

**Tema 6: Aplicaciones web en dispositivos móviles**

# **Tema 6: Aplicaciones web en dispositivos móviles**

## **Tema 6: Aplicaciones web en dispositivos móviles**

# **1. Móviles vs. Escritorio**

# La pantalla

- Aunque la resolución está a la par de los escritorios **no podemos colocar demasiada información** ya que la pantalla es físicamente muy pequeña
- **Más alta que ancha**, al contrario que en el escritorio.
- **Es incómodo hacer scroll horizontal**, por lo que la información tiende a colocarse en una columna



# Dispositivos de entrada

- Introducir datos en un móvil es incómodo y tedioso.
  - La aplicación debe reducir esta necesidad al mínimo imprescindible
  - Ejemplo: recordar login y password, recordar preferencias,...
- Hay que tener en cuenta las características del dispositivo de entrada
  - Los controles táctiles tienen que ser grandes, para poder ser pulsados con el dedo.



# Otros elementos

- Capacidad de procesamiento
  - Ciertos APIs no tienen todavía suficiente rendimiento para los móviles (WebGL, canvas,...)
- Batería
  - Javascript gasta batería, motivo por el que la mayoría de navegadores móviles paran su ejecución en las pestañas en segundo plano
  - También ciertos APIs gastan mucha batería por el hardware que usan “por debajo” (ej. geolocalización)



## **Tema 6: Aplicaciones web en dispositivos móviles**

# **2. Tipos de aplicaciones**

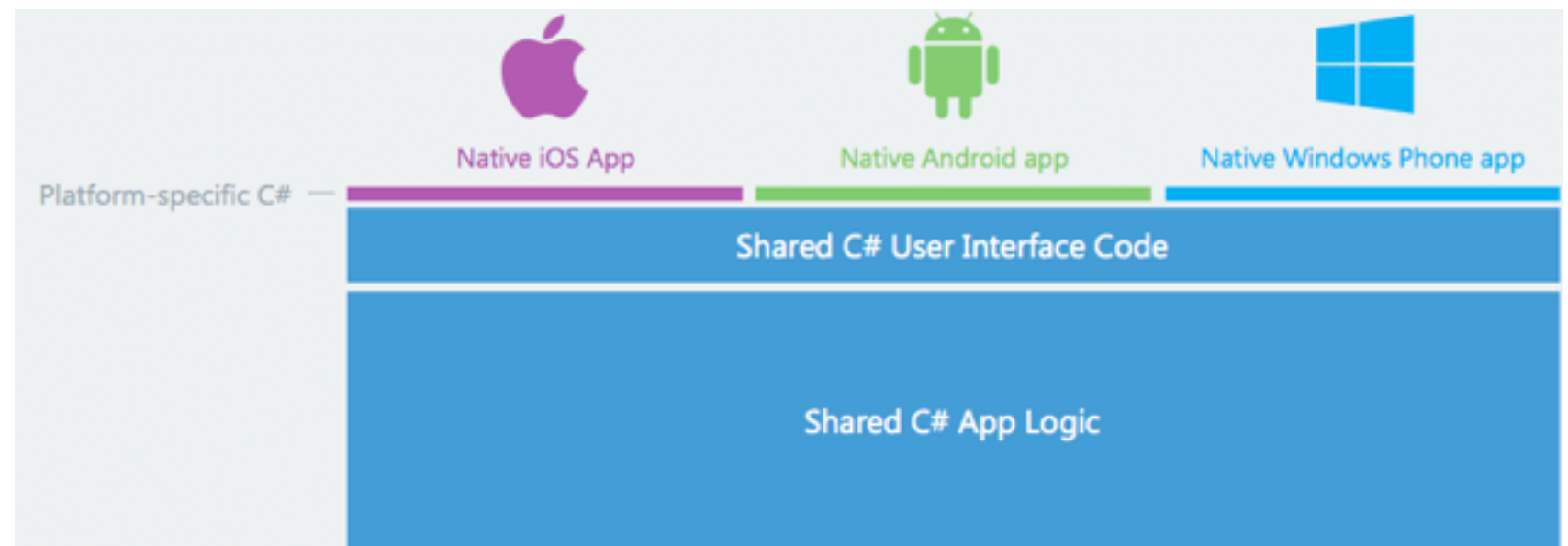
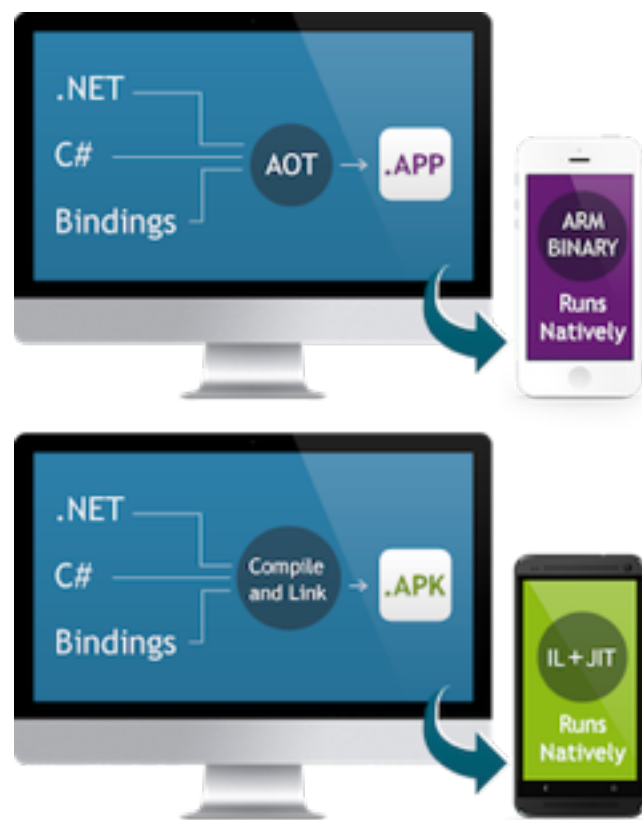
# 1. Aplicaciones nativas

- Con el SDK nativo de la plataforma
- Ventajas:
  - “Exprimen” al máximo el hardware
  - “*Look & Feel*” de la plataforma
- Inconvenientes
  - Escasa portabilidad
  - Desarrollo costoso



## 2. SDKs multiplataforma

- Programar en un solo Lenguaje/API, desplegar código nativo en todas las plataformas
- Ventajas:
  - Portabilidad
  - Rendimiento



