Tema 6
Parte II

APIs de HTML5 para dispositivos móviles

Javascript para móviles

Desarrollo de aplicaciones web móviles

1

Introducción: ¿por qué lo llaman HTML5...

...cuando quieren decir Javascript?

APIs de HTML5

- Las partes más innovadoras de HTML5 son en realidad los APIs javascript
  - Alguno ya lo conocéis (geolocalización, aunque técnicamente hablando este no es parte de HTML5)
- Ejemplos de APIs
  - Web Sockets: comunicación bidireccional cliente-servidor
  - WebWorkers: *multithreading*
  - Canvas: dibujo en 2D
  - Almacenamiento local
  - Cache para modo *offline*
- Mozilla WebAPIs ¿futuro?
  - https://wiki.mozilla.org/WebAPI

HTML5 en móviles

- Como vimos, muchos navegadores para móviles usan WebKit
  - WebKit tiene un buen soporte de APIs HTML5
  - Tabla de compatibilidad de APIs de HTML5 para móviles: mobilehtml5.org
- No todos los APIs HTML5 son adecuados para móviles
  - Canvas, por ejemplo, puede ser demasiado “pesado”
- Hay ciertos APIs para móviles que no son exactamente parte de HTML5
  - Touch Events

APIs de HTML5 para dispositivos móviles

2

Canvas

Dibujo en 2D

Canvas


```
<canvas id="c" width="400" height="300">
</canvas>
<script>
    var c = document.getElementById("c").getContext("2d");
    //gradiente desde xinic, yinic hasta xfinal yfinal
    var grad = c.createLinearGradient(0, 0, 200, 100)
    //en el p. inic. el gradiente es blanco, y en el final negro
    grad.addColorStop(0, 'white')
    grad.addColorStop(1, 'black')
    c.fillStyle = grad
    c.fillRect(10, 10, 200, 150);
    c.beginPath();
    c.arc(75, 75, 50, 0, Math.PI * 2, true); //cara
    c.moveTo(110, 75);
    c.arc(75, 75, 35, 0, Math.PI, false);   //boca
    c.moveTo(65, 65);
    c.arc(60, 65, 5, 0, Math.PI * 2, true); //ojo izq
    c.moveTo(95, 65);
    c.arc(90, 65, 5, 0, Math.PI * 2, true); //ojo der
    c.strokeStyle = 'rgb(255, 0, 0)';
    c.lineWidth=10;
    //dibujar. Hasta este momento no se dibuja nada
    c.stroke();
</script>
```

APIs de HTML5 para dispositivos móviles

3

Almacenamiento local

Datos compartidos entre páginas (¡por fin!)

