Mobile Web Appplications Development with HTML5

Lecture 2: Building a Mobile Web App

Claudio Riva

Aalto University - Fall 2012

Lecture 2: Building a Mobile Web App

- Our first mobile web app
- New HTML5 form semantics
- Local storage
- Offline usage

Our first mobile web app

Our goal for today is:

- Create a mobile ready web page
- Add a few entry forms
- Style the forms with CSS
- Implement the server
- Rewrite the app with jQuery Mobile
- Add local storage
- Make the app available offline

Viewport Meta Tag


```
<meta name="viewport" content="width=device-width; user-scalable=0;" />
```

Viewport Meta Tag

- Introduced by Apple with the iPhone for mobile browsing
- Specify exactly the dimension of the page to render
- Not part of HTML5 but will be incorporated in the future
- More info at Safari Development website.

```
<meta name="viewport" content="width=device-width, user-scalable=0, initial-scale=1.0, maximum-scale=1.0;" />
```

Hiding the URL bar

On some browser the URL bar is always visible

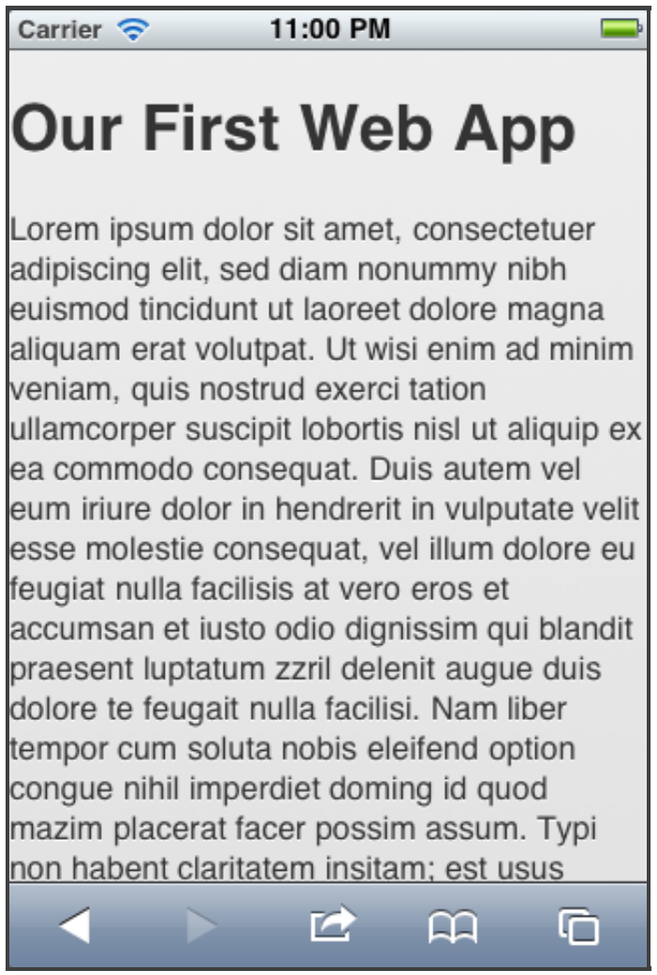
UI frameworks (like jQuery and Sencha) automatically hide the bar using the following tricks

Scroll to the top of the page after the browser has finished to load the page

```
<body onload="window.scrollTo(0, 1);">
```

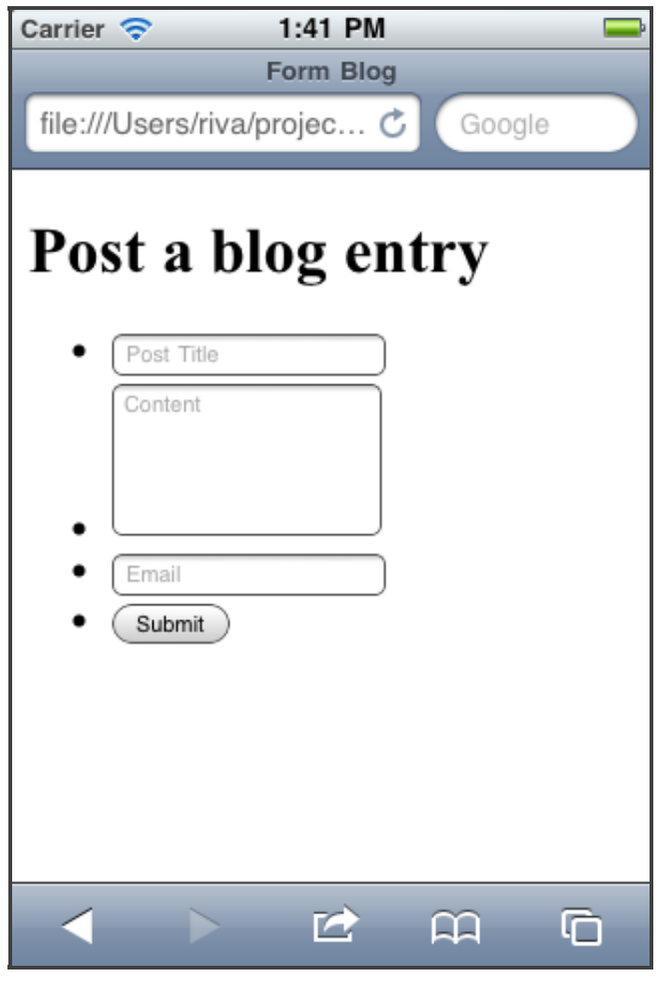
On iOS there are specific tags:

```
<meta name="apple-mobile-web-app-capable"
content="yes" />
<meta name="apple-mobile-web-app-status-bar-
style" content="black" />
```