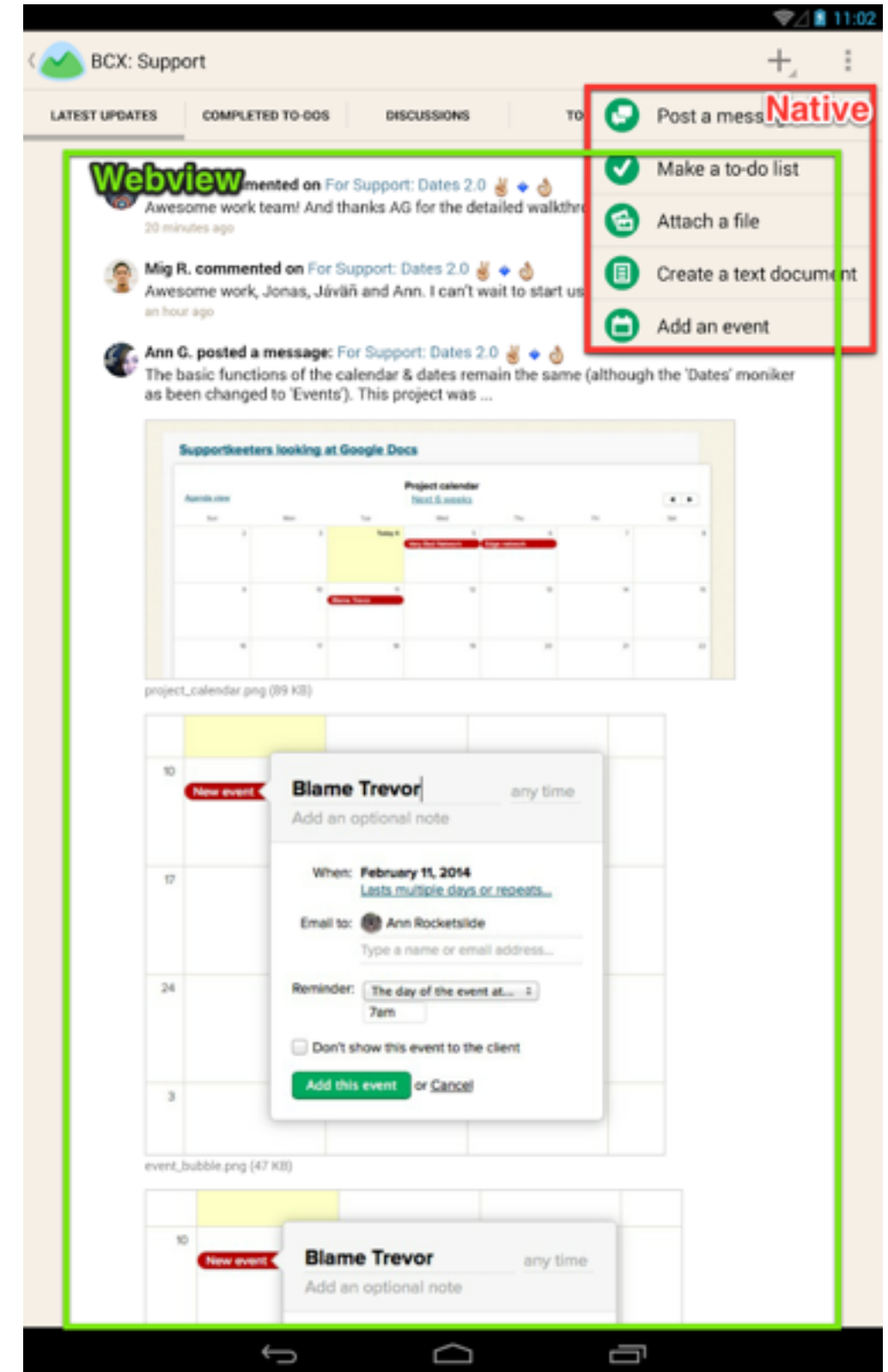
<http://xamarin.com>



# 3. Aplicaciones híbridas

- La aplicación web está contenida en un “envoltorio” nativo
- Ventajas:
  - Portabilidad
  - Acceso a APIs nativos desde Javascript
- Inconvenientes:
  - “Look & Feel” no del todo nativo
  - Rendimiento algo menor (relevante o no según el tipo de app)

*Hybrid sweet spot: Native navigation, web content*, del blog de @dhh



# 4. Aplicaciones web

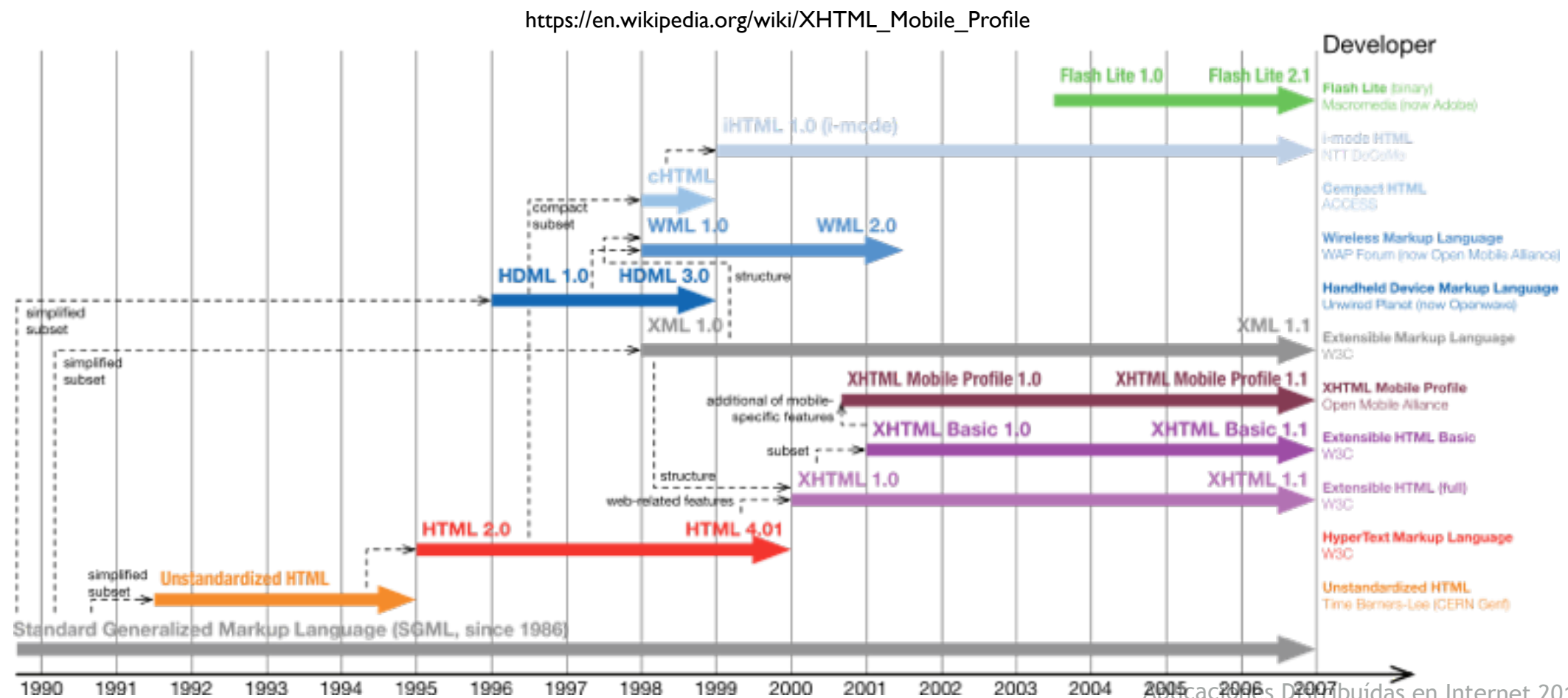
- Aplicación hecha enteramente con Javascript, interfaz en HTML/CSS
- Ventajas:
  - Portabilidad
  - Baja “barrera de entrada” para el desarrollador: solo HTML/CSS/JS
  - Se puede escapar al control de las tiendas de apps oficiales
- Inconvenientes:
  - Rendimiento no aceptable para algunos tipos de aplicaciones (p.ej. juegos 3D)
  - Algunos APIs no accesibles (p.ej. captura de videos)
  - No se pueden vender en las tiendas de apps oficiales

## **Tema 6: Aplicaciones web en dispositivos móviles**

# **3. HTML/CSS/ Javascript estándar en móviles**

# Estándares de marcado

- Hay diferentes **variantes de HTML y CSS** específicas para móviles
  - XHTML Basic (subconjunto de XHTML) y CSS MP (del W3C)
  - XHTML MP (Mobile Profile, ampliación de XHTML Basic) y WAP CSS (de la OMA-Open Mobile Alliance)
- En la actualidad se tiende a usar **simplemente HTML5**



# El viewport

- El dispositivo va a “simular” que tiene una resolución distinta a la real
  - El iPhone original tenía 320px de ancho, pero escalaba las páginas tomando como referencia 980px
  - Esto se hizo para que las webs aparecieran como en el escritorio (y no “cortadas” en vertical)
  - Por motivos históricos/prácticos se ha conservado la idea



<http://developer.apple.com/library/ios/#documentation/AppleApplications/Reference/SafariWebContent/UsingtheViewport/UsingtheViewport.html>

# Controlar el viewport

- Etiqueta `<meta>` en la cabecera

```
<!-- 'user-scalable=no' significa que no se puede hacer zoom. Esto  
es raro en la web, pero habitual en una app nativa -->  
<meta name="viewport" content="width=320,user-scalable=no">  
  
<!-- teóricamente 'device-width' es la resolución horizontal del  
dispositivo -->  
<meta name="viewport" content="width=device-width">
```

- En realidad, muchos dispositivos con resolución mayor de 320px aún así dan 320 como **device-width**