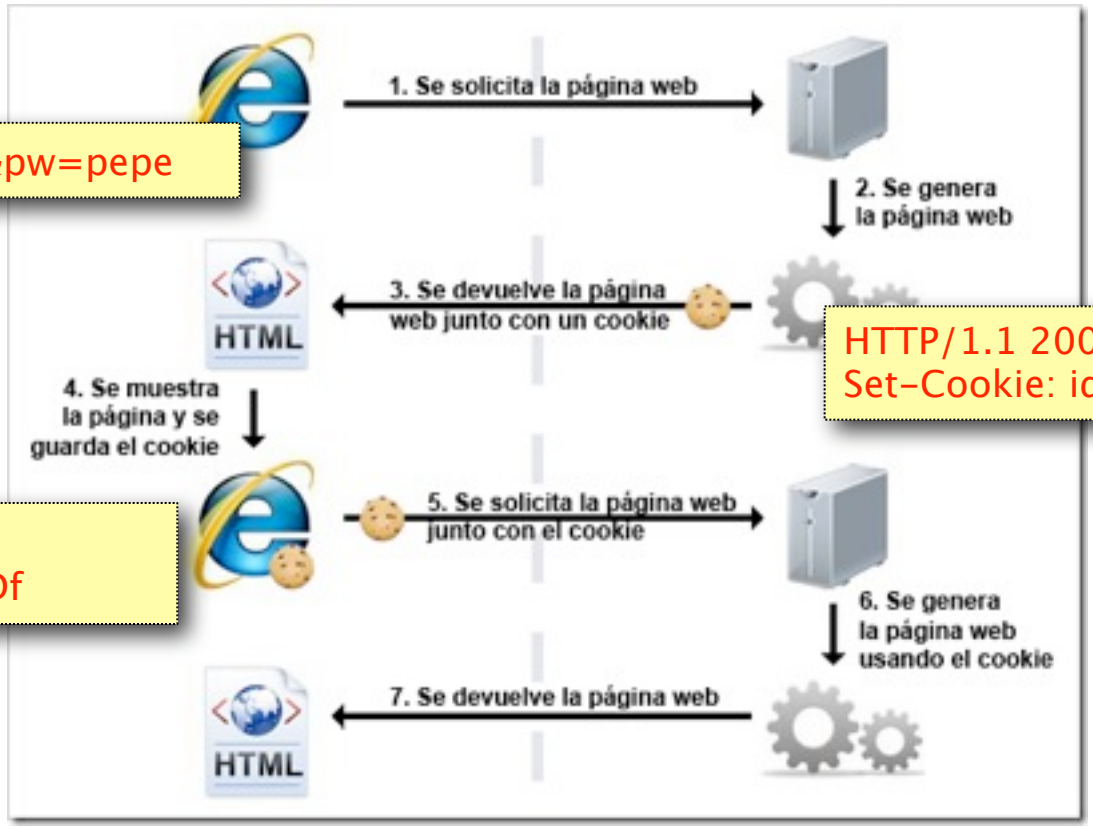
Breve historia del almacenamiento local

- Justificación: las variables JavaScript tienen de ámbito la página. Si se cambia o se recarga, vuelven al valor original
- Algunos mecanismos de almacenamiento local
  - Cookies: omnipresentes en la actualidad. Existen desde los inicios de la web
  - Google gears: extensión para el navegador que permite almacenar datos en local en una base de datos SQL
  - HTML5 local storage
  - Bases de datos en local (de HTML5)
    - HTML web SQL database
    - HTML5 IndexedDB

Cómo funcionan las *cookies*

- Son pares “nombre=valor”. A partir del momento en que se crean, el navegador las envía al servidor que las creó en cada petición

GET login?login=pepe&pw=pepe



HTTP/1.1 200 OK
Set-Cookie: idUser=Ax76Df

GET loquesea.html
Cookie: idUser=Ax76Df

Gestionar *cookies* desde Javascript

- A través de document.cookie
  - Para **guardar** un par nombre valor: asignarlo a document.cookie (junto con las propiedades de la cookie)
  - Para **leer**: se obtienen todos los pares a la vez (!) separados por ";"

```
//Fecha de hoy
caduca = new Date();
//Fecha justo dentro de 1 año
caduca.setFullYear(caduca.getFullYear()+1);
nombre = prompt("¿cómo te llamas?");
document.cookie = "nombre=" + escape(nombre)
      //expires indica la fecha de "caducidad" de la cookie
      + "; expires=" + caduca.toGMTString()
```

- Problemas
  - Generan demasiado tráfico cliente/servidor
  - El API javascript es tedioso de usar
  - Longitud limitada a 4Kb por cookie, 20 cookies por sitio

Almacenamiento local en HTML5

- Se almacenan pares “clave=valor”
  - Aunque la especificación no restringe el tipo para el valor, por el momento todos los navegadores lo almacenan como String
  - Esto quiere decir que al recuperarlo tendremos que convertirlo al tipo original.
  - Un enfoque muy típico es usar JSON, así podemos guardar objetos
    - Convertir de objeto a cadena: `JSON.stringify(objeto)`
    - de cadena a objeto: `JSON.parse(cadena)`
- Hay dos tipos de almacenamiento
  - Objeto **localStorage**: el ámbito del dato es el sitio web. Se conserva aunque se cierre el navegador
  - Objeto **sessionStorage**: el ámbito es la ventana (o solapa, si tenemos varias abiertas). Se conserva hasta que ésta se cierre