Basic Form for Posting to a Blog

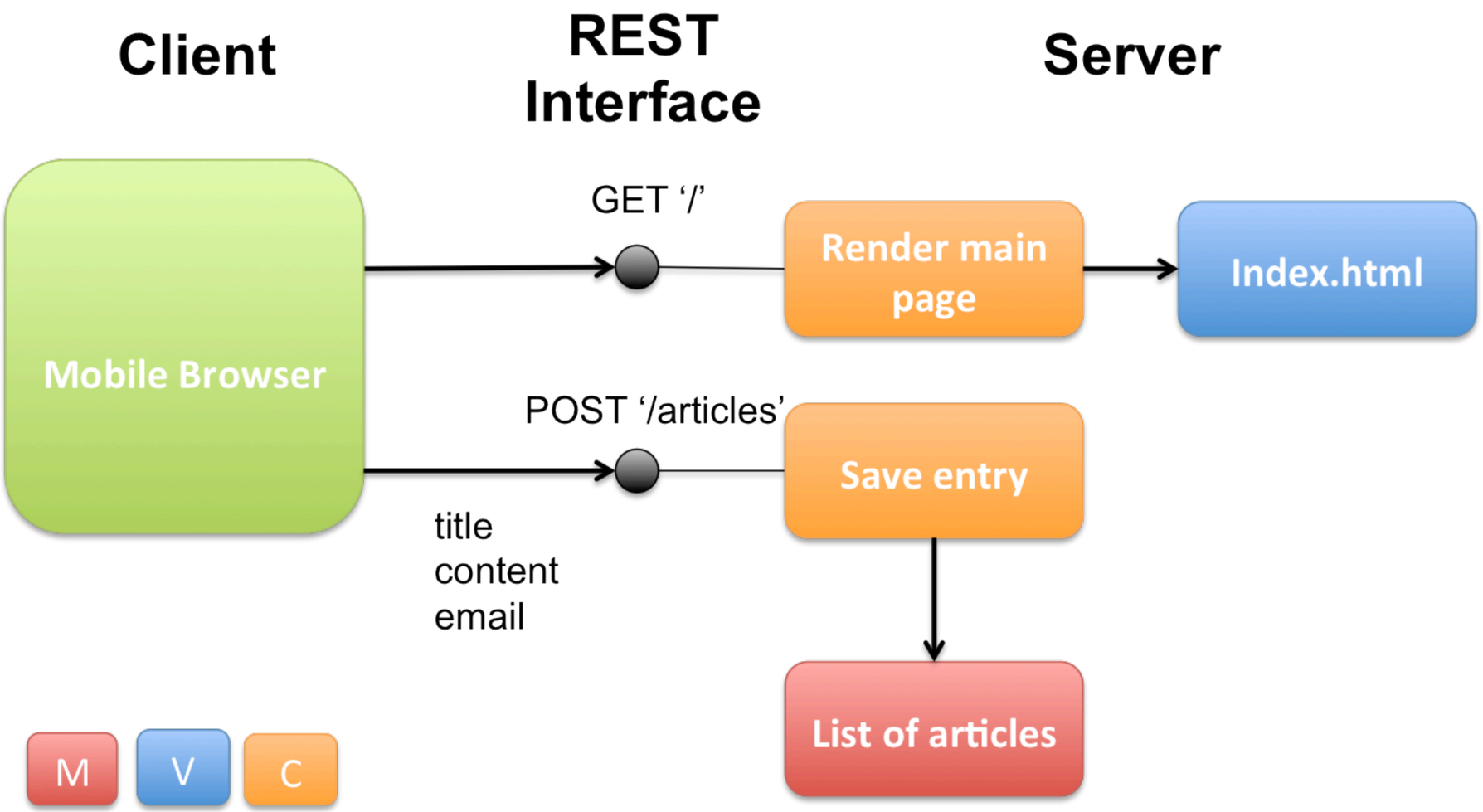
```
!!! 5
%html
  %head
    %meta(charset="utf-8")
    %meta(content="IE=edge,chrome=1" http-equiv="X-UA-Compatible")
    %meta(name="viewport" content="width=device-width, initial-scale=1")
    %title
      Form Blog
  %body
    %h1 Post a blog entry
    %form
      %ul
        %li
          %input(type = "text" name="title" id="title" placeholder="Post Title")
        %li
          %textarea(name="content" rows = "5" placeholder = "Content")
        %li
          %input(type = "email" placeholder = "Email")
        %li
          %input(type = "submit" name = "Post")
```