# “Retina” Web

- “Tipos” de pixels:
  - Device pixels: los del hardware
  - CSS pixels: los que se usan en las reglas CSS
- Como hemos visto, un dispositivo puede reportar que tiene 320 px de ancho (CSS pixels) cuando en realidad podría tener 640
- `window.devicePixelRatio`: ¿Cuántos píxeles reales es 1 pixel CSS?
- ¿Para qué se usan los pixels “extra”?
  - Para mostrar texto más nítido
  - Podemos usar esta información para servir imágenes de “alta resolución” vs. “resolución estándar”

# Subir archivos

- En HTML para subir un archivo al servidor en un formulario se usa un `<input type="file">`
  - En iOS sigue siendo imposible subir algo que no sea audio/video
- Se puede especificar el tipo MIME aceptable con **accept**

```
<input type="file" accept="application/pdf">  
<input type="file" accept="image/jpeg">  

```



# “Trucos” para que la app parezca algo más nativa

- **Icono en el escritorio**

```
<!-- iOS y Android versiones más antiguas-->
<link rel="apple-touch-icon" sizes="120x120" href="mi-icono.png">
<!-- Chrome for Android <M39 -->
<link rel="icon" sizes="128x128" href="mi-icono128x128.png">
<!-- Chrome for Android >M39
En el manifest.json está la info del icono (Entre otras cosas) -->
<link rel=""manifest" href="manifest.json">
```

- Problema: el usuario debe añadir manualmente el “favorito” a la pantalla de inicio

- **Aplicación a pantalla completa (sin interfaz de navegador)**

```
<!-- pantalla completa en iOS -->
<meta name="apple-mobile-web-app-capable" content="yes">
<!-- pantalla completa en Chrome for Android <M39.
En las posteriores se puede usar el manifest.json -->
<meta name="mobile-web-app-capable" content="yes">
```

# Otras ayudas

- Librerías Javascript adaptadas a móviles
  - **zepto.js** es una librería con un API compatible con el de jQuery pero mucho más ligera
- Plantillas HTML+CSS
  - **mobile boilerplate** incluye estilos por defecto (CSS reset), scripts de ayuda (eliminar barra de navegación, ...), librerías útiles (zepto.js) y un esqueleto HTML5 básico adaptado a móviles



# Javascript en móviles

- Tabla de compatibilidad de APIs de HTML5 para móviles: [mobilehtml5.org](http://mobilehtml5.org)
- Aquí hablamos de APIs estándar. Hay otros propios de ciertas plataformas, por ejemplo FirefoxOS, o Windows
- No todos los APIs HTML5 son adecuados para móviles por cuestiones de eficiencia
  - Ejemplos: Canvas, WebGL, ...

# Algunos APIs útiles en móviles

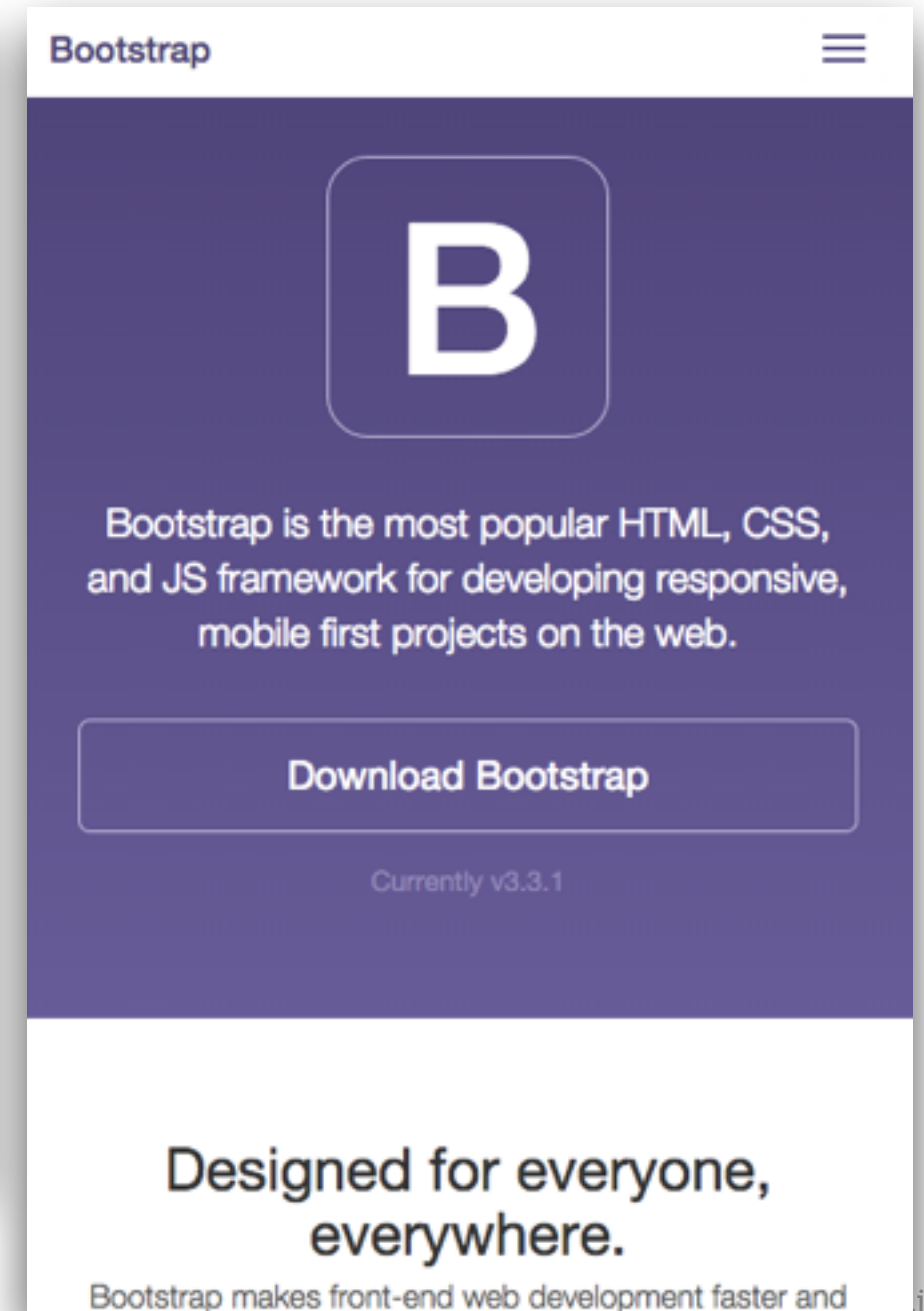
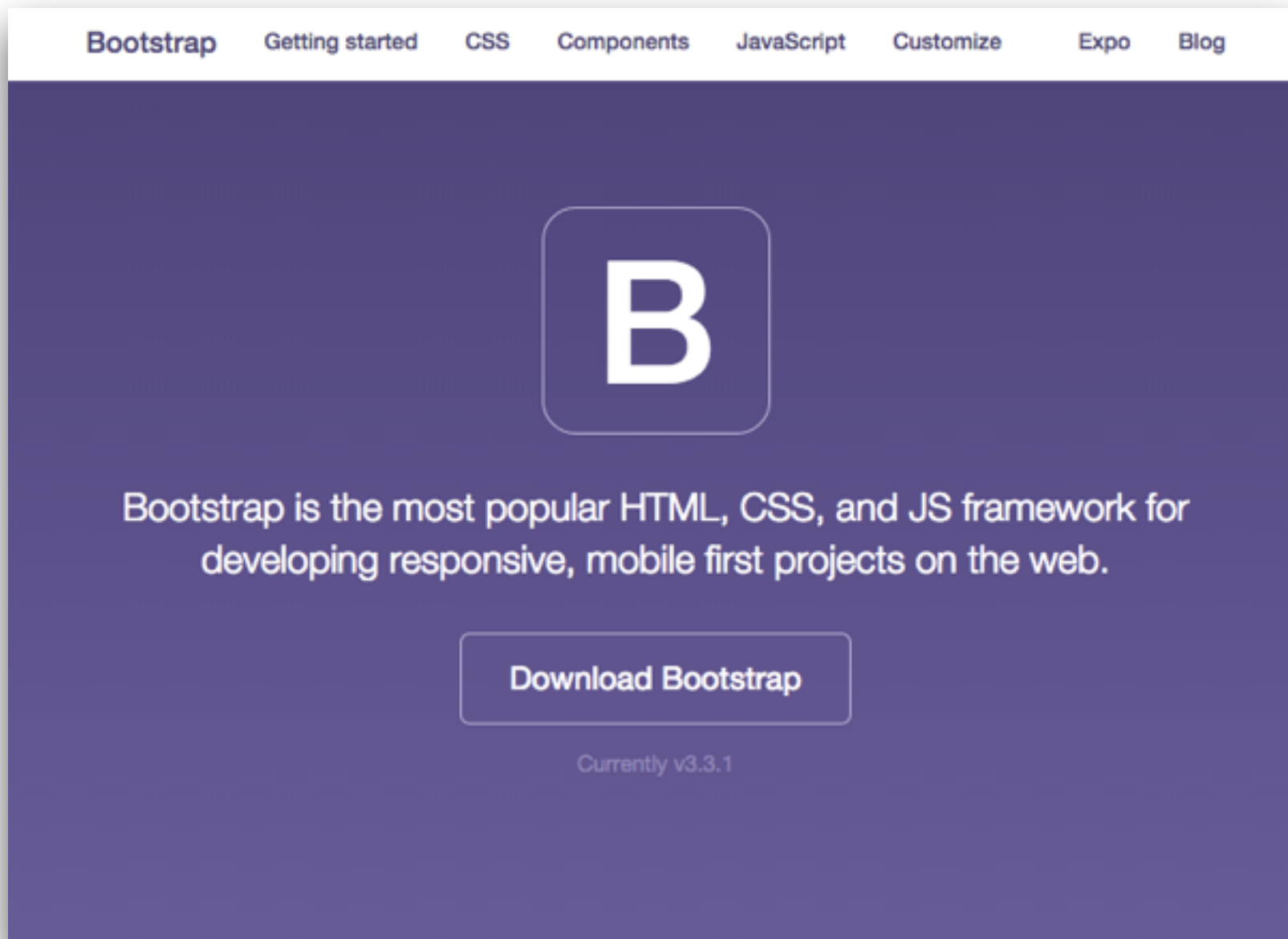
- Uso sin conexión al servidor
  - **localStorage**
  - **Web SQL / Indexed DB:** bases de datos en el cliente
  - **offline:** forzar al navegador a que cachee los HTML/CSS/JS, imágenes, etc. para que se pueda usar la app *offline*
- Interacción con el hardware
  - **Geolocalización**
  - **Acelerómetro y giroscopio**
  - **Touch**
  - **Battery status**
- Otros
  - **Web speech:** síntesis (iOS, Android) y reconocimiento (Android) del habla

