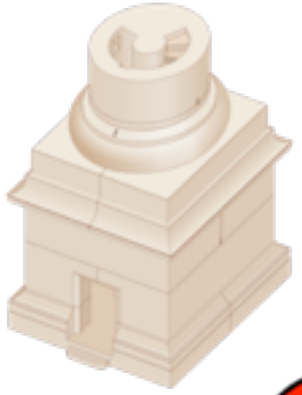
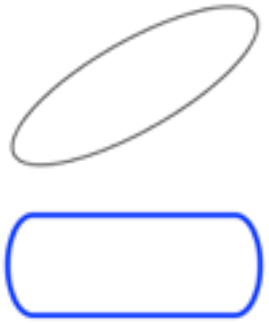
HTML5 SVG: Scalable Vector Graphics

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"

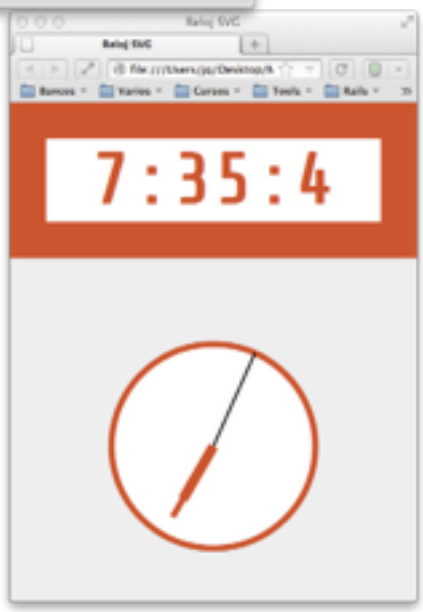
- **"Ajuste SVG"** ilustra como reescalar una imagen SVG
  - Las imagenes en SVG reescalan sin perder calidad
    - porque son gráficos vectoriales
    - tutorial: http://www.w3schools.com/svg/
  - Las imágenes GIT, JPEG o PNG no reescalan bien
    - porque tienen una resolución determinada
- Esta WebApp 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 "-"

```
<!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"

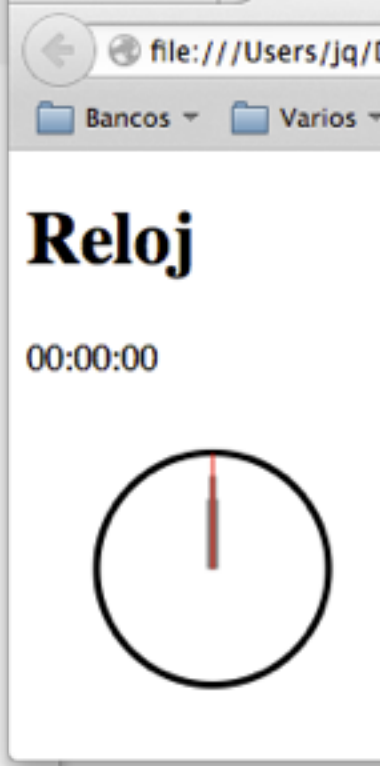
- **"Reloj SVG"** genera un reloj sencillo con SVG
  - El reloj se compone de
    - Un círculo negro
    - Tres lineas para las manecillas del reloj
- SVG puede animarse con javaScript
  - modificando la representación DOM del reloj
    - Versión 1: las manecillas se mueven con transform
      - https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/transform
    - 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

```
<head><title>Galeria</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

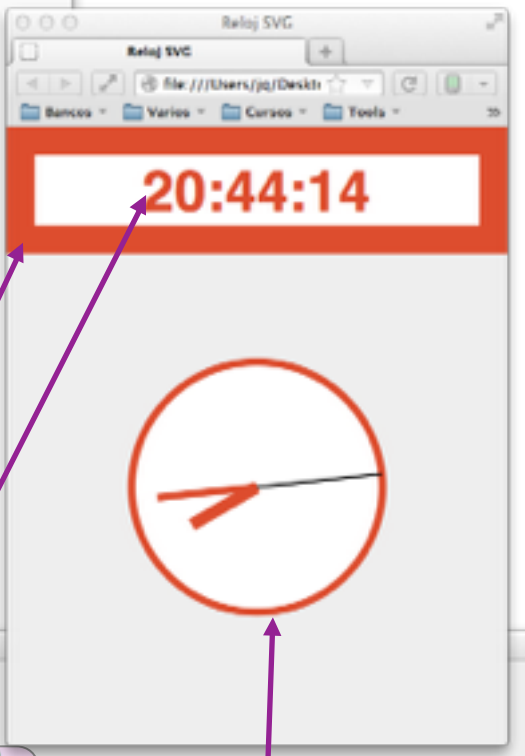
```
<!DOCTYPE html>
<html>
<head>
  <title>Reloj SVG</title>
  <meta charset="UTF-8">
  <script type="text/javascript" src="http://zeptojs.com/zepto.min.js" >
  </script>

<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/0B48KCWfVwClEMjFhUHM4d3FnSTg/09-clock_CSS.htm
    - https://googledrive.com/host/0B48KCWfVwClEMjFhUHM4d3FnSTg/10_clock_CSS_a.html
    - https://googledrive.com/host/0B48KCWfVwClEMjFhUHM4d3FnSTg/10_clock_CSS_b.htm
    - https://googledrive.com/host/0B48KCWfVwClEMjFhUHM4d3FnSTg/10_clock_CSS_c.htm
    - https://googledrive.com/host/0B48KCWfVwClEMjFhUHM4d3FnSTg/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>
```



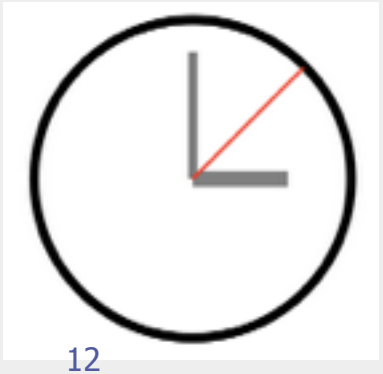

```
.......
<body>
<div class="mitad">
  <div id="tex">texto</div>
</div>

<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>
```



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
  - http://www.w3schools.com/html/html5_canvas.asp
  - http://www.w3schools.com/tags/ref_canvas.asp
  - https://developer.mozilla.org/En/Canvas_tutorial

Ejemplo "Reloj CANVAS"

- **"Reloj CANVAS"** es similar al reloj programado con SVG
  - Pero programado en el canvas
    - Tiene el mimos circulo 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

```html
<!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 línea
    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 línea del circulo
    ctx.lineWidth = width;        // anchura de la línea 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>
```