Adding Some Style

```
body {font-family: Arial; margin: 0px;}
h1 {
 background: -webkit-gradient(linear, left top, left bottom, color-stop(0.0, #666),
    color-stop(1.0, #000));
  -webkit-box-shadow: 0 0 4px #333;
  color: #EEE;
  margin: 0 0 4px 0; padding: 4px; width: 100%;
  text-align: center;
}
form ul { padding: 6px; margin: 0px; list-style-type: none; }
form ul li {
  margin: 0 0 4px 0; -webkit-border-radius: 4px;
  border: 1px solid #666; padding: 4px;
}
input, textarea {
  -webkit-appearance: none;
  border: 0; width: 95%;
}
form ul li.noborder { border: 0; padding: 0px;}
input[type=submit] {
  border: 0;
  background: -webkit-gradient(linear, left top, left bottom, color-stop(0.0, #EEE),
    color-stop(1.0, #AAA));
  -webkit-border-radius: 6px;
  -webkit-box-shadow: 0 0 4px #333;
  width: 100%; padding: 6px;
}
```

Basic Design



A Simple Server with Sinatra htmlFormBlog

```
require 'sinatra'
require 'json'

$articles = []

def timestamp
  Time.now.strftime("%H:%M:%S")
end

get '/' do
  File.read('index.html')
end

get '/articles' do
  content_type :json
  {:articles => $articles}.to_json
end

post '/articles' do
  article = params.merge( {'timestamp' => timestamp}).to_json
  puts article
  $articles << article
end
```

jQuery Family

jQuery Library

- Library for manipulating, traversing the DOM and animations.
- Check also Zepto.js

jQuery UI

- Customizable UI library for websites

jQuery Mobile

- HTML5-based user interface for mobile devices
- Lightweight, flexible and themeable design.
- 3-level graded platform support system
- Check out the gallery

Custom Data Attributes more info

Custom data attributes allows us to embed data with the page

Any element attribute that start with **data-** is treated as a storage area for private data

Custom data attributes do not affect the rendering

```
<div class="title" data-name="Hello" data-lang="en" data-city="Helsinki">
  Hello World
</div>
```

Key elements of jQuery Mobile

Single page template and Multi-page template

Page transitions

Ajax Navigation

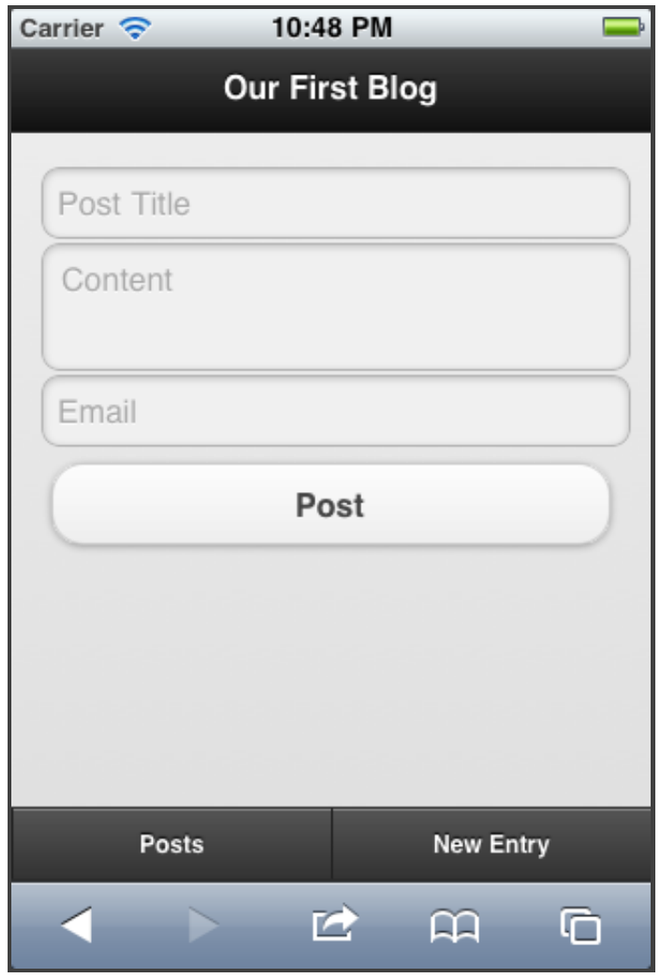
- First page is requested via HTTP request
- Subsequent pages are loaded via Ajax and injected into the DOM (only the "page" section)