**Tema 6: Aplicaciones web en dispositivos móviles**

# **4. Frameworks web para móviles**

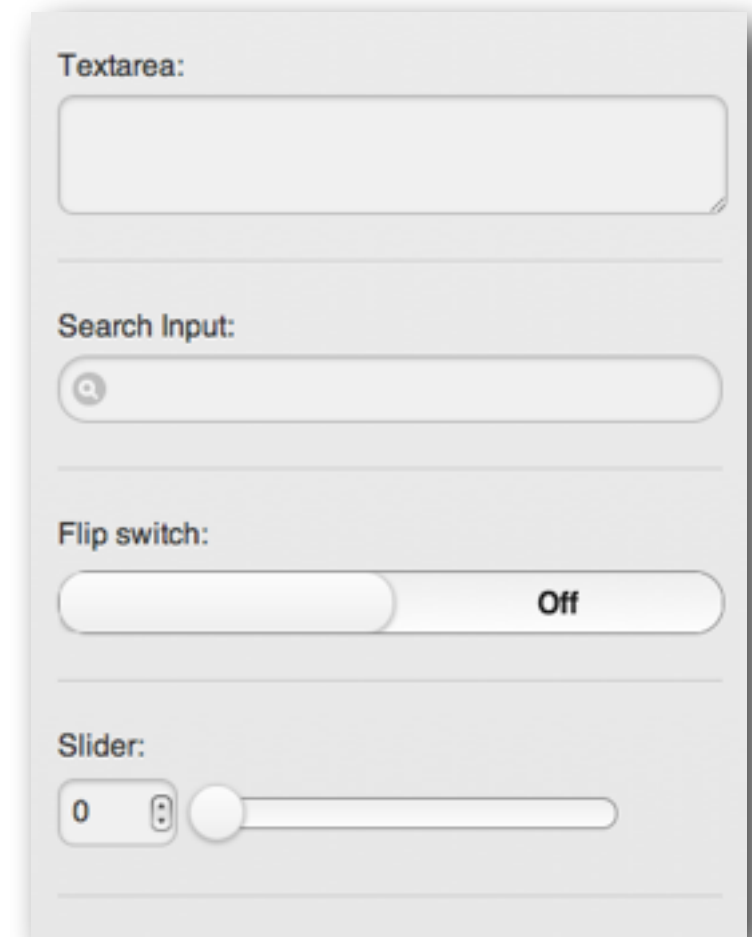
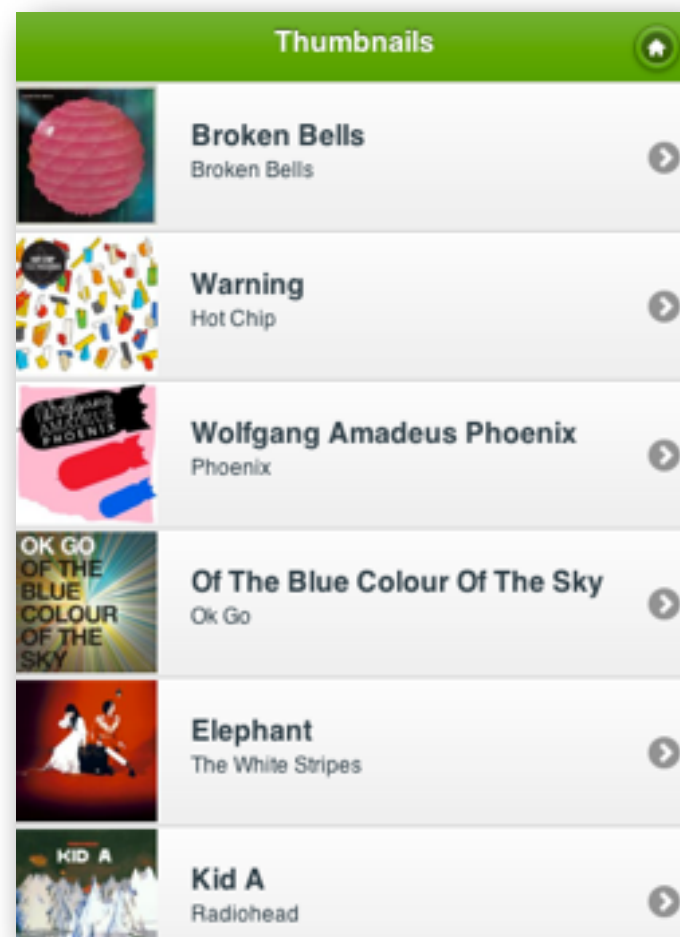
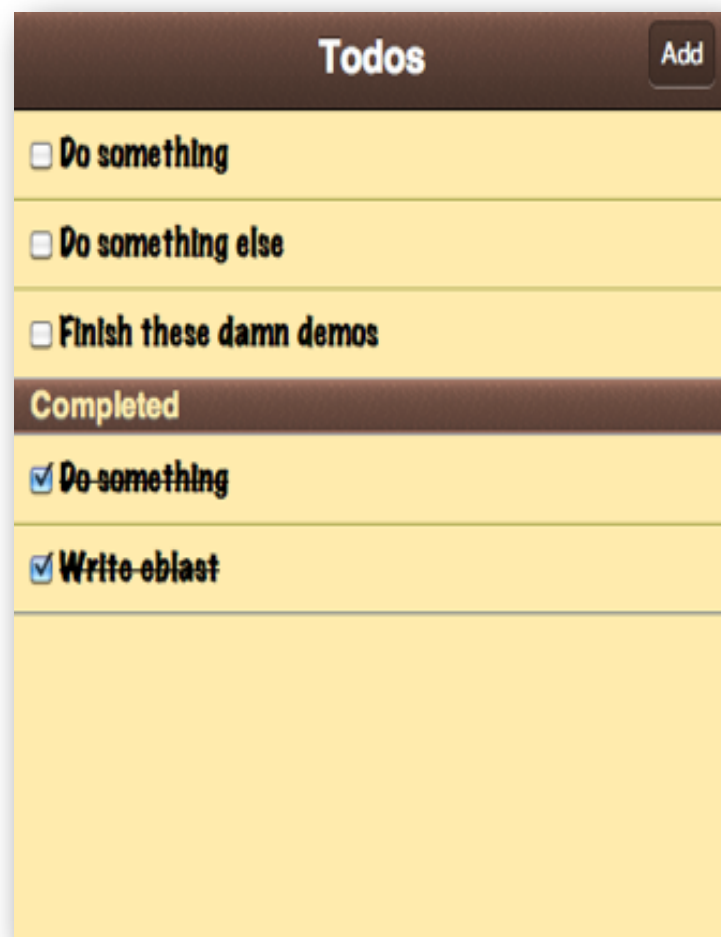
# Diseño “responsive”