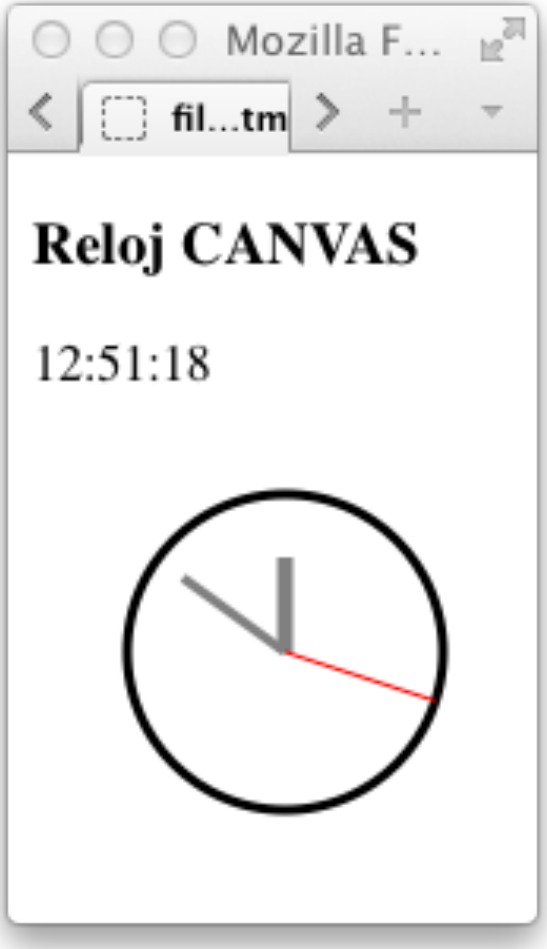
Reloj CANVAS

Animar las manecillas del reloj

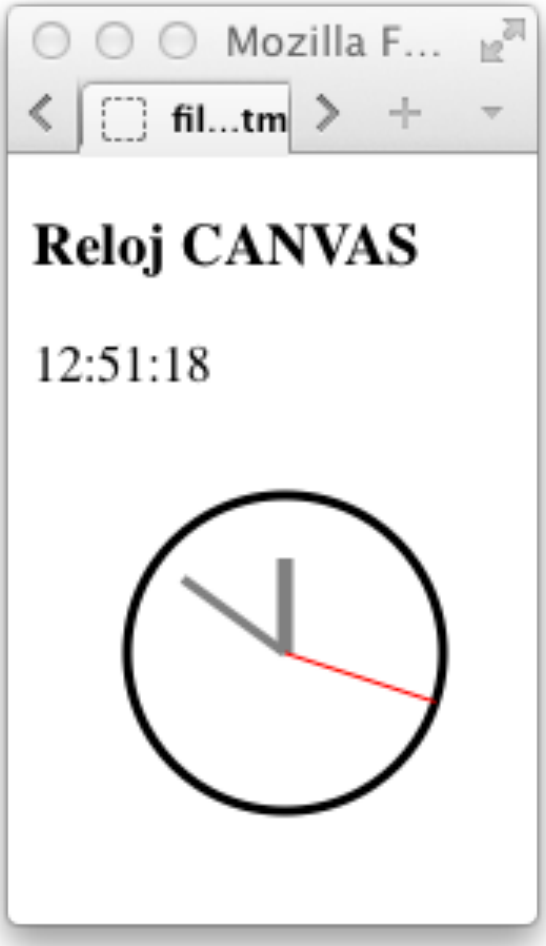
- El script calcula cada segundo las manecillas
  - una vez calculadas borra el canvas
    - y vuelve a dibujar el reloj completo
  - Secundero (50 px), minutero (40 px), hora (30 px)

- 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



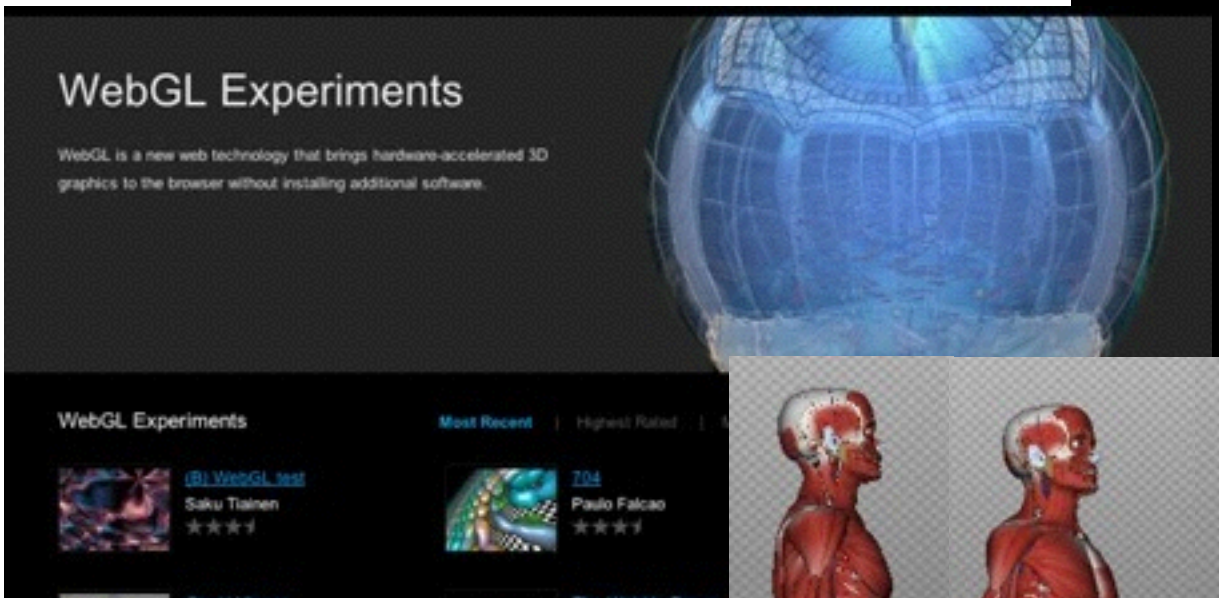
```
<!DOCTYPE html>
<html>
 <head>
  <script type="text/javascript" src="zepto.min.js" ></script>
  <script type="application/javascript">
    function x2(n,i,x1,r) {return x1 + r*Math.sin(2*Math.PI*n/i);};
    function y2(n,i,y1,r) {return y1 - r*Math.cos(2*Math.PI*n/i);};

    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>
```