API básico

- `getItem(clave)`
- `setItem(clave,valor)`
- `length`: propiedad de solo lectura que indica cuántos pares clave/valor hay almacenados
- `key(i)`: devuelve el nombre de la clave i-ésima (para poder recuperar su valor con `getItem`)
- `clear()`: eliminar todos los datos

Ejemplo


```
function guardarNombre() {
   nombre = prompt("¿cómo te llamas?")
   localStorage.setItem("usuario", nombre)
   //esta sintaxis es equivalente a lo anterior
   localStorage.usuario = nombre
   //y esta también
   localStorage["usuario"] = nombre
    edad = prompt("¿Cuántos años tienes?")
   localStorage.setItem("edad", edad)
}

function mostrarNombre() {
   alert("Me acuerdo de ti, " + localStorage.usuario +
        " vas a cumplir " + parseInt(localStorage.edad) + 1
        + " años")
}

function mostrarTodosLosDatos() {
   datos=""
    for(var i=0; i<localStorage.length; i++) {
      clave = localStorage.key(i)
      datos = datos + clave + "=" + localStorage[clave] + '\n'
    }
    alert("LocalStorage contiene " + datos)
}
```

Bases de datos en el cliente

- Hay dos estándares
  - Web SQL: una base de datos relacional, accesible con SQL
  - IndexedDB: una base de datos de pares clave-valor (tipo NoSQL)
- El estándar apoyado oficialmente es IndexedDB. Web SQL se ha dejado de mantener, no habrá versiones futuras, pero...
  - Ahora mismo Web SQL funciona en todos los WebKit móviles, IndexedDB en ninguno

Web SQL

- Características
  - Es **SQL**, inmediatamente familiar para millones de desarrolladores
  - API **asíncrono**. Todas las operaciones usan *callbacks*
  - **Transaccional**: hay que hacer una transacción para todas las operaciones SQL (incluyendo SELECT)
  - Todas las implementaciones existentes usan **SQLite** www.sqlite.com
  - Limitaciones: 5Mb por aplicación

Ejemplo Web SQL


```
//abrir BD(nombre, versión, nombre_ampliado, tamaño en bytes)
var bd = openDatabase('Tareas', '1.0', 'tareas pendientes',5*1024*1024);

//crear una tabla: asíncrono y transaccional, como todas las llamadas
bd.transaction(function(trans) {
    //ejecutar SQL(sentencia_SQL, array_con_parámetros)
    trans.executeSql('CREATE TABLE IF NOT EXISTS ' +
                  'tarea(ID INTEGER PRIMARY KEY ASC, texto TEXT, ' +
                   prioridad INTEGER)', []);
});

//insertar datos
var texto = prompt("Introduce nueva tarea");
var prioridad = parseInt(prompt("Introduce su prioridad"));
bd.transaction(function(trans) {
    trans.executeSql('INSERT INTO tareas(texto,prioridad) VALUES (?,?)',
                     [texto, prioridad])

});
```

Ejemplo Web SQL (II)


```
//obtener datos
bd.transaction(function(trans) {
    //a executeSQL le podemos pasar un callback que se ejecutará en caso
    //de éxito, y otro en caso de error
    trans.executeSql('SELECT * FROM tareas', [], verDatos, error)
});

//el callback automáticamente recibe la transacción y, si es un SELECT,
//los registros resultado de la consulta
function verDatos(trans, rs) {
   //rows son los registros en sí
   for(var i=0; i<rs.rows.length; i++)
      alert("Tarea: " + rs.rows.item(i).texto + "\n" +
            "Prioridad: " + rs.rows.item(i).prioridad)
}

function error() {
   alert("Se ha producido un error con la BD");
}
```