- En principio podría valer un framework estilo **bootstrap**, ya que es *responsive* (se adapta a la resolución de pantalla)



# Qué aporta un framework móvil

- Elementos de interfaz adaptado a móviles (botones grandes, *switches*, listas,...)
- Transiciones entre pantallas
- Orientación a aplicaciones web más que a sitios web



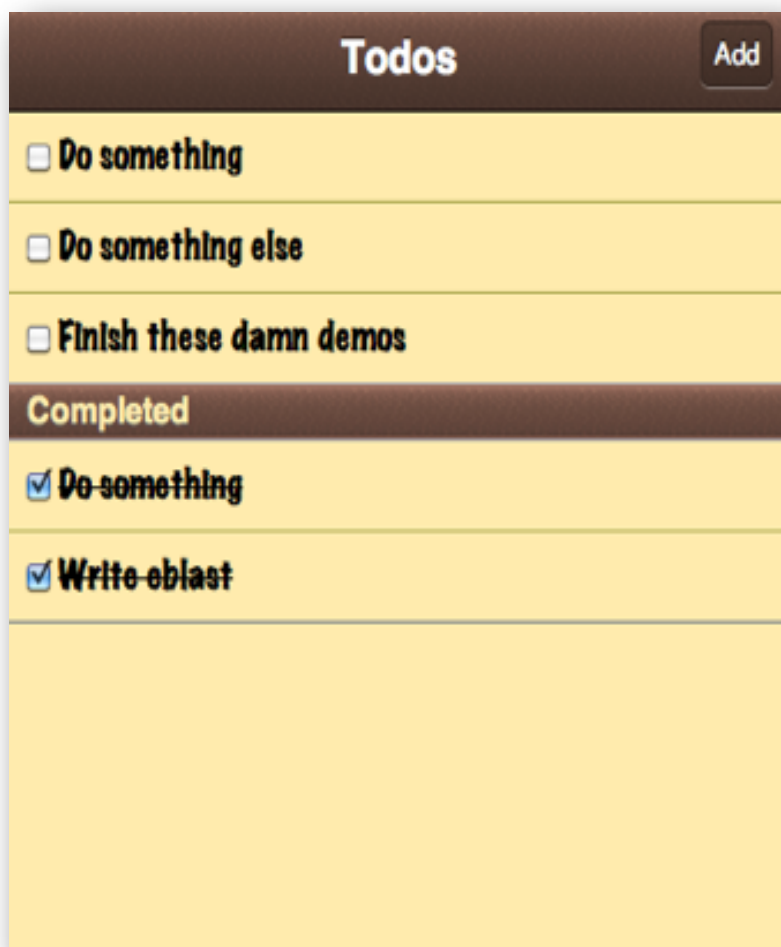
# Aplicaciones web vs. sitios web

- La mayoría de frameworks móviles están orientados al desarrollo de **aplicaciones web móviles**, no de sitios web
  - Sitio web móvil: conjunto de HTML enlazados entre sí y adaptados a móviles
  - **Aplicación web móvil: un único HTML contiene todas las pantallas (o páginas) de la aplicación.** Al igual que una app. nativa contiene todo el GUI, pero en lugar de estar desarrollado con Java u Obj-C usa JS+HTML+CSS
- No obstante, también pueden usarse para desarrollar sitios web más “convencionales”



# Tipos de frameworks

- “De marcado”
  - Definimos los elementos de pantalla con etiquetas HTML convencionales, con clases CSS o atributos HTML propios del framework
  - En la inicialización, el *framework* les asigna un aspecto (CSS) y un “comportamiento” (Javascript) especiales

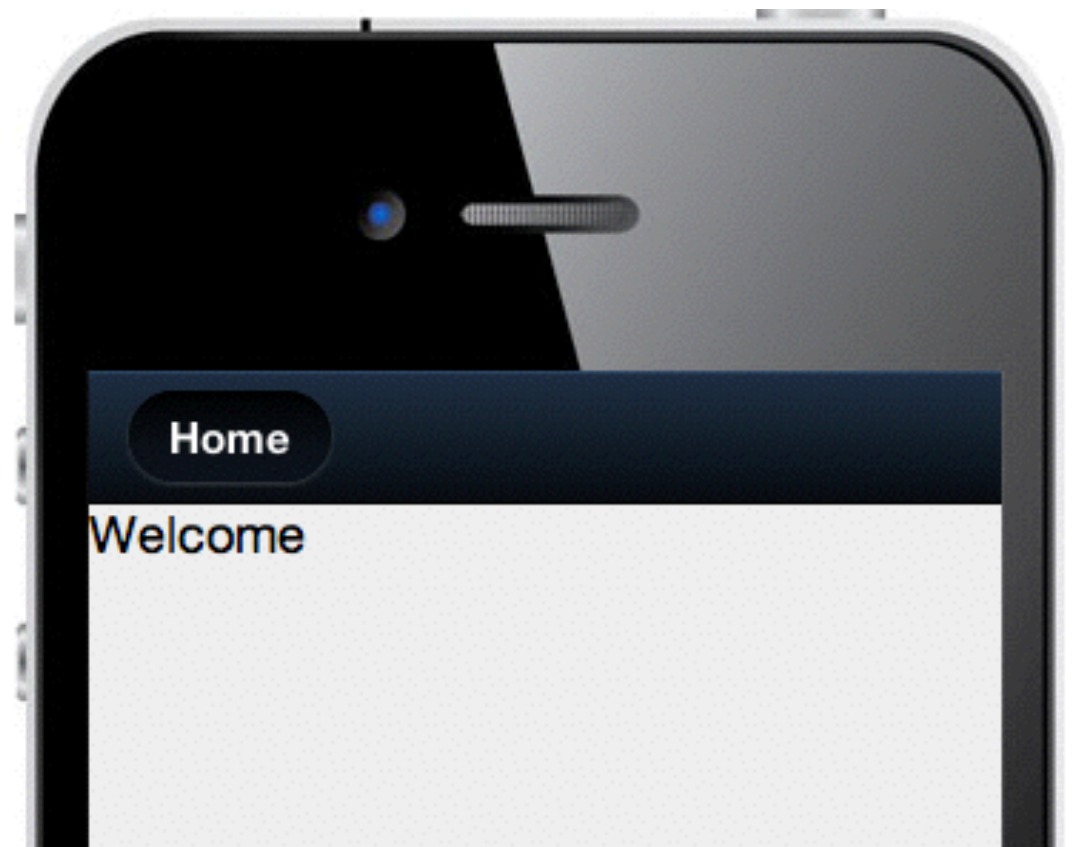


```
<!-- ejemplo con jQT (jq tj s .com) -->
<div id="home" selected="true" class="edgetoedge form">
<div class="toolbar">
  <h1>Todos</h1>
  <a href="#add" class="btn2 slideup">Add</a>
</div>
<ul>
  <li><input type="checkbox"/>Do something<small></small></li>
  <li><input type="checkbox"/>Do something else</li>
  <li><input type="checkbox"/>Finish these damn demos</li>
</ul>
<h4>Completed</h4>
<ul>
  <li><input type="checkbox" checked/></li>
  <li><input type="checkbox" checked />Write eblast</li>
</ul>
</div>
```