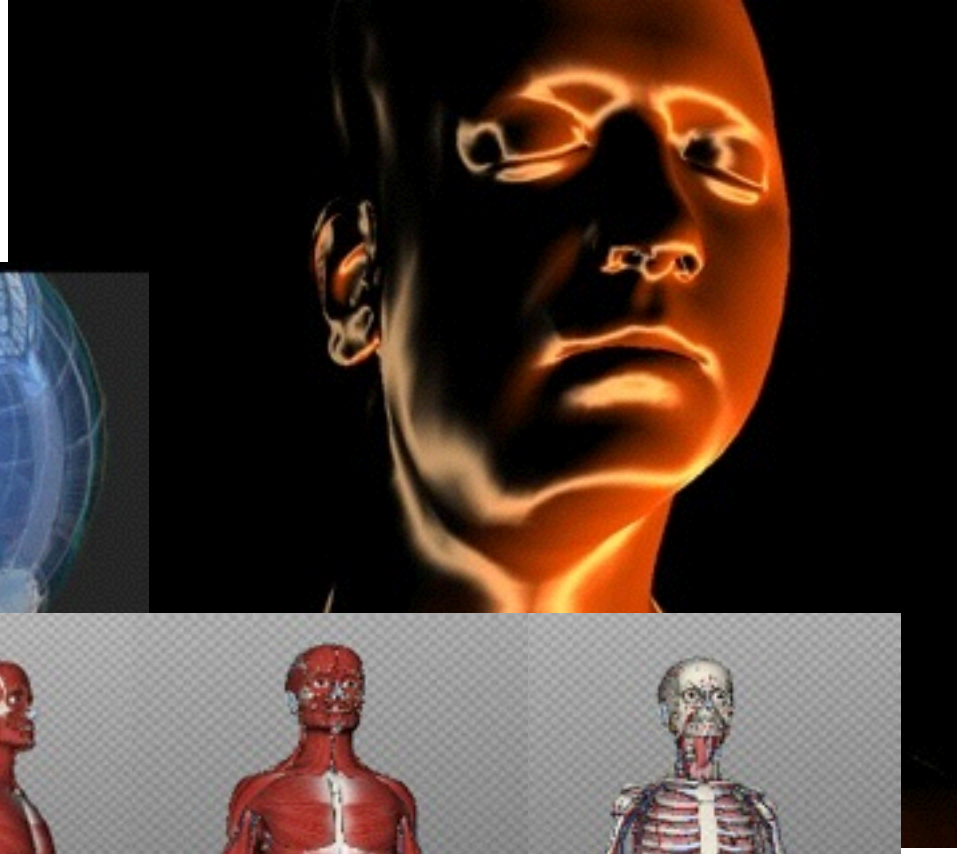


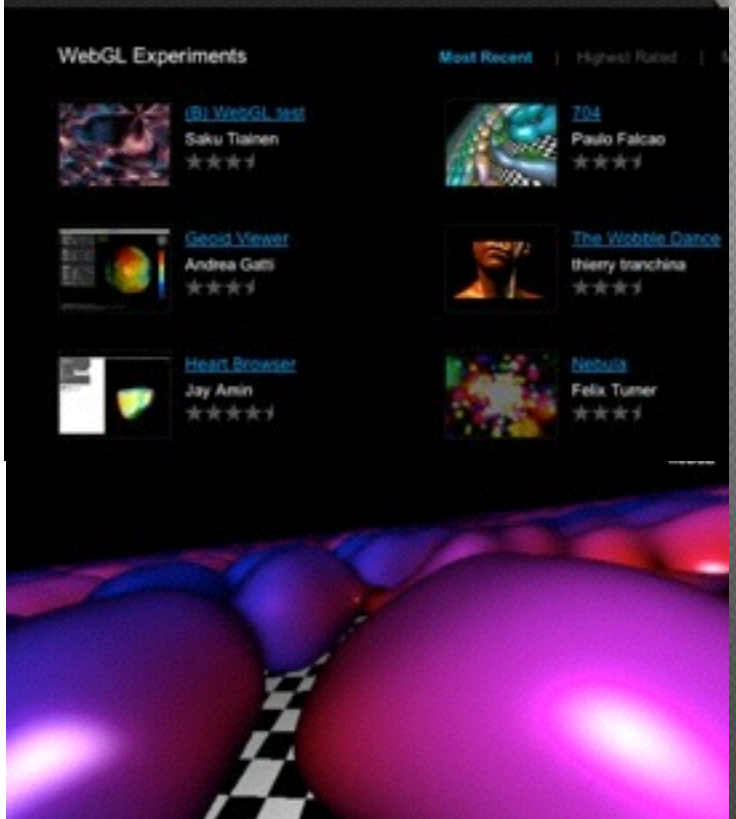
CANVAS: Reloj animado

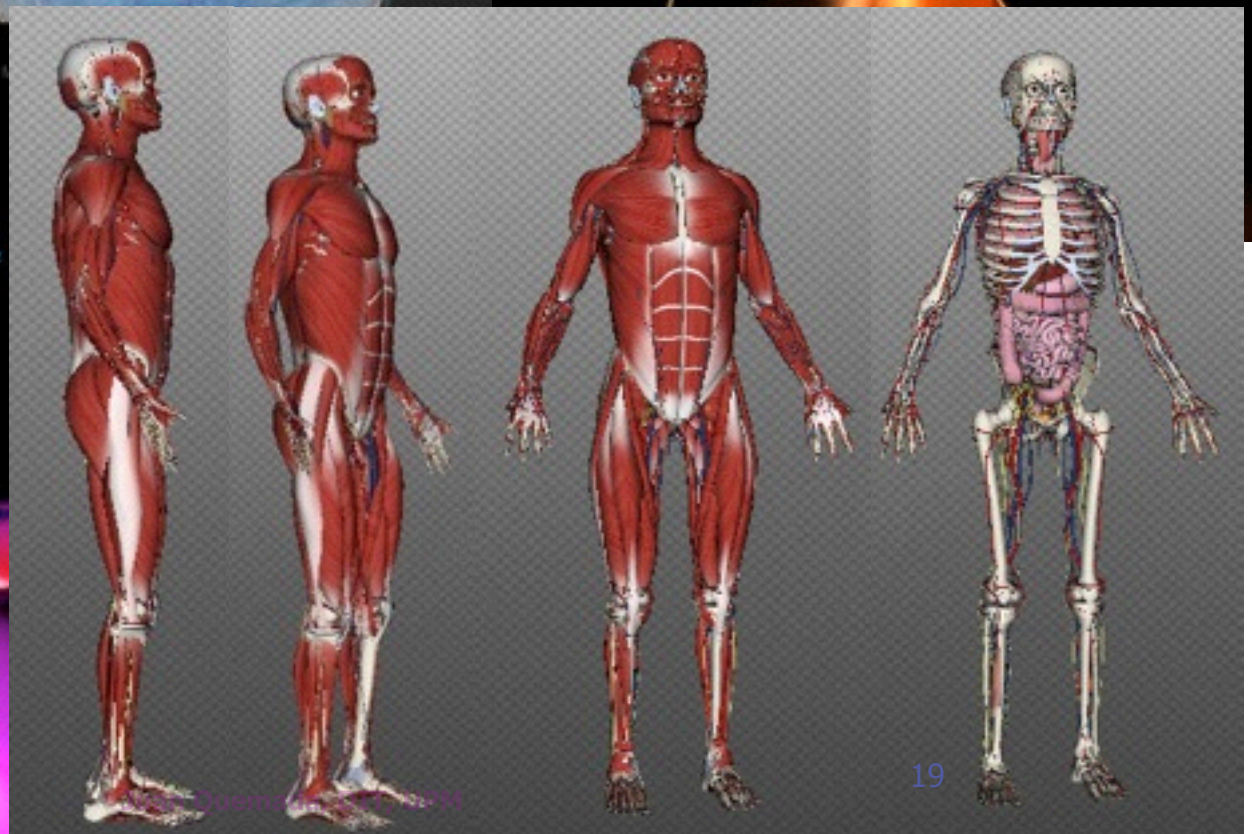
WebGL: Web en 3D

http://www.chromeexperiments.com/webgl









Aplicaciones FirefoxOS

FirefoxOS

- FirefoxOS es el nuevo SO
  - para móviles y tabletas
    - Desarrollado por la Fundación Mozilla
- Sus aplicaciones se programan en
  - HTML5, CSS y Javascript
- Recursos y tutoriales
  - https://marketplace.firefox.com/developers/docs/firefox_os
  - https://marketplace.firefox.com/developers/docs/quick_start
  - https://developer.mozilla.org/es/docs/Aplicaciones/Comenzando_aplicaciones
  - https://developer.mozilla.org/en-US/Apps/Reference