4

Interfaz táctil

Touch y MultiTouch

APIs de HTML5 para dispositivos móviles

Eventos táctiles

- touchstart, touchmove, touchend
  - Similares a los de ratón: mousedown (pulsado botón), mousemove, mouseup (soltado botón)
- Cuando se produce uno de estos eventos, el evento tiene un array “touches”
  - Cada posición tiene “target” (elemento DOM tocado), coordenadas, etc
  - En dispositivos sin *multitouch* este array solo tiene una posición
- El evento click también se genera con toques
  - Pero hay un retraso de unos 300ms en su disparo, tiempo que el navegador espera para ver si se hace doble clic en lugar de uno solo, (ya que doble clic significa zoom)
  - Fast buttons: implementación alternativa sin el retardo http://code.google.com/mobile/articles/fast_buttons.html

Ejemplo: dibujar en canvas


```
<canvas id="c"></canvas>
<script>
var context = null
var canvas = null

//inicializa el canvas
function initCanvas() {
    canvas = document.getElementById("c")
    //le damos el alto y ancho de la ventana
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight
    context = canvas.getContext("2d")
}

//cuando se carga la página o se cambia la orientación, inicializar
window.addEventListener("orientationchange", resetCanvas)
window.addEventListener("load", function() {
    //truco para que desaparezca la barra de URL en Android
    window.scrollTo(0, 1);
    initCanvas()
```

Ejemplo: dibujar en canvas (II)


```
//continúa...

//comienzo de toque: movemos la "punta del lápiz" a ese punto
document.getElementById("c").addEventListener("touchstart",
  function(e) {
    context.beginPath();
    context.moveTo(e.touches[0].pageX, e.touches[0].pageY);
    e.preventDefault();
 },
 false
)

//arrastramos el dedo: se dibuja el trazo con una recta
document.getElementById("c").addEventListener("touchmove",
 function(e) {
    context.lineTo(e.touches[0].pageX, e.touches[0].pageY);
    context.stroke();
 },
 false
)
</script>
```

5

Sensores

Acelerómetro, giroscopio, ...

APIs de HTML5 para dispositivos móviles

Sensores de orientación

- Los dispositivos modernos tienen acelerómetros y/o giroscopios que nos permiten detectar la orientación (horizontal, vertical) e incluso la posición en 3D
- Evento `orientationchange`
  - El evento se dispara cuando cambia la orientación portrait-landscape
  - La orientación actual se puede obtener con `window.orientation`: 0 portrait, 90 landscape (derecha), -90 landscape (izquierda)
- Posición 3D:
  - Los eventos se disparan x veces por segundo (típicamente 50-60)
  - evento `deviceOrientation`: orientación en los 3 ejes usando giroscopio
  - evento `deviceMotion`: orientación usando acelerómetro

APIs de HTML5 para dispositivos móviles

6

Aplicaciones *offline*

Sitios web sin conexión a Internet

Aplicaciones web *offline*

- Podemos forzar al navegador a que cachee los HTML, js, etc. para que así se pueda usar la aplicación *offline*
- El comportamiento de la cache se establece en un fichero aparte, referenciado en la etiqueta <html>

```
#este sería el archivo "cache.manifest"
#La línea siguiente es obligatoria
CACHE MANIFEST
#Listar los archivos a cachear
CACHE:
index.html
misfunciones.js
css/style.css
#esto no deberia cachearse, no tiene sentido
NETWORK:
js/ajax.js
#si se intenta acceder a esto, mostrar el otro
FALLBACK:
chat.html error.html
```

```
<html manifest="cache.manifest">
   …
<html>
```

Referencias: sitios web

- html5rocks.com
  - Excelente sitio sobre HTML5, con muchas demos, tutoriales, ejemplos de código.... “patrocinado” por Google
- diveintohtml5.org
  - Muy buen libro sobre HTML5, disponible libremente *online*. La edición impresa la vende O’Reilly

Referencias: libros

- http://proquestcombo.safaribooksonline.com/ (solo desde la UA)
  - *Build Mobile Websites and Apps for Smart Devices*, Ed. SitePoint, 2011

  - *Pro Android Web Apps*, *Develop for Android Using HTML5, CSS3 & JavaScript*, Apress 2010