# Tipos de frameworks (II)

- “De código”
  - Más parecidos al concepto típico de “librería”. Cuando queremos definir por ejemplo un *widget* lo hacemos con código Javascript

```
Ext.application({
  name: 'Sencha',
  launch: function() {
    Ext.create("Ext.TabPanel", {
      fullscreen: true,
      items: [
        {
          title: 'Home',
          iconCls: 'home',
          html: 'Welcome'
        }
      ]
    });
  }
});
```



# Algunos frameworks



<http://propertycross.com/> una *app* de ejemplo implementada con muchos frameworks distintos, para que sea más sencillo compararlos y evaluarlos

# jQuery Mobile

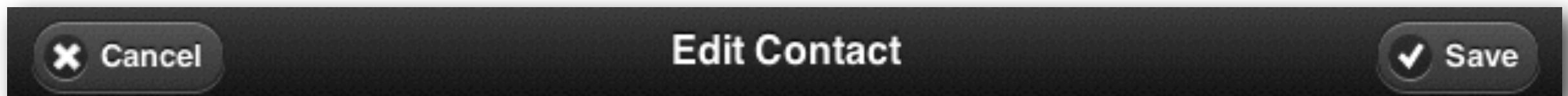
- El *framework* web para móviles más conocido y difundido
  - <http://jquerymobile.com/>
  - Bajo el “paraguas” de jQuery
  - Multiplataforma: iOS, Android, Windows Phone, ...
- Proporciona
  - widgets optimizados para móviles
  - personalización del estilo visual (temas)
- Problemas
  - Es algo “pesado”
  - Tiene un “look” mejorable



# Usar jQuery Mobile

- Incluir en el proyecto
  - El .js de jQuery
  - El .js, CSS e iconos de jQuery Mobile
- La mayoría de *widgets* se obtienen poniendo atributos especiales (**data-\***) a etiquetas HTML convencionales

```
<div data-role="header" data-position="inline">
  <a href="index.html" data-icon="delete">Cancel</a>
  <h1>Edit Contact</h1>
  <a href="index.html" data-icon="check">Save</a>
</div>
```

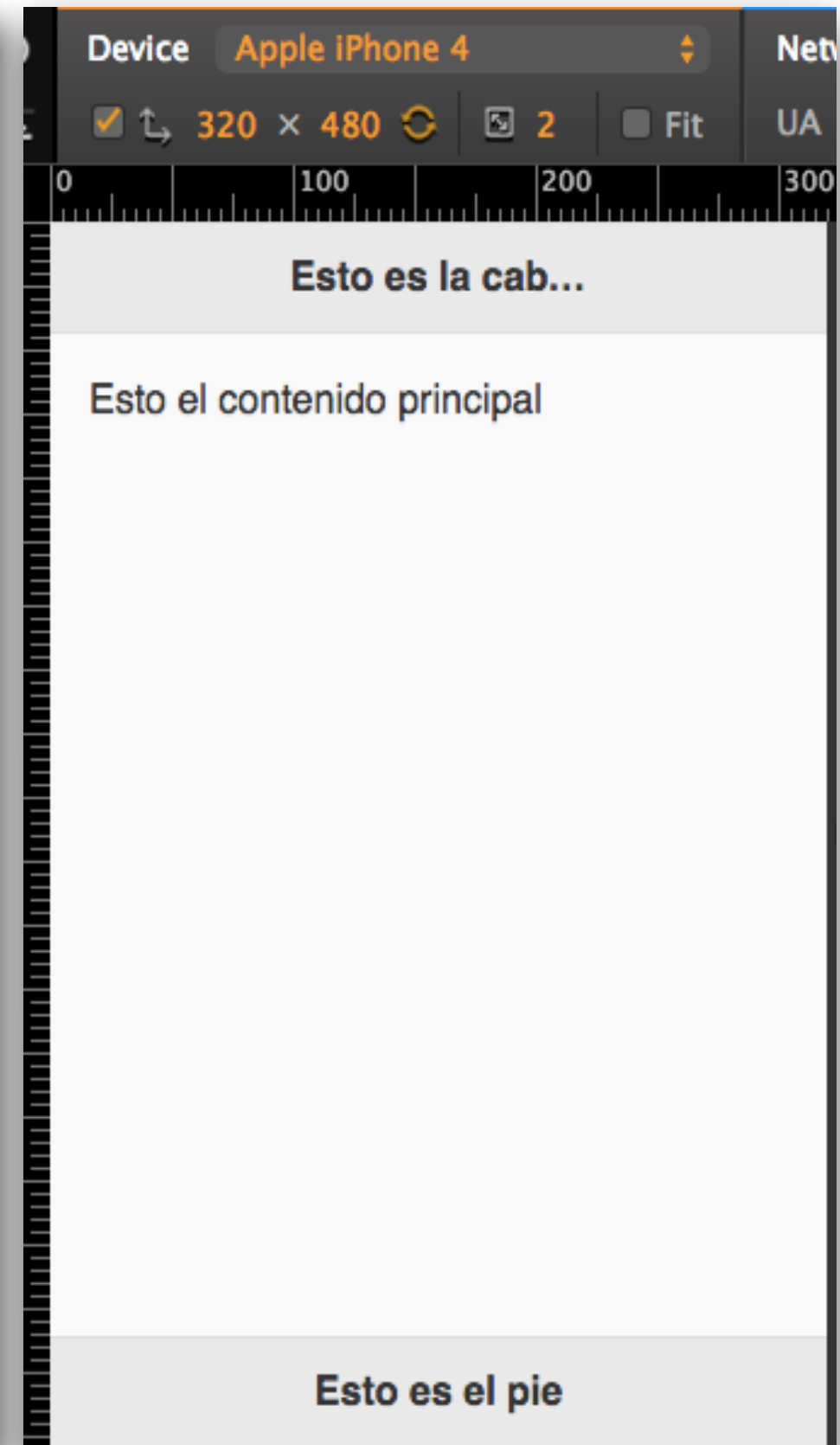


# Estructura básica de una página

- En **un solo archivo HTML** hay **varias páginas** (pantallas) de móvil
  - La aplicación está “autocontenida” y no necesitamos conexión con el servidor salvo que necesitemos pedirle datos dinámicos
- Cada pantalla será una etiqueta con **data-role="page"**
  - Habitualmente se usa <div>, pero se puede usar cualquier otra etiqueta
  - Cada pantalla puede tener **cabecera**, **contenido** y **pie**

