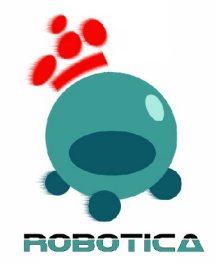
LABORATORIO TECNOLOGÍAS AUDIOVISUALES EN LA WEB

José María Cañas Plaza

jmplaza@gsyc.es



Grado Sistemas Audiovisuales y Multimedia
curso 2015-2016

HTML5

Contenido

- Introducción
- GeoLocalización
- `WebWorkers`
- Funcionamiento fuera de línea, `AppCache`
- Almacenamiento local, `LocalStorage`
- `Drag & Drop`

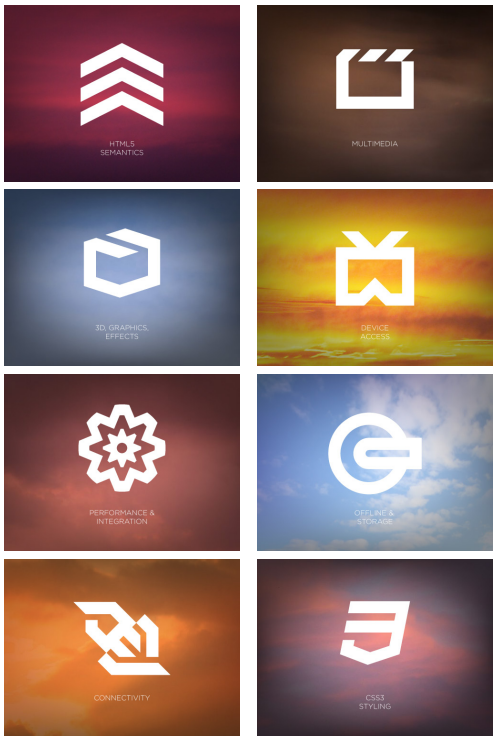
Introducción

- Nuevo estandard documentos web 2014/10
- Nuevas necesidades, más funcionalidad
- Nuevas etiquetas y APIs
- Diferente a Flash de Adobe, aunque ambos se usan para audio y video
- Trabajo conjunto de *Web Hypertext Application Technology Working* Group (WHATWG) y el *World Wide Web Consortium* (W3C) (2007)
  - WHATWG empezó en 2004, buscando compatibilidad con lo anterior
  - W3C estaba enfocado en XHTML 2.0.

- Soporte creciente en los navegadores:
  Chrome, Firefox, Safari, Opera, IE
- Adopción 2010/04 Apple, 2011/11 Adobe
- Standard vivo
- Neutralidad en la red
- Comprobar primero en JavaScript si el navegador soporta el API

Colección de estándares

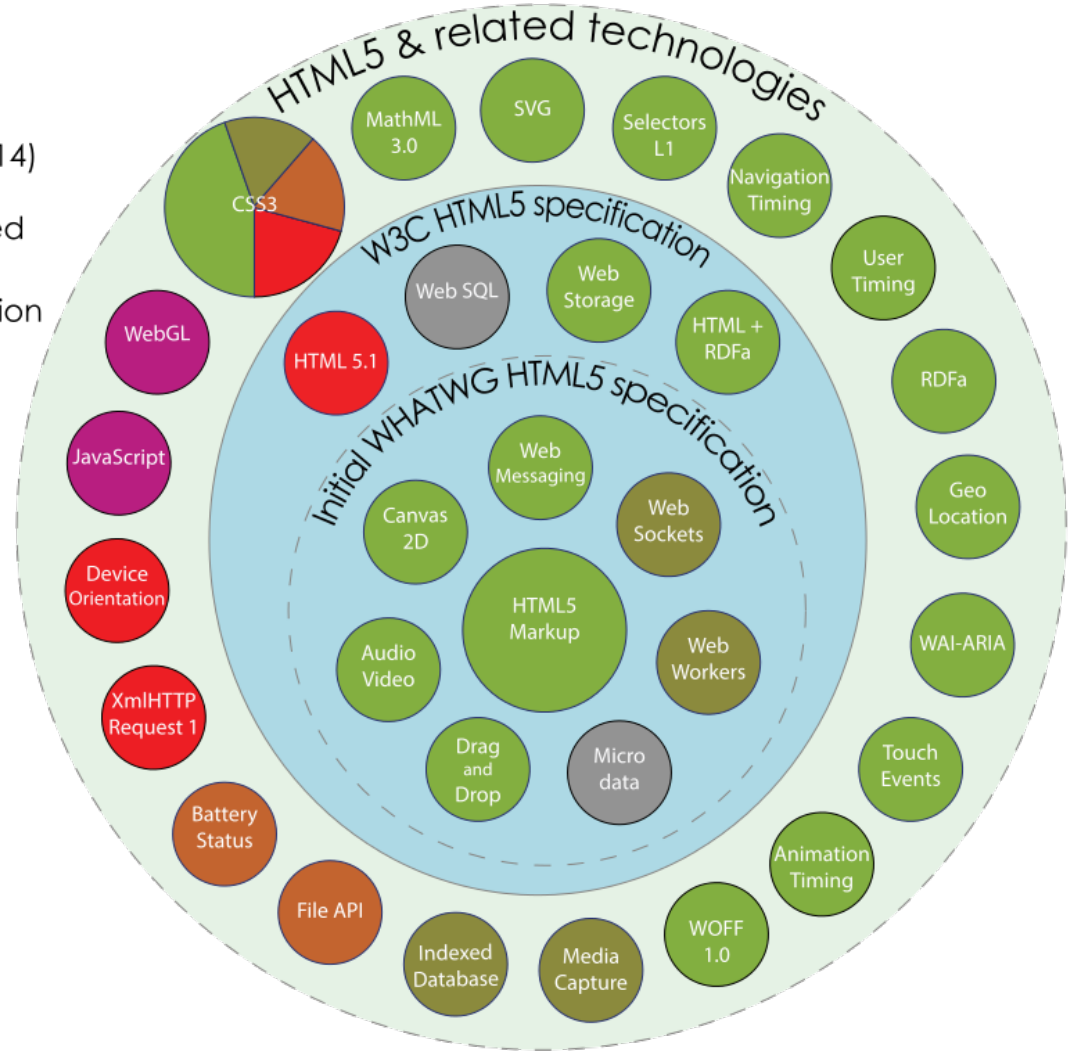
- Semantics: nuevas etiquetas, estructura
- Multimedia: Audio y Video
- 3D Graphics & effects: SVG y Canvas, WebGL
- *Device Access: GeoLocation*
- *Performance & Integration: WebWorkers, AJAX-2*
- *Offline & Storage: AppCache y LocalStorage*
- Connectivity: WebSockets, ServerSideEvents
- CSS3



HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



Interfaz de Geolocalización

- Solicita siempre permiso al usuario
- Fuentes reales de datos de posición:
  - Dirección IP
  - MAC de Wifi y Bluetooth
  - Sensor GPS
  - ID de celda GSM/CDMA
- Se devuelve la posición con cierta precisión
- `navigator.geolocation.getCurrentPosition`
- `navigator.geolocation.watchPosition`

```
function geoFindMe(){
      if(navigator.geolocation){
          navigator.geolocation.getCurrentPosition(success, error, geo_options);
      }else{
          alert("Geolocation services are not supported by your web browser.");
      }

function success(position) {
 var latitude = position.coords.latitude;
 var longitude = position.coords.longitude;
 var altitude = position.coords.altitude;
 var accuracy = position.coords.accuracy;

 //do something with above position thing e.g. below
 alert('I am here! lat:' + latitude +' and long : ' +longitude );
}

function error(error) {
 alert("Unable to retrieve your location due to "+error.code + " : " + error.message);
 };

var geo_options = {
 enableHighAccuracy: true,
 maximumAge : 30000,
 timeout : 27000
 };
}
```

Objeto devuelto por getCurrentPosition

coords.latitude : latitud, número decimal

coords.longitude : longitud, número decimal

coords.accuracy : precisión, depende de la fuente

coords.altitude : altitud en metros respecto nivel del mar

coords.altitudeAccuracy

coords.heading : orientación en grados respecto el norte, horario

coords.speed : velocidad en m/s

timestamp : sello temporal de la respuesta

WebWorkers

- Modelo monohilo de las páginas: si se ejecuta un script la página no responde hasta que se termina
- Hilos JavaScript en segundo plano
- Los eventos del usuario humano no le interrumpen
- Concurrencia, paralelismo real, multinúcleo
- Tareas intensivas en CPU
- No pueden acceder al DOM, ni al objeto Window
- Se comunican mediante mensajes con el hilo principal

- Comprobar si hay soporte

```
if(typeof(Worker) !== "undefined") {
    // Yes! Web worker support!
    // Some code.....
} else {
    // Sorry! No Web Worker support..
}
```

- Crear un WebWorker: `var worker = new Worker("worker_script.js");`
- Terminar: `worker.terminate();`
- Enviarle un mensaje: `worker.postMessage("Hello World!");`
- Recibir un mensaje suyo:

```
//También con worker.addEventListener('message',function(e){...},false}
worker.onmessage = function(event){
    document.getElementById("result").innerHTML = event.data;
};
```

- `event.data` para el contenido del mensaje

- Desde el WebWorker se reciben y emiten mensajen también

```
self.addEventListener('message', function(e) {
  var data = e.data;
  switch (data.cmd) {
    case 'start':
      self.postMessage('WORKER STARTED: ' + data.msg);
      break;
    case 'stop':
      self.postMessage('WORKER STOPPED: ' + data.msg +
                       '. (buttons will no longer work)');
      self.close(); // Terminates the worker.
      break;
    default:
      self.postMessage('Unknown command: ' + data.msg);
  };
}, false);
```

- Los datos se copian, no se comparten
- Via JSON, cualquier objeto

Almacenamiento local (Web Storage)