Enhanced form elements

Several list views

Events and Data attributes

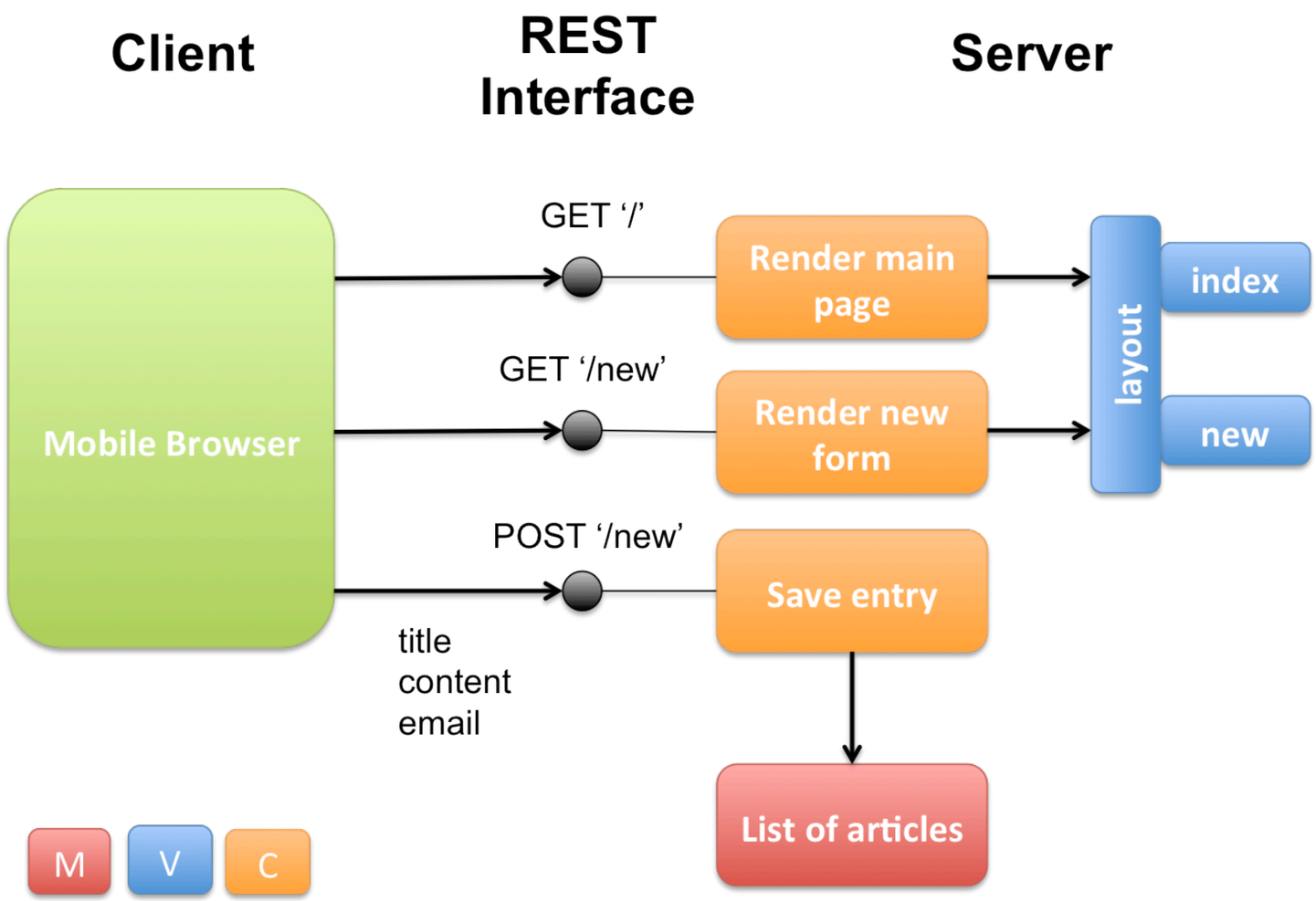
A New Design with jQuery mobile htmljQueryBlog



jQuery Mobile Page


```
%div(data-role="page" id="home")
  %div(data-role="header")
    %h1 Our First Blog

  %div(data-role="content")
    = yield

  %div(data-role="footer”)
    %div(data-role="navbar")
```