# Hola jQuery Mobile

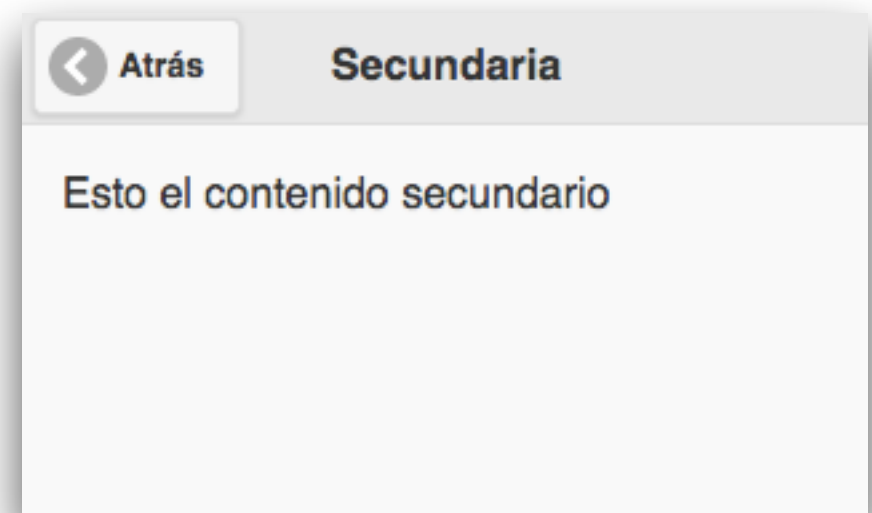
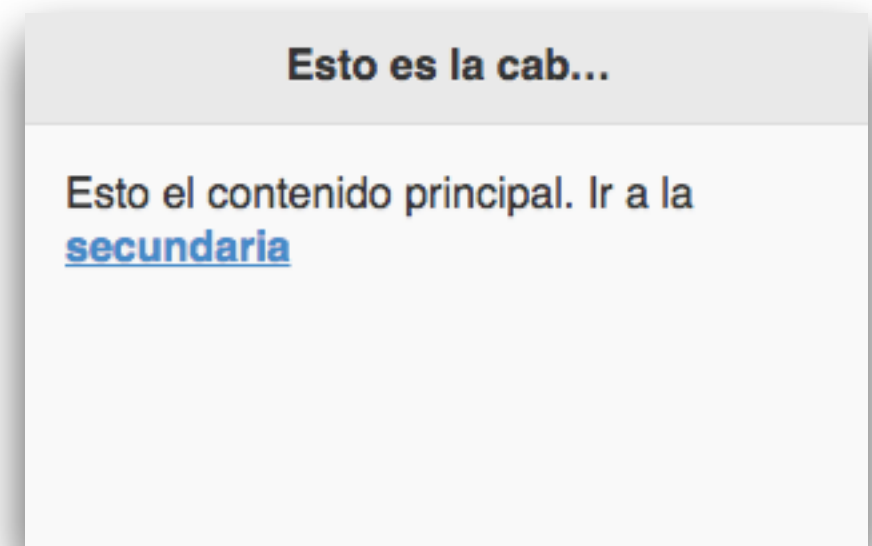
```
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
  <meta name="viewport" content="width=device-width,
initial-scale=1">
  <link rel="stylesheet" href="http://
code.jquery.com/mobile/1.4.5/
jquery.mobile-1.4.5.min.css" />
  <script src="http://code.jquery.com/
jquery-1.11.1.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.4.5/
jquery.mobile-1.4.5.min.js"></script>
</head>
<body>
  <div data-role="page">
    <div data-role="header">
      <h1>Esto es la cabecera</h1>
    </div>
    <div role="main" class="ui-content">
      Esto el contenido principal
    </div>
    <div data-role="footer" data-position="fixed">
      <h2>Esto es el pie</h2>
    </div>
  </div>
</body>
</html>
```





# Navegación entre páginas

```
<div data-role="page" id="principal">
  <div data-role="header">
    <h1>Esto es la cabecera</h1>
  </div>
  <div role="main" class="ui-content">
    Esto el contenido principal. Ir a la
    <a href="#secundaria" data-transition="flip">
      secundaria</a>
  </div>
</div>
<div data-role="page" id="secundaria">
  <div data-role="header">
    <h1>Secundaria</h1>
    <a href="#principal" data-rel="back">Atrás</a>
  </div>
  <div role="main" class="ui-content">
    Esto el contenido secundario
  </div>
</div>
```





# ¿Cómo se implementa?

jQuery Mobile “**secuestra**” el **evento de click** sobre los enlaces. Si detecta que el enlace corresponde a un cambio de página:

1. Se dispara una **transición CSS3** con la animación de cambio de página
2. Dinámicamente **se cambian las clases CSS** de la página actual para ocultarla y de la anterior para mostrarla
  - Disparando además una serie de eventos propios de jQuery Mobile con los que podemos “enganchar” para detectar el cambio de página
3. Se actualiza con javascript el **historial de navegación** (para que si pulsamos el “back” del navegador todo funcione correctamente)

# Enlaces externos

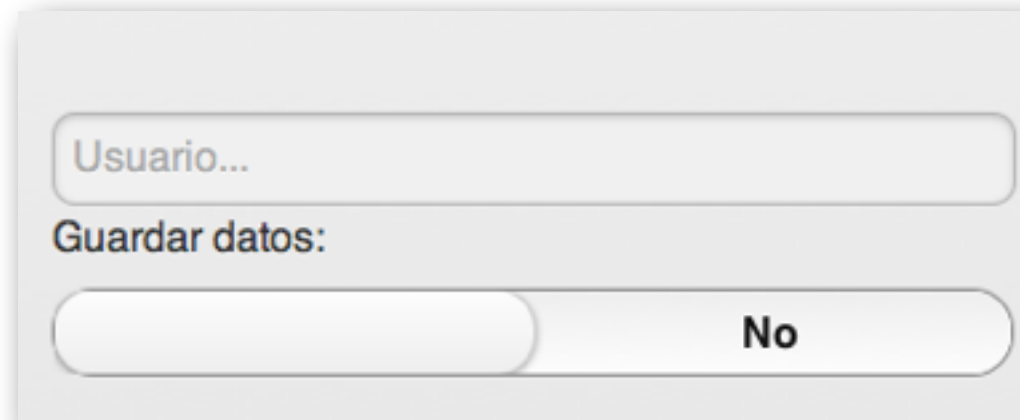
Si el href no lleva el “#”, se asume que enlazamos a otro HTML

- Pero por defecto no se cambia totalmente el HTML actual
  - Automáticamente se hace una petición AJAX para obtener el nuevo HTML
  - Se “examina” el nuevo HTML en busca de la primera etiqueta con data-role=”page”
  - Se inserta dinámicamente esta etiqueta en el DOM del HTML actual, ignorando el resto
  - Se hace la transición a la nueva página
- También podemos hacer una navegación convencional (cambiando totalmente el HTML por el nuevo)
  - Ponerle al enlace un `rel="external"` o un `data-ajax="false"`

# Campos de formulario

- Se usan etiquetas HTML convencionales
- Automáticamente se les da “look” de móvil y en algunos casos, funcionalidad táctil adicional

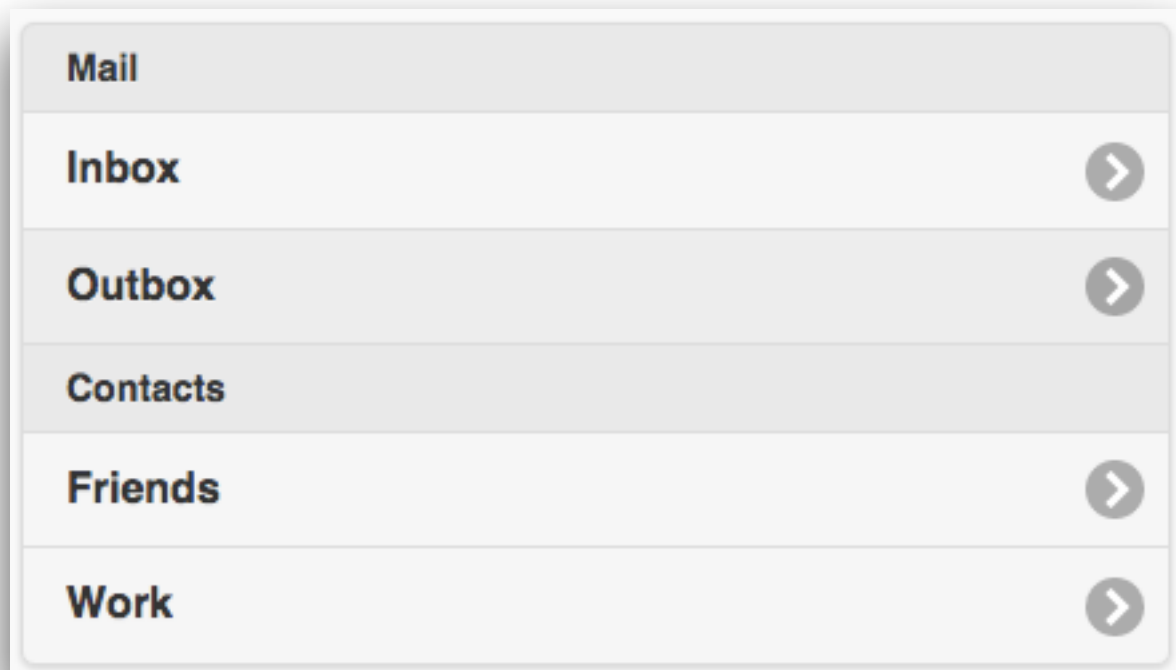
```
<div data-role="fieldcontain">
  <label for="usuario" class="ui-hidden-accessible">Usuario:</label>
  <input type="text" id="usuario" value="" placeholder="Usuario..." />
  <label for="flip">Guardar datos:</label>
  <select name="flip" id="flip" data-role="slider">
    <option value="no">No</option>
    <option value="si">Sí</option>
  </select>
</div>
```



Visual representation of the HTML code above, showing a text input field with placeholder "Usuario...", a label "Guardar datos:", and a slider control with "No" and "Sí" options.