Apps FirefoxOS

- Aplicaciones para instalar en móviles y tabletas
  - Funcionando con o sin conexión a Internet
    - Construidas con tecnologías Web: URLs, HTTP, HTML, CSS, JavaScript, ...
  - También pueden instalarse en PCs o navegadores

- Tipos de WebApps Firefox OS
  - **hosted:** alojadas en un servidor
  - **packaged:** se descargan empaquetadas
    - Hay tres tipos
      - **Plain**: aplicaciones Web convencionales
      - **Certified**: con acceso a algunos elementos restringidos
      - **Privileged**: con acceso a todos los recursos del móvil

Hosted Apps

- Webapps publicadas en modo ejecutable en un servidor Web
  - Pueden ejecutarse desde un navegador
  - Además de ser publicadas en una tienda a través de su URL
    - para su instalación en un dispositivo FirefoxOS
  - Mas info:
    - https://developer.mozilla.org/en-US/Marketplace/Options/Hosted_apps

- Características
  - Necesitan conexión a Internet para ser ejecutadas
  - Solo puede haber una hosted WebApp por dominio Web

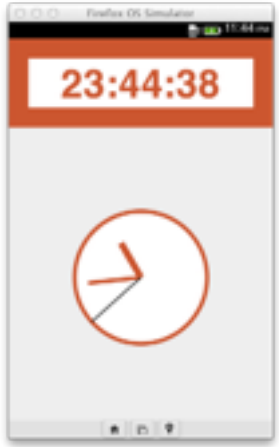
- Se gestionan a través del objeto predefinido **navigator.mozApps**
  - con métodos **install(....)** o **checkInstalled(....)**

Packaged WebApps

- Webapps empaquetadas en un fichero ZIP
  - Pueden ser publicadas en una tienda
    - para su instalación en dispositivos Firefox OS
  - Mas info:
    - https://developer.mozilla.org/en-US/Marketplace/Options/Packaged_apps
- Se pueden publicar en tiendas o en servidores
  - Solamente se pueden instalar
    - No se pueden ejecutar directamente en un navegador
  - Pueden ser autonomas
    - Funcionando sin conexión a Internet
- Se instalan a través del objeto predefinido **navigator.mozApps**
  - con método **installPackage(....)**

Firefox Marketplace

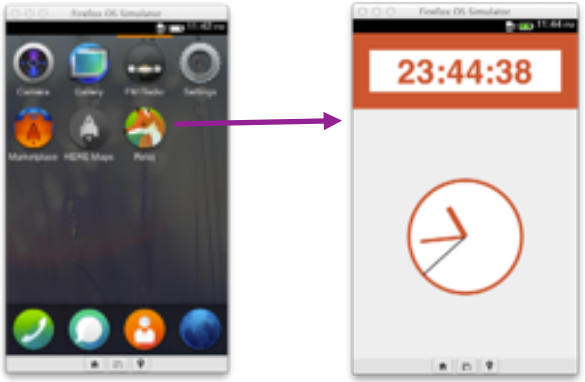
- Firefox Marketplace permite crear
  - tiendas de aplicaciones alojadas en servidores
    - https://developer.mozilla.org/en-US/Marketplace
- Los Marketplace FirefoxOS pueden competir entre si
  - serán más o menos populares
    - dependiendo de la aceptación de los usuarios
  - serán más o menos seguras
    - dependiendo del control de la seguridad realizado
- Market places con apps de Mozilla
  - https://marketplace.firefox.com