- `LocalStorage`, persistente a cierres navegador
- `SessionStorage`, sólo para sesión
- Parecido a las Cookies
  - mayor tamaño ($> 5MB$ por dominio)
  - no se intercambian con servidor
  - el servidor no puede acceder
- Atributo-valor, ambos Strings

```
// Almacena un valor para la duración de la sesión
sessionStorage.setItem('key', 'value');

// Recuperación de un valor, se pierde al cerrar navegador
alert(sessionStorage.getItem('key'));

// Almacena un valor en navegador más allá de la sesión
localStorage.setItem('key', 'value');

// Recuperación de valor, persiste si cerramos y reabrimos navegador
alert(localStorage.getItem('key'));
```

- Sólo strings, pero con JSON se puede guardar indirectamente cualquier objeto

```
// Store an object instead of a string
localStorage.setItem('key', {name: 'value'});
alert(typeof localStorage.getItem('key')); // string

// Store an integer instead of a string
localStorage.setItem('key', 1);
alert(typeof localStorage.getItem('key')); // string

// Store an object using JSON
localStorage.setItem('key', JSON.stringify({name: 'value'}));
alert(JSON.parse(localStorage.getItem('key')).name); // value
```

Funcionamiento fuera de línea, AppCache

- El navegador copia datos (recursos) del servidor
  - La aplicación web funcione incluso sin conectividad
  - Las interacciones con servidor se reducen, sólo cambios
  - Los datos guardados cargan antes
- Fichero `Manifest`

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

- Tiene su propio tipo MIME

- Fichero de texto, especifica qué recursos cachear y qué no
- CACHE MANIFEST: estos descargan y están disponibles sin conexión
- NETWORK: ficheros aquí nunca se cachean
- FALLBACK: recurso de sustitución cuando no hay conexión

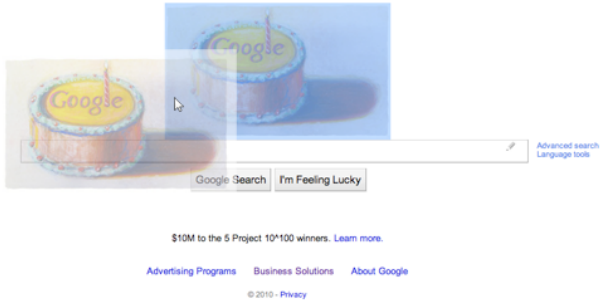
```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html/ /offline.html
/main.py /static.html
images/large/ images/offline.jpg
```

- Una vez cacheado el recurso, el navegador lo usa (aunque cambie en servidor)
- La caché de aplicación se refresca si el fichero `Manifest` cambia (por ejemplo en un comentario)
- Estado de la caché es accesible desde JavaScript
- `var appCache = window.applicationCache;`
- `update()` y `swapCache()`

```
switch (appCache.status) {
  case appCache.UNCACHED: // UNCACHED == 0
    return 'UNCACHED';
    break;
  case appCache.IDLE: // IDLE == 1
    return 'IDLE';
    break;
  case appCache.CHECKING: // CHECKING == 2
    return 'CHECKING';
    break;
  case appCache.DOWNLOADING: // DOWNLOADING == 3
    return 'DOWNLOADING';
    break;
  case appCache.UPDATEREADY:  // UPDATEREADY == 4
    return 'UPDATEREADY';
    break;
  case appCache.OBSOLETE: // OBSOLETE == 5
    return 'OBSOLETE';
    break;
  default:
    return 'UKNOWN CACHE STATUS';
    break;
};
```

Interfaz Drag&Drop

- Arrastrar y soltar
- HTML5 permite varios tipos:
  - Drag and Drop texto y código HTML
  - Drag and Drop elementos HTML
  - Drag and Drop archivos
- Integrado con el escritorio
- Basado en eventos



DnD está basado en eventos

- `dragstart, drag, dragenter, dragleave, dragover, drop, dragend`
- Atributo `draggable="true"`
- Cuando se mueve un elemento HTML su ID se pasa al elemento padre que lo recibe.
- Propiedad `dataTransfer` lleva los datos que se envían en una acción de arrastrar
- Se rellena en evento `dragstart` y se lee o maneja en evento `drop`.
- `e.dataTransfer.setData(format, data)` pone el contenido del objeto al tipo MIME `format`

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}
function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}
function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
</head>

<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height
</body>
</html>
```

Referencias

- http://www.w3schools.com/html/
- http://www.html5rocks.com, artículos de Eric Bidelman
  - WebWorkers: http://www.html5rocks.com/en/tutorials/workers/basics/
  - DragNDrop: http://www.html5rocks.com/en/tutorials/dnd/basics/
  - AppCache: http://www.html5rocks.com/en/tutorials/appcache/beginner/
- https://www.w3.org/TR/html5/
- http://html5-demos.appspot.com/
- Wikipedia: https://en.wikipedia.org/wiki/HTML5