Design with Templates



Sinatra Server with Templates htmljQueryBlog

```
require 'sinatra'
require 'json'
require 'haml'

$articles = [{:title => "Welcome", :content => "My first post", :email => "hello@blog.com", :timestamp => "1.1.2012 10:20:30"}]

def timestamp
  Time.now.strftime("%d.%m.%Y %H:%M:%S")
end

get '/' do
  #Renders the haml template index.html.haml
  #with the default layout layout.html.haml
  haml :index, :layout => :layout
end

get '/new' do
  haml :new
end

post '/new' do
  #Symbolize the params keys
  article = params.inject({}) { |h,(k,v)| h[k.to_sym] = v; h }

  article[:timestamp] = timestamp
  $articles << article

  puts article
  redirect to ("/")
end
```

The Layout ('views/layout.haml')

```
!!! 5
%html
  %head
    %meta(charset="utf-8")
    %meta(content="IE=edge,chrome=1" http-equiv="X-UA-Compatible")
    %meta(name="viewport" content="width=device-width, initial-scale=1")
    %link(rel="stylesheet" href="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.css")
    %script(type="text/javascript" src="http://code.jquery.com/jquery-1.8.2.min.js")
    %script(type="text/javascript" src="http://code.jquery.com/mobile/1.2.0/jquery.mobile-1.2.0.min.js")
    %title
      Form Blog
  %body
    %div(data-role="page" id="home")
      %div(data-role="header")
        %h1 Our First Blog

      %div(data-role="content")
        = yield

      %div(data-role="footer" data-position="fixed")
        %div(data-role="navbar")
          %ul
            %li
              %a(href="/" data-ajax="false") Posts
            %li
              %a(href="/new" data-ajax="false") New Entry
```

The Views htmljQueryBlog

List of posts ('views/index.haml')

```
%ul(data-role = "listview")
  - $articles.each do |a|
    %li
      %p.ui-li-aside
        =a[:timestamp]
      %h3
        =a[:title]
      %p
        %strong
          =a[:email]
      %p
        =a[:content]
```

New Post Form ('views/new.haml')

```
%form(action="/new" method="post")
  %input(type = "text" name="title" id="title" placeholder="Post Title")
  %textarea(name="content" rows = "5" placeholder = "Content")
  %input(type = "email" name="email" placeholder = "Email")
  %button#postEntry(type = "submit") Posts
```

Form Validation

CSS3 introduces two new pseudo-classes for input elements:

- **:valid** : element is valid according to the input's type setting (e.g. email)
- **:invalid** : element failed to validate according to the input's type

HTML5 also introduced the **required** attribute that indicate the field must have valid data before it can be submmitted.

Styling for invalid elements is done with CSS:

```
:invalid {
  background-color: #ffdddd;
}
```

```
$(":invalid").addClass('red');
```

Validation htmljQueryValBlog

CSS

```
[required] {
  border-color: #88a;
  -webkit-box-shadow: 0 0 3px rgba(0, 0, 255,
.5);
}
:invalid {
  border-color: #e88;
  -webkit-box-shadow: 0 0 5px rgba(255, 0, 0,
.8);
}
input.invalid {
  background: url(error.png) no-repeat 4px 6px;
  padding-left: 30px;
}
```

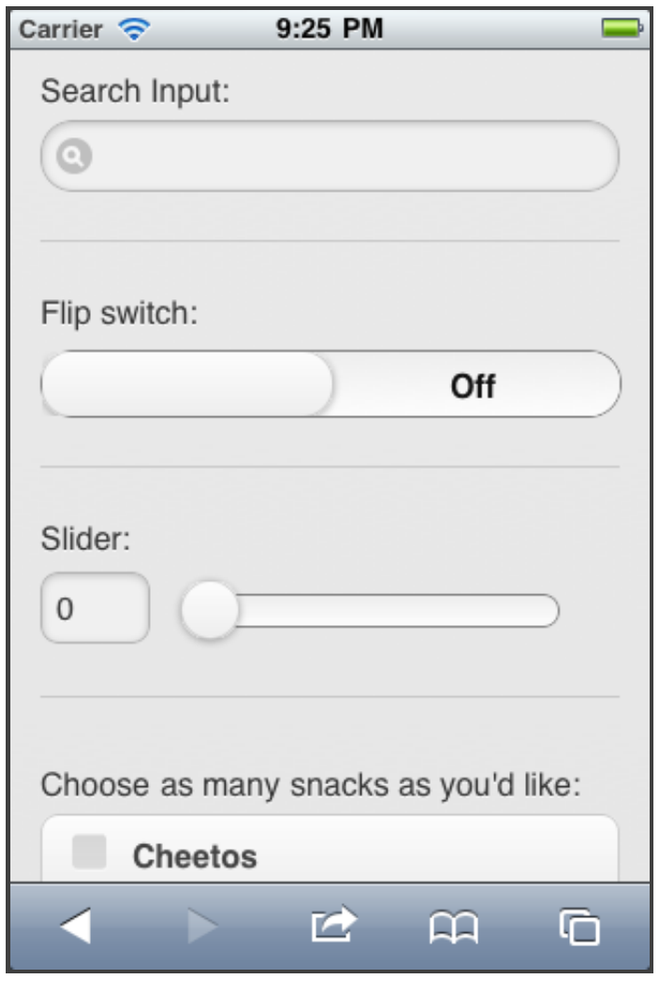
JS

```
$(function() {
  $("#postEntry").bind("click", function() {
    $(".invalid").removeClass("invalid")
    if ($(":invalid").length) {
      $(":invalid").addClass("invalid");
      return false;
    }
  })
})
```