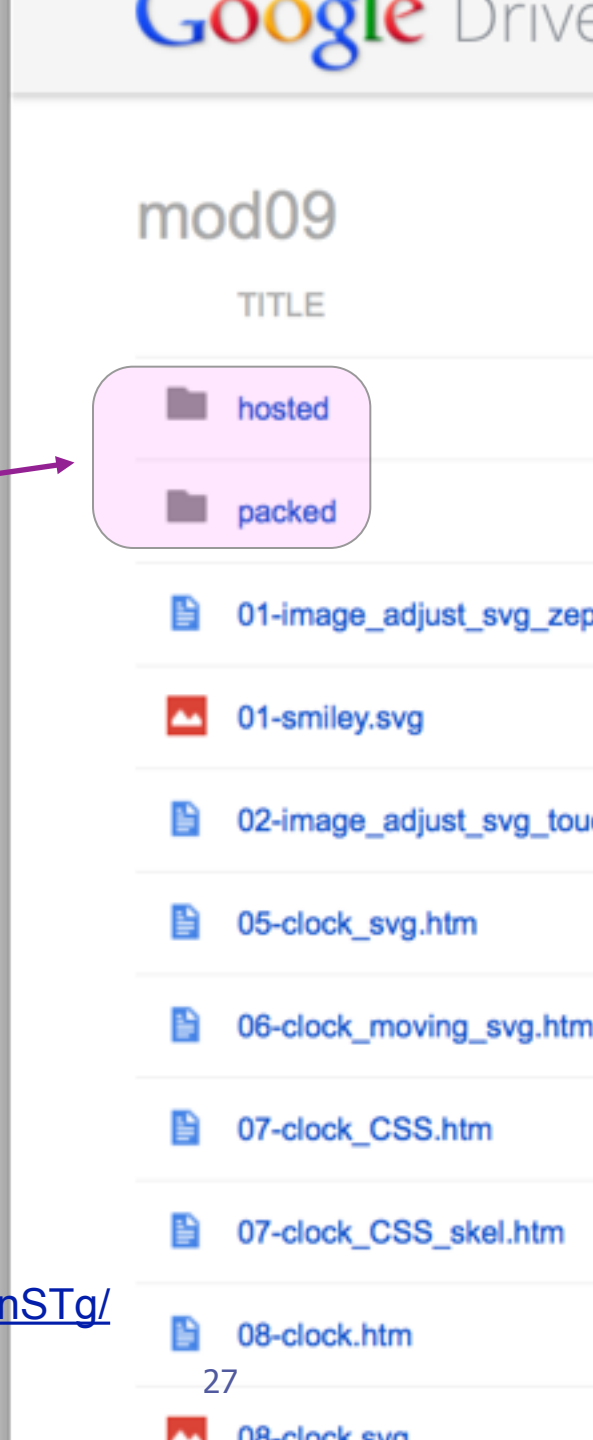
Reloj SVG como App FirefoxOS

Reloj SVG como App

- Incluimos el reloj SVG como
  - **hosted App**
  - **packaged App**

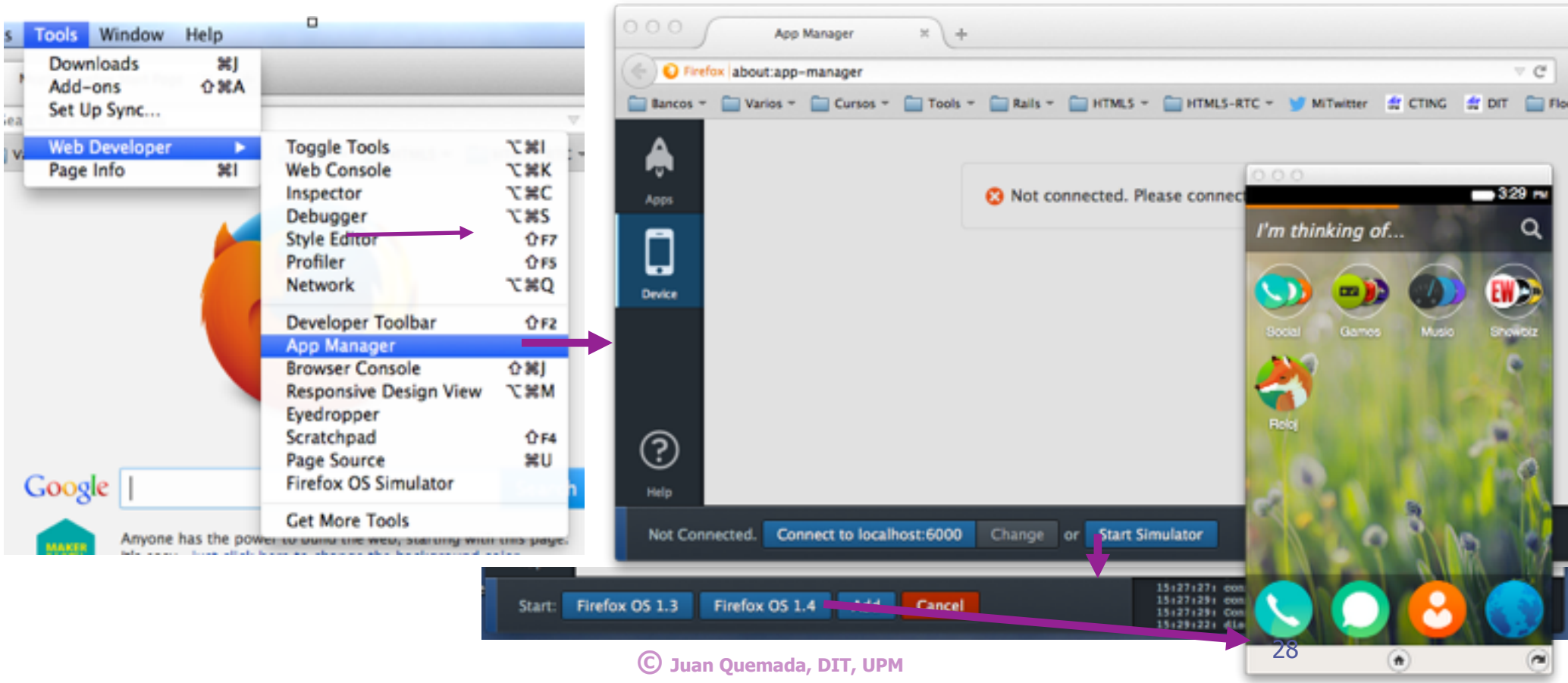


- Ambas están publicadas en
  - Google Drive de asignatura
    - https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/



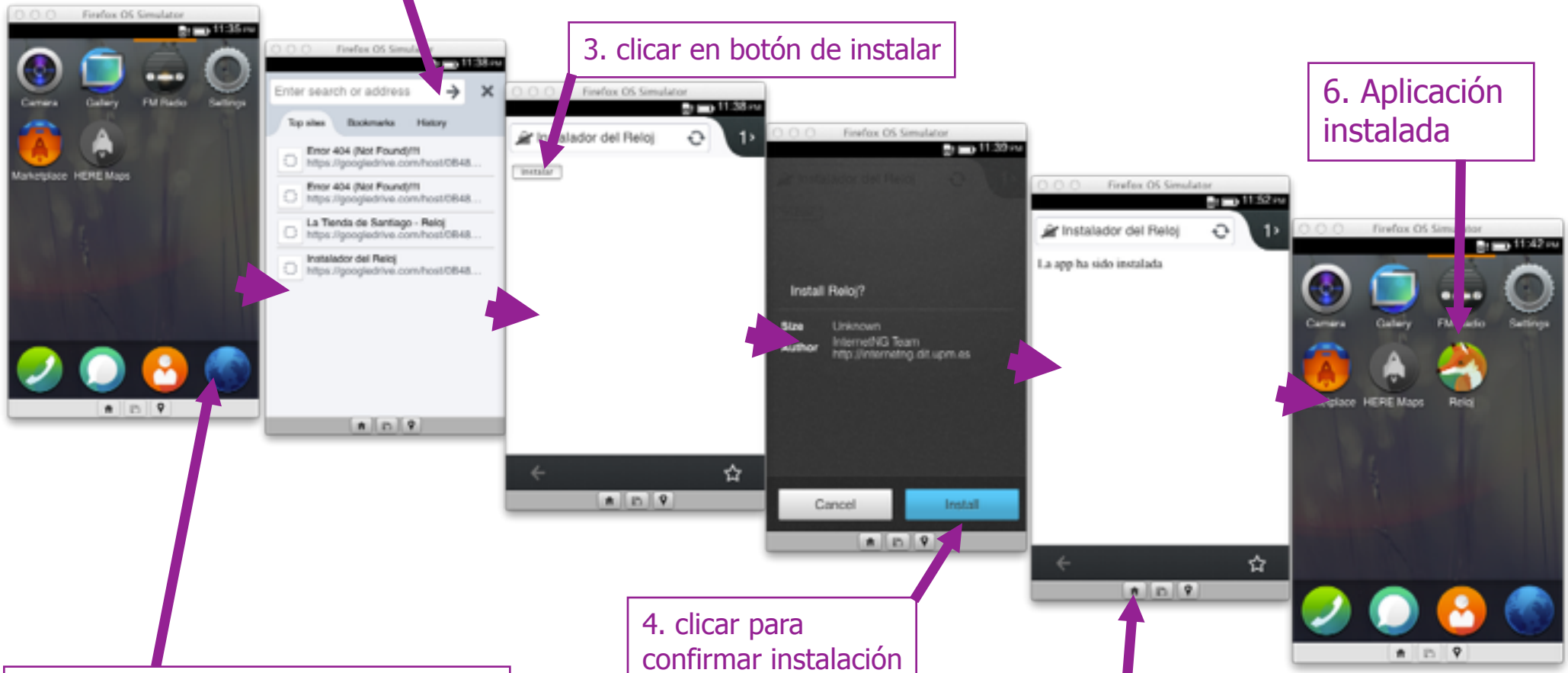
Arrancar el simulador de FirefoxOS en Firefox

- Las apps se instalan en el simulador incluido en Firefox
  - Firefox incluye a partir de Firefox 30.0 un simulador de FirefoxOS
    - Se arranca seleccionando **"Tools -> Web Developer -> App Manager"**
      - Después se arranca (**Start Simulator)** y se selecciona la versión 1.4 (**Firefox 1.4**)
    - **OJO!** La app con el Reloj dada está probada en Firefox 1.4 y 1.3 y pueden no funcionar en otras versiones



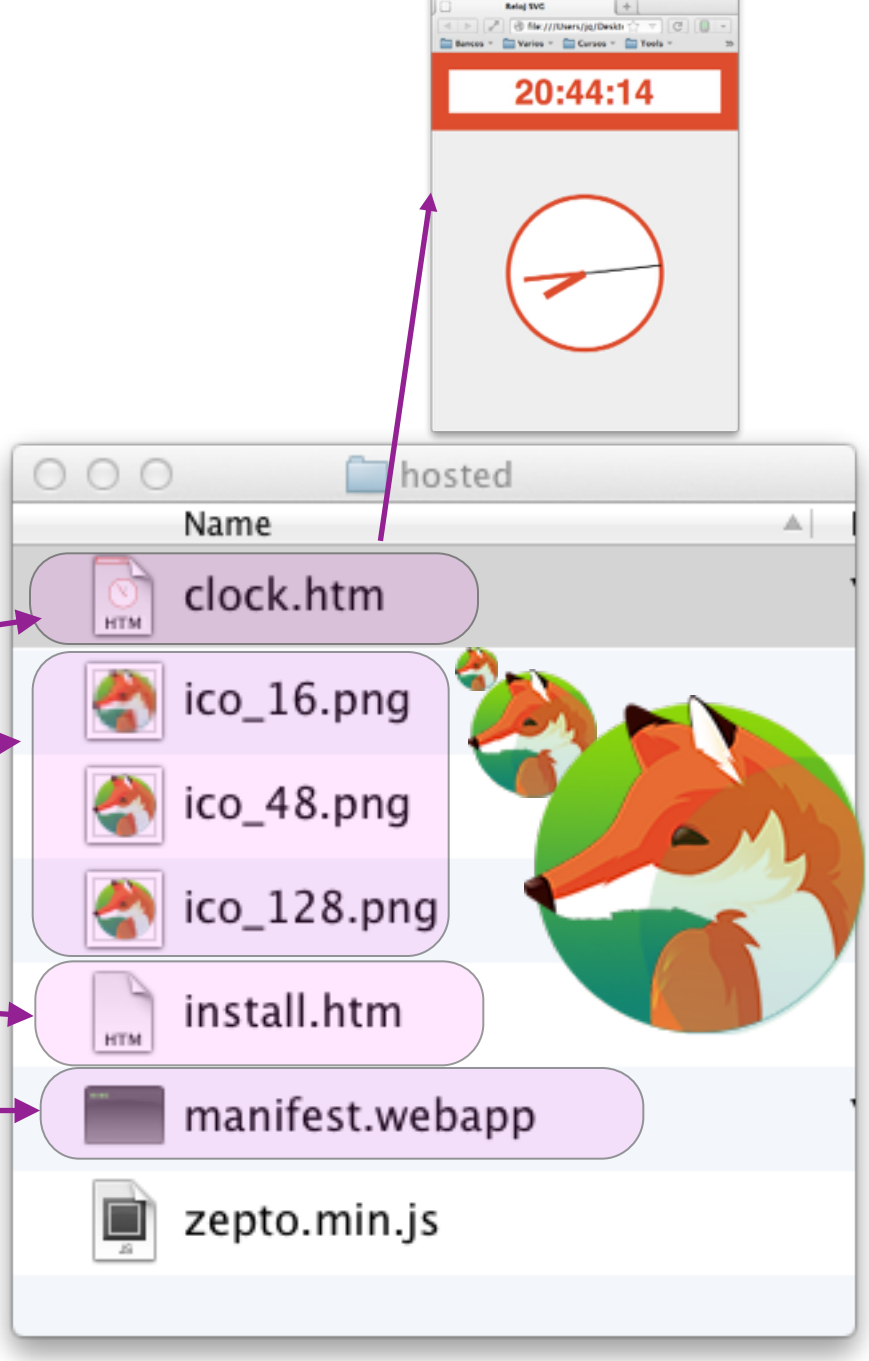
Instalar hosted App en simulador FirefoxOS

2. Introducir URL del instalador en el navegador y clicar (ejecutar instalador)
https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/hosted/install.htm



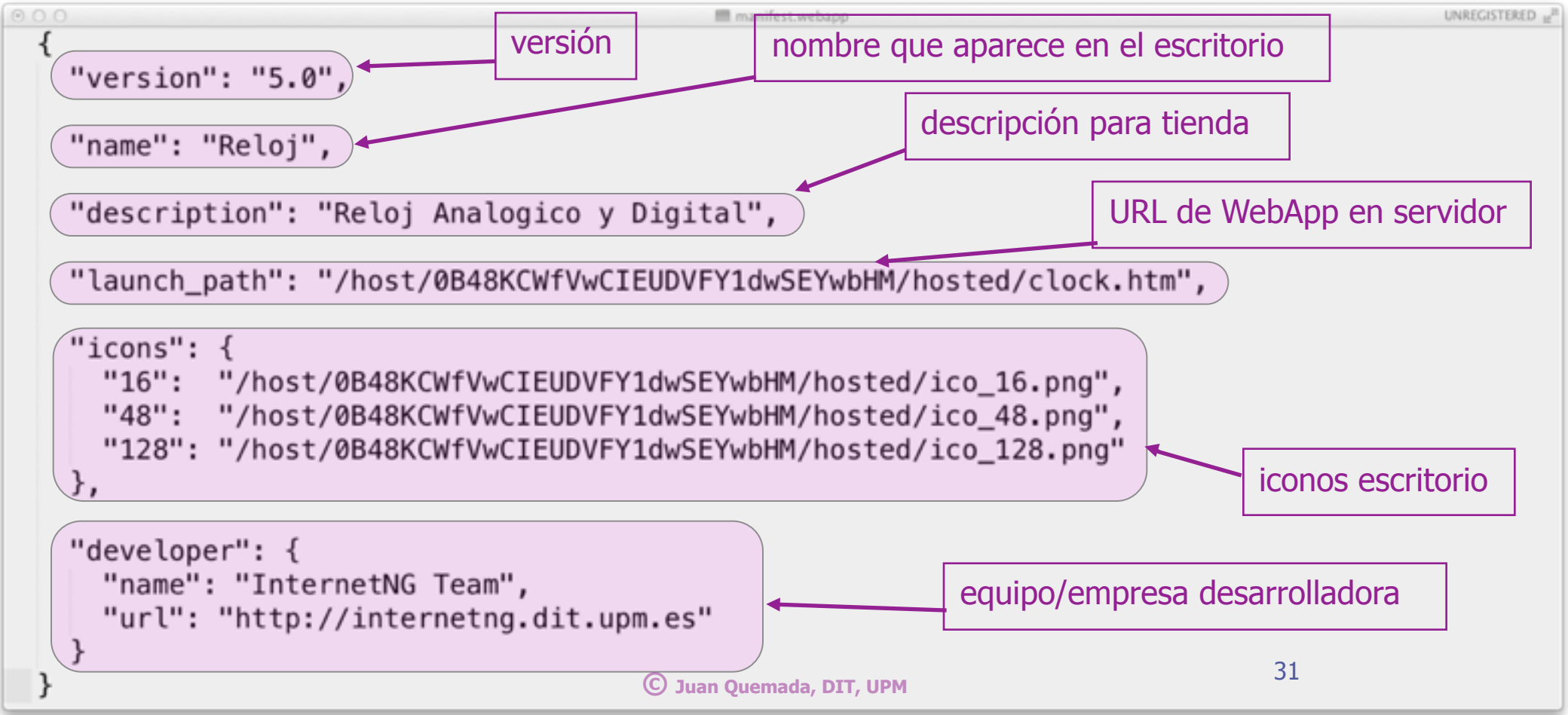
Reloj SVG: hosted App

- Para crear una hosted App con Reloj SVG se debe añadir a **clock.htm**:
  - **Iconos**
    - que se mostrarán en el escritorio
  - **Instalador**
    - de la WebApp en el escritorio
  - **Manifiesto**
    - con información de instalación
  - El ejemplo suministrado incluye además estilos y tipografías



manifest.webapp

- Fichero en formato JSON (JavaScript Object Notation)
  - Con información de instalación, recursos, diseñador, …
    - Los URLs son relativos con path absoluto
- JSON es un formato muy habitual para manifiestos

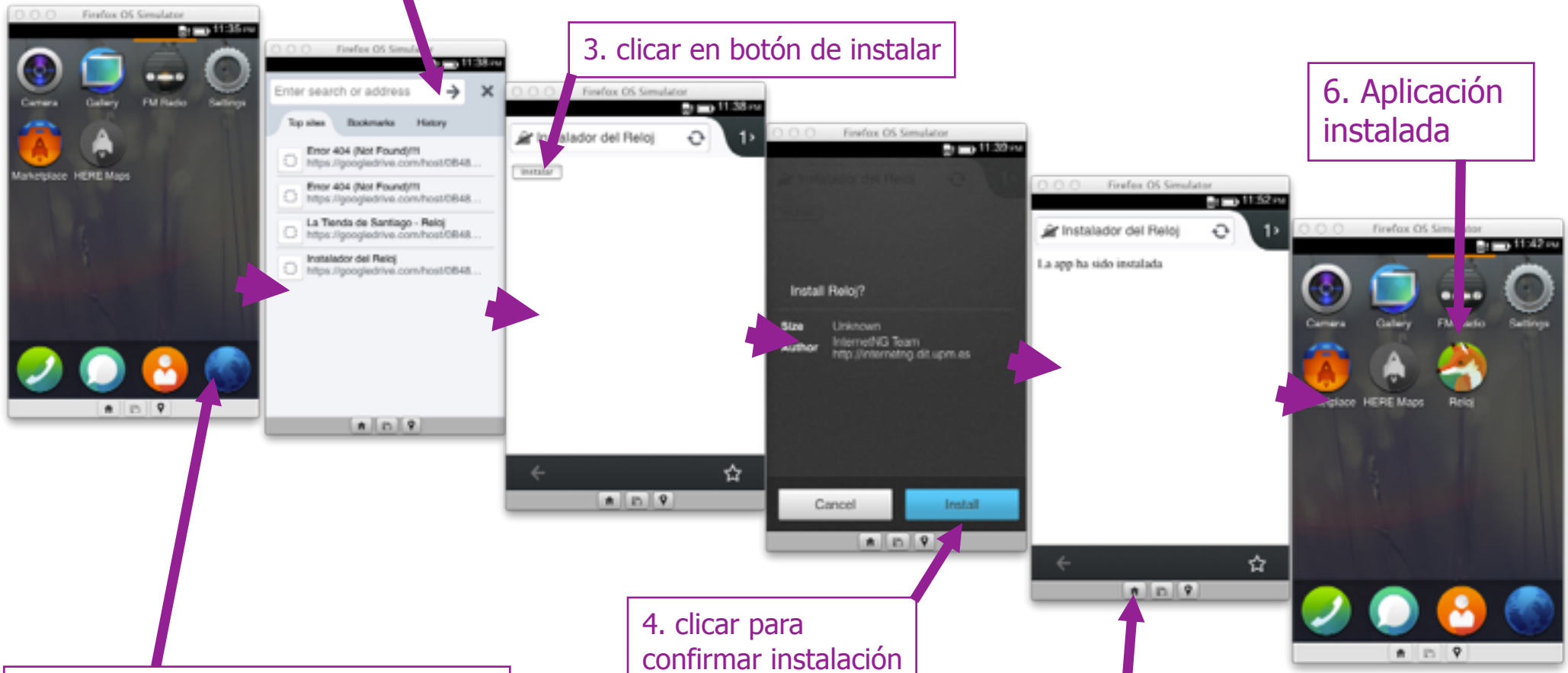


install.html

```
<!DOCTYPE html>
<html><head><title>Instalador del Reloj</title>
<meta charset="UTF-8">
<script type="text/javascript" src="http://zeptojs.com/zepto.min.js" ></script>
</head>
<body>
  <div id="install">Estado de la instalación del Reloj 5.0</div>
<script>
    $(document).ready(function() {
      var gManifestName = location.href.replace("install.htm","") + 'manifest.webapp';
      var request = navigator.mozApps.checkInstalled(gManifestName);

      request.onsuccess = function() {
        if (request.result) { $("#install").text("La app ya esta instalada");  // App instalada
        else {                                                                 // App no instalada
          $("#install").html("<input type='button' value='Instalar'>");

          $("#install").click(function() {
              var req = navigator.mozApps.install(gManifestName);
              req.onsuccess = function(data) {
                  $("#install").text("La app ha sido instalada").unbind('click');
              }
              req.onerror = function(errObj) {
                  alert("Error de instalacion: " + this.error.name);
              }
          });
        }
      }
      request.onerror = function() { alert('Error de instalacion: ' + this.error.message); }
    });
</script>
</body>
</html>
```

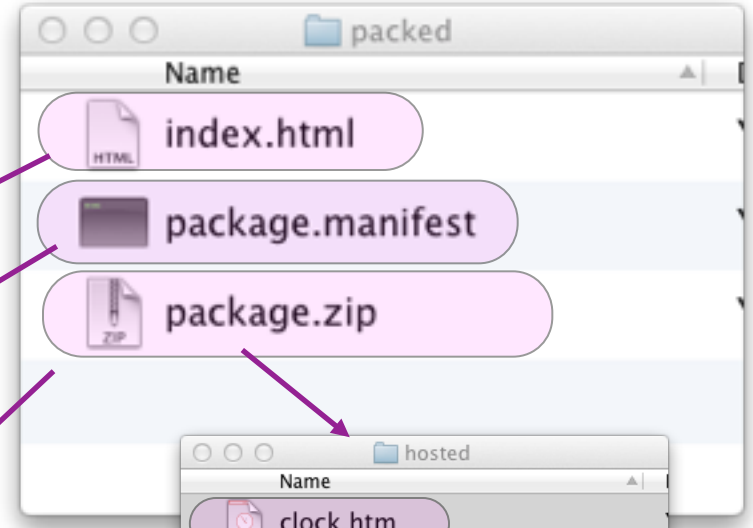
Instalar hosted App en simulador FirefoxOS

