◆ Las marcas audio y video de HTML5
  - Permiten incluir audio y video en páginas Web
    - ◆ [http://www.w3schools.com/html/html5\\_video.asp](http://www.w3schools.com/html/html5_video.asp)

```
<!DOCTYPE html>
<html>
  <body>
    <h3>Ejemplo con video</h3>

    <video src="CuadricopterM.webm"
           width="320" height="240"
           controls
           preload
           autoplay>
    </video><br>
    Mi primer video
  </body>
</html>
```



# Video: formatos

- ◆ Contenedor **OGG**
  - Video: **Theora** (VP7), Audio: **Vorbis**
    - ◆ Calidad menor

Ogg Theora



- ◆ Contenedor **MP4**
  - Video: **H264**, Audio: **ACC**
    - ◆ Existen Patentes

MP4 / H.264



- ◆ Contenedor **WebM**
  - Video: **VP8**, Audio: **Vorbis**

WebM



```
<video width="320" height="240" controls preload autoplay>  
  <source src="CuadricopterM.m4v" type=video/mp4>  
  <source src="CuadricopterM.webm" type=video/webm>  
</video><br>
```

# 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 malware
    - ◆ Acceso JavaScript limitado a caja de arena

```
<!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 **páginas** en otros “frames” que **proviene** del mismo origen
  - **Esto evita en el ejemplo anterior que el juego**
    - ◆ Y 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 a las páginas Web
    - ◆ Los scripts JavaScript 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



Final del tema  
Muchas gracias!