# Listas

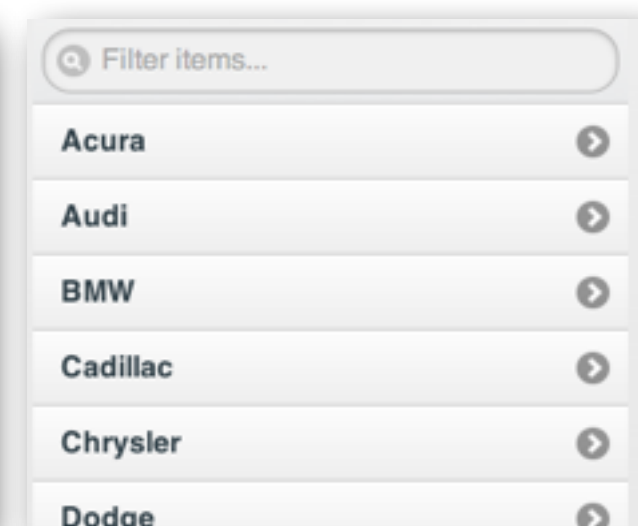
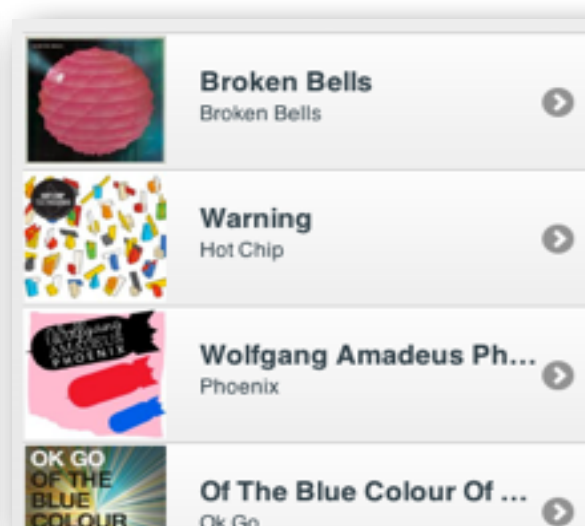
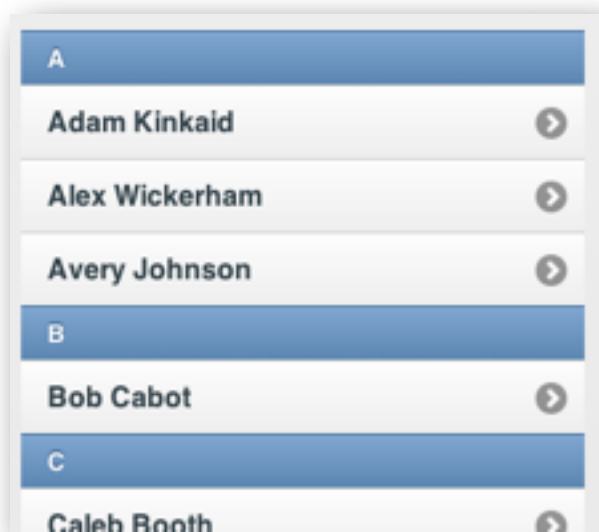
- En móviles es muy típico tener listas que ocupan toda la pantalla
  - Lista de tareas, contactos, eventos cercanos ...
    - En jQuery Mobile, `<ul>` con `data-role="listview"`
- También es muy típico que al pulsar en un elemento de la lista vayamos a otra pantalla para ver los detalles
  - Se pone un enlace dentro de cada `<li>` a la página adecuada



```
<ul data-role="listview" data-inset="true" data-divider-theme="a">
  <li data-role="list-divider">Mail</li>
  <li><a href="#">Inbox</a></li>
  <li><a href="#">Outbox</a></li>
  <li data-role="list-divider">Contacts</li>
  <li><a href="#">Friends</a></li>
  <li><a href="#">Work</a></li>
</ul>
```

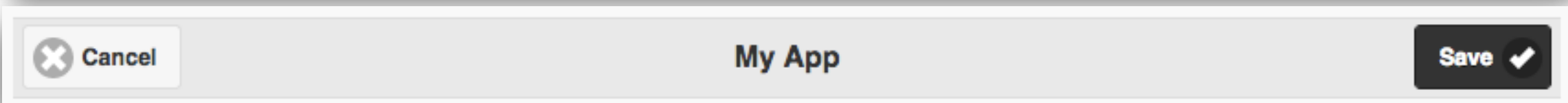
# Listas (II)

- La variedad y sofisticación de las listas en jQuery Mobile es considerable. **Algunas** posibilidades:
  - Thumbnails (<img> dentro de cada <li>, automáticamente se ajustará el tamaño)
  - Separadores (etiquetas con `data-role="list-divider"`)
  - “Badges” (típica “burbuja” que indica cuántos elementos hay): `class=ui-li-count`
  - Filtrar automáticamente elementos: lista con `data-filter="true"`

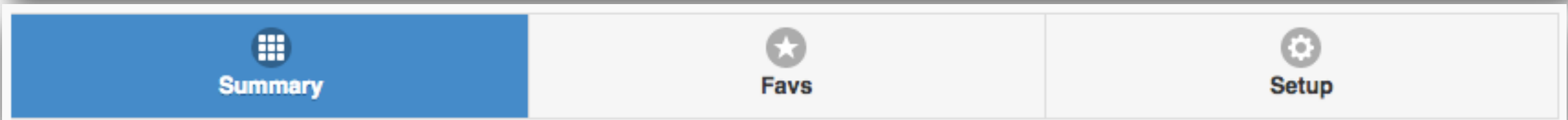


# Barras

```
<div data-role="header">
  <a href="#" class="ui-btn-left ui-btn ui-btn-inline ui-mini ui-corner-all ui-
  btn-icon-left ui-icon-delete">Cancel</a>
  <h1>My App</h1>
  <button class="ui-btn-right ui-btn ui-btn-b ui-btn-inline ui-mini ui-corner-
  all ui-btn-icon-right ui-icon-check">Save</button>
</div>
```



```
<div data-role="footer">
  <div data-role="navbar">
    <ul>
      <li><a href="#" data-icon="grid" class="ui-btn-active">Summary</a></li>
      <li><a href="#" data-icon="star" >Favs</a></li>
      <li><a href="#" data-icon="gear">Setup</a></li>
    </ul>
  </div>
</div>
```



# Eventos importantes

- Usar el `$(document).ready` para inicializar **aspectos globales** de la aplicación
- Usar el evento `pageinit` de cada página para inicializar lo propio de ella
  - Por ejemplo vincular un *listener* a un botón
- Usar el `pagebeforeshow` para inicializar lo que puede cambiar cada vez que se muestra
  - Por ejemplo, si es una lista generada dinámicamente, insertar los `<li>`

# Ejemplo

```
//Esto se ejecuta una sola vez para el documento entero
$(document).ready(function() {
    //personalizar el texto del botón para ir a la página anterior
    $.mobile.toolbar.prototype.options.backBtnText = "Atrás"
})

//Esto se ejecuta una sola vez para cada página, una vez inicializada
//por jQuery (añadidas las clases CSS, ...)
$('#pagBusqueda').on('pageinit', function() {
    $('#botonBuscar').click(buscar)
})

//Aquí rellenaríamos la página "detalles" con los datos pertinentes,
//justo antes de mostrarla
$('#pagDetalles').on('pagebeforeshow', function() {
    ...
})
```