Form

```
%form(action="/new" method="post")
%input(type = "text" name="title" id="title" placeholder="Post Title" required)
%textarea(name="content" rows = "5" placeholder = "Content")
%input(type = "email" name="email" placeholder = "Email")
%button#postEntry(type = "submit") Post
```

New Form Input Types



Overview at HTML5 Rocks

More details at W3Schools

Mobile version from jQuery Mobile

Contenteditable Attribute

The contenteditable attribute specifies whether a user can edit the content of an element.

This paragraph is editable.

```
<p contenteditable='true'>This paragraph is editable</p>
```

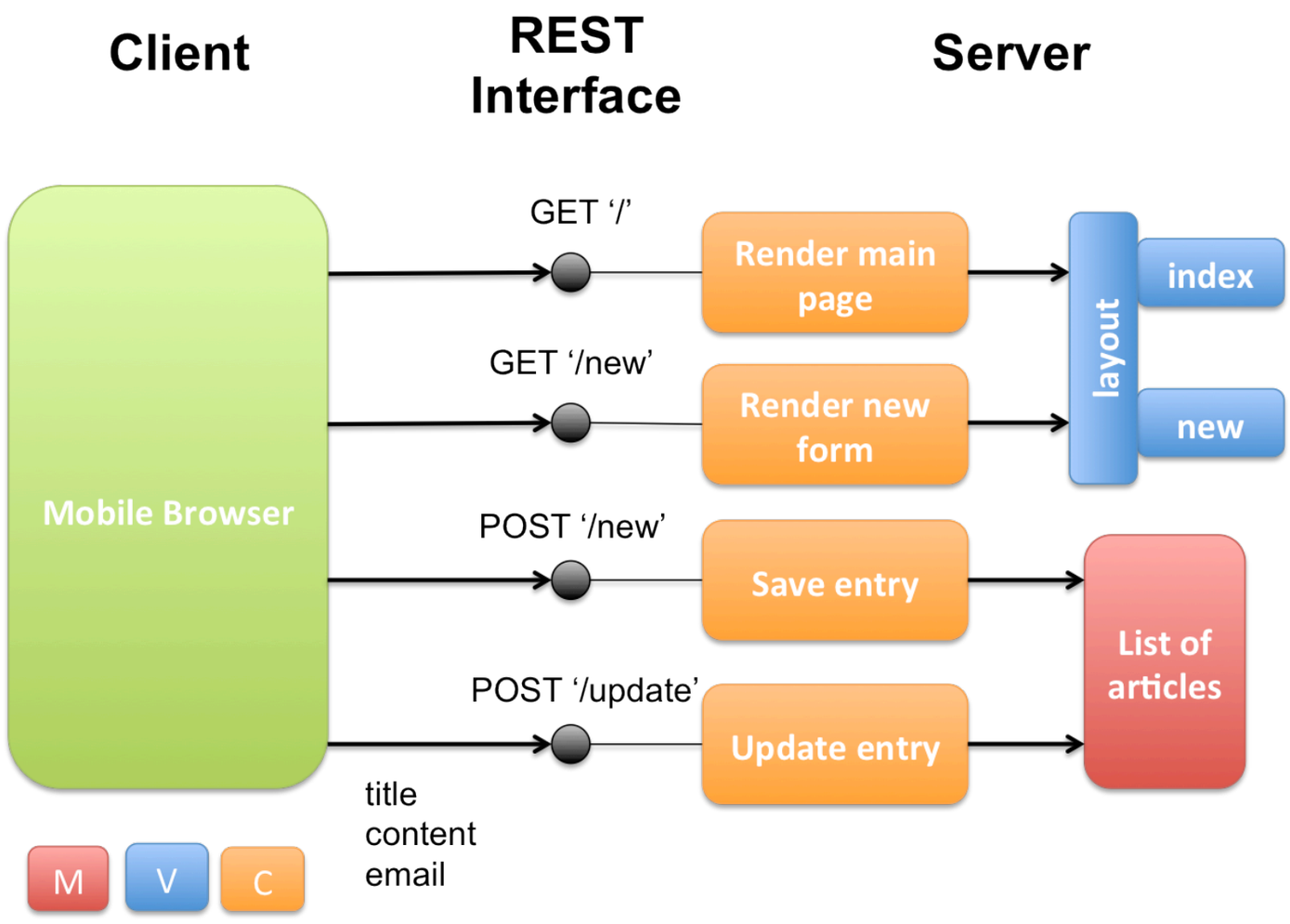
Supported in iOS 5 and Android Browser 3.0 (Canluse)