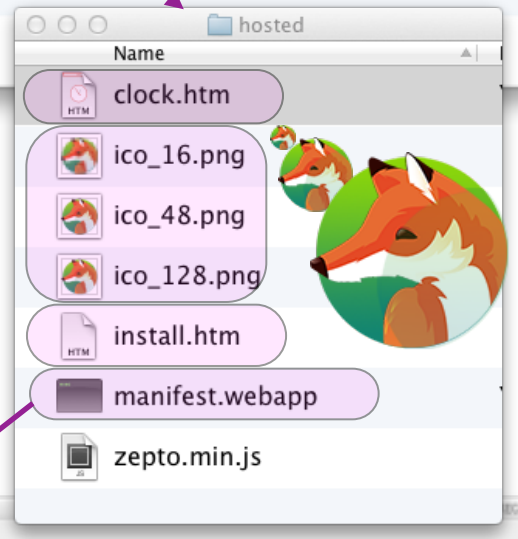
2. Introducir URL del instalador en el navegador y clicar (ejecutar instalador)
https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/hosted/install.htm

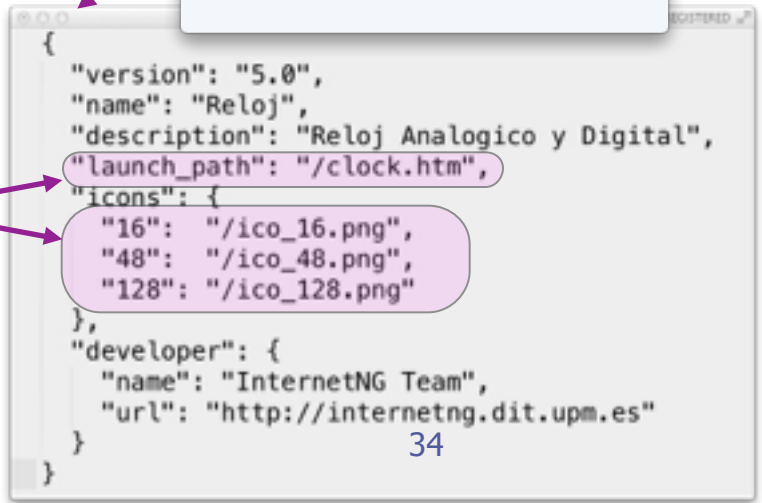


Reloj SVG: packaged App

- Componentes
  - **Instalador**
    - de la WebApp en el escritorio
  - **Manifiesto**
    - con información de instalación del paquete
  - **App empaquetada**
    - **Zip** con todos los ficheros de la App
      - los mismos ficheros de la hosted App
        - Salvo manifiesto: URLs relativos
    - **OJO**: Los ficheros deben empaquetarse sin más, sin estar en ningún directorio

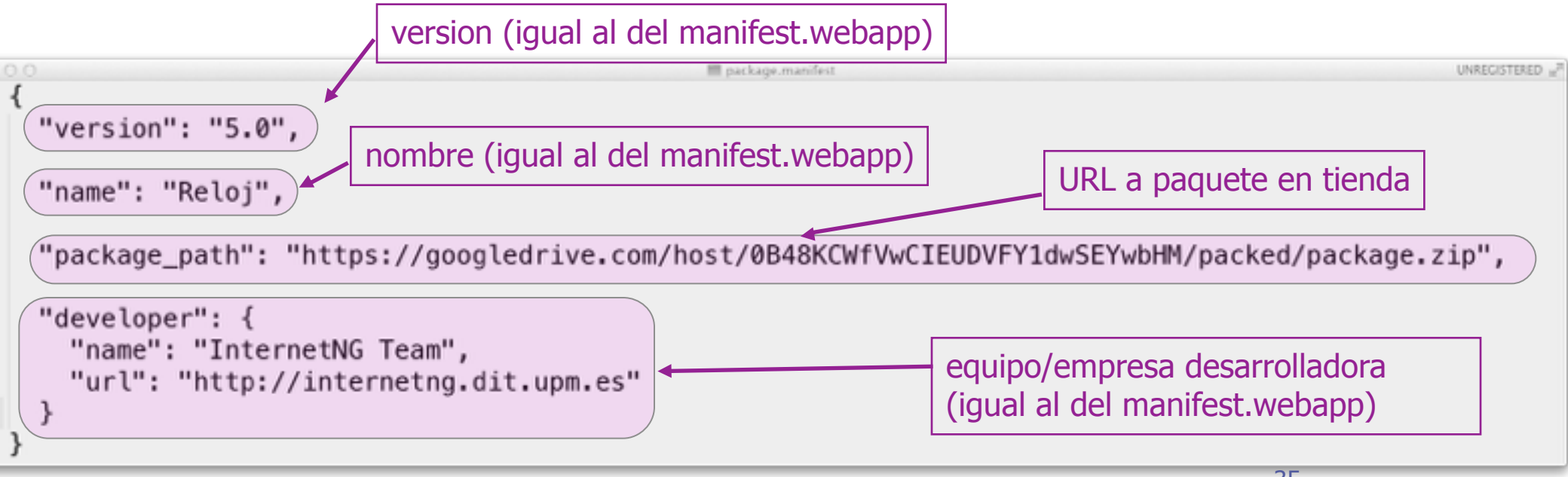






package.manifest

- Fichero en formato JSON (JavaScript Object Notation)
  - Con informacion sobre la webapp empaquetada
- JSON representa datos como literales de objetos JavaScript
  - Todos los datos están serializados en el fichero como un string



index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>La Tienda de Santiago - Reloj</title>
  <meta charset="UTF-8">
</head>

<body>
    <p>Página de Auto-Instalación del Reloj 5.0</p>

    <script>
      (function() {
        var manifestUrl = location.href.replace("index.html","") +'package.manifest';

        if (! navigator.mozApps.installPackage) {
          alert("ERROR: Esta aplicación no es compatible con tu dispositivo.");
          return;
        }

        var req = navigator.mozApps.installPackage(manifestUrl);
        req.onsuccess = function() {
          alert("Instalacion completada."+this.result.origin);
        };
        req.onerror = function() {
          alert("Error de instalación: "+this.error.name);
        };
      }
      )();
    </script>
</body>
</html>
```

Instalar packed App en simulador FirefoxOS

2. Introducir URL del instalador en el navegador y clicar (ejecutar instalador)
https://googledrive.com/host/0B48KCWfVwCIEMjFhUHM4d3FnSTg/packed/

3. clicar en botón de instalar

6. Aplicación instalada

1. clicar para desplegar el navegador y poder instalar la hosted App

4. clicar para confirmar instalación

4. clicar para volver a escritiorio