Editable Blog htmljQueryValEditBlog

List of posts ('views/index.haml')

```
%ul(data-role = "listview")
  - $articles.each_with_index do |a, i|
    %li
      %p.ui-li-aside
        =a[:timestamp]
      %h3(contenteditable="true" data-name="title" data-id='#{i}')
        =a[:title]
      %p
        %strong
          =a[:email]
      %p(contenteditable="true" data-name="content" data-id='#{i}')
        =a[:content]
```

Design supporting updates



Editable Blog htmljQueryValEditBlog

javascript code

```
$("[contenteditable]").focus(function() {
  var $this = $(this);
  $this.data('before', $this.html());
  return $this;
});
$("[contenteditable]").live('blur', function() {
    var $this = $(this);
    if ($this.data('before') !== $this.html()) {
        $this.data('before', $this.html());
        $this.trigger('change');

        var msg = {};
        msg['id'] = $this.attr('data-id');
        msg[$this.attr('data-name')] = $this.html();
        $.post('/update', msg);
    }
    return $this;
});
```

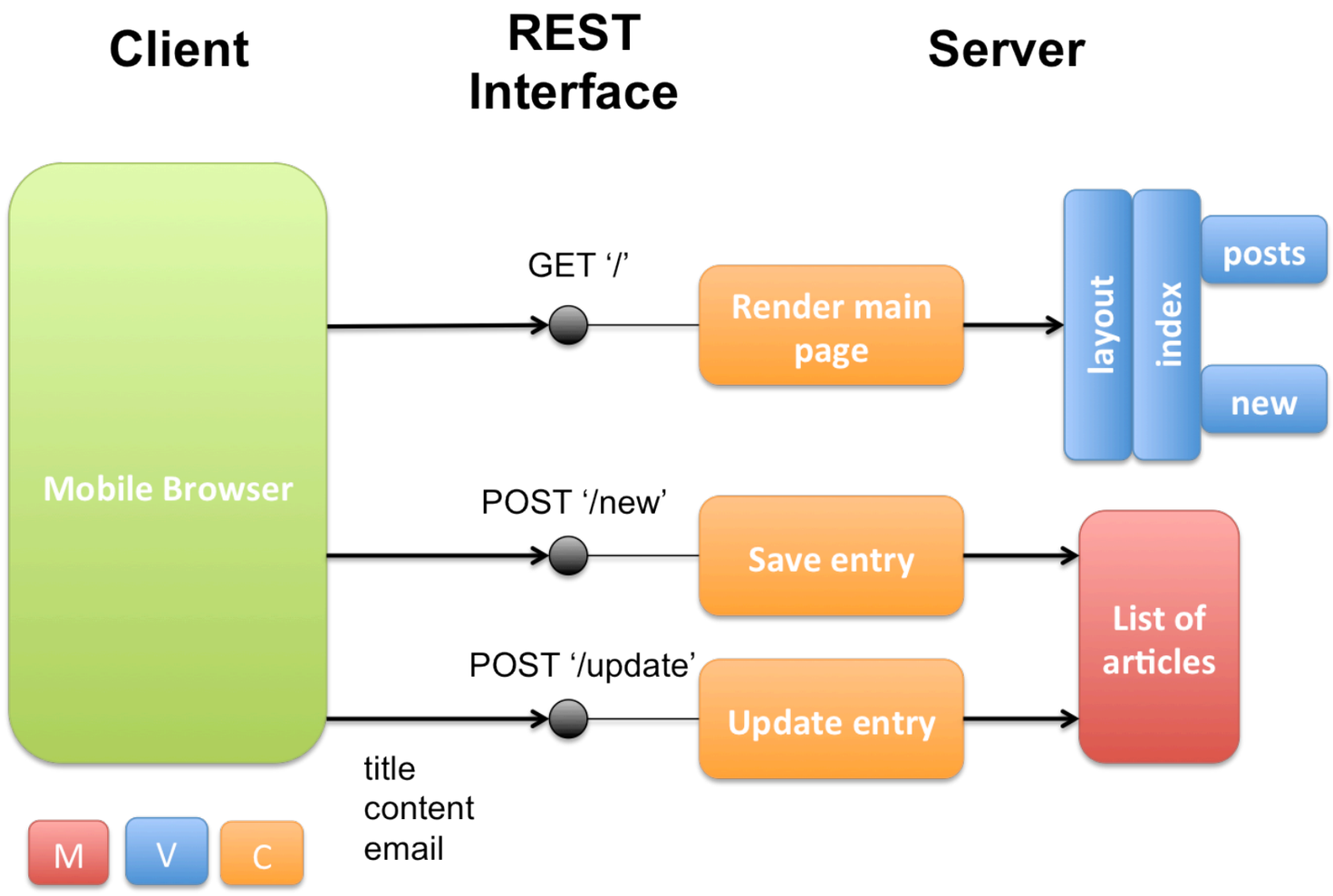
server.rb

```
post '/update' do
  #Symbolize the params keys
  article = params.inject({}) {|h,(k,v)| h[k.to_sym] =
v; h}

  #Update the timestamp
  article[:timestamp] = timestamp

  #Replace the entry in the list of articles
  $articles[article[:id].to_i].merge!(article)

  puts $articles
end
```

Create a multipage app with jQuery

- One single HTML document can contain multiple pages
- Each page must have a unique ID
- The framework handles the page switching and updating the url hash
- Link behaviour:
  - If it's an internal page, there is a transition to that page
  - If it's an external page, the default behaviour is to trigger an ajax request to load the page. Only the content of the `body` (of of the `body-role="page"` if present) is injected into the DOM
  - If the attributes `rel="external"` or `data-ajax = "false"` are present, the link triggers a full page reload. links with external attribute will cause a full page refresh

Design with multiple JQM pages



A multipage Blog htmljQueryMultiBlog

layout.haml

```
%body
  = yield
```

index.haml

```
%div(data-role="page" id="home")
  = haml :header

  %div(data-role="content")
    = haml :posts

  = haml :footer

%div(data-role="page" id="new")
  = haml :header

  %div(data-role="content")
    = haml :new

  =haml :footer
```

new.haml

```
%form(action="/new" method="post")
  %input(type = "text" name="title" id="title"
placeholder="Post Title" required)
  %textarea(name="content" rows = "5" placeholder =
"Content")
  %input(type = "email" name="email" placeholder = "Email")
  %button#postEntry(type = "submit") Post
```

header.haml

```
%div(data-role="header")
  %h1 Our First Blog
```

footer.haml

```
%div(data-role="footer" data-position="fixed")
  %div(data-role="navbar")
    %ul
      %li
        %a(href="/" data-ajax="false") Posts
      %li
        %a(href="#new" data-transition="flip") New Entry
```

posts.haml

```
%ul(data-role = "listview")
  - $articles.each_with_index do |a, i|
    %li
      %p.ui-li-aside
        =a[:timestamp]
      %h3(contenteditable="true" data-name="title" data-id='#
{i}')
        =a[:title]
      %p
        %strong
          =a[:email]
      %p(contenteditable="true" data-name="content" data-id='#
{i}')
        =a[:content]
```

Offline Usage

For a web application to work offline, you need to consider the following:

- Store required data locally through **localStorage**
- Define what files to cache locally via a **manifest file**
- Manage connection changes with **online and offline events**
- Define a **synchronization** strategy with the data stored in the cloud

Offline Storage

- Historically browsers (thin clients) do not support a way to store data.
- Cookies allow to attach data to the HTTP requests but:
  - Cookies are sent back and forth with every request
  - Cookies cannot be share across different subdomains
  - Limited storage (20 cookies x 4KB)
- Local storage in the browser gives the advantage of:
  - Reduce number of requests to the server
  - Store locally static or semi static data
  - Enable offline usage (e.g. for mobile devices)

Offline Storage

HTML5 introduces two key-value storing mechanisms:

- **localStorage**: persistent storage across browser restarts
- **sessionStorage**: storage that resets when a browser session restarts

```
//set a key
localStorage.myData = "Hello World";
localStorage.setItem('myData', "Hello World");

//get a key
var data;
data = localStorage.myData;
data = localStorage.getItem('myData');

//delete a key
localStorage.myData = undefined;
localStorage.removeItem('myData');
```

Offline Storage - Benefits

- Same origin policy
- approx 2-5MB of storage per host (test)
- Only strings can be stored (UTF-16)
- Covert to JSON before storing
- Browser provide tools for looking at the local storage
- Performance (Check this) is better than cookies but not the fastest (depends on browser impementation)

Add Options to the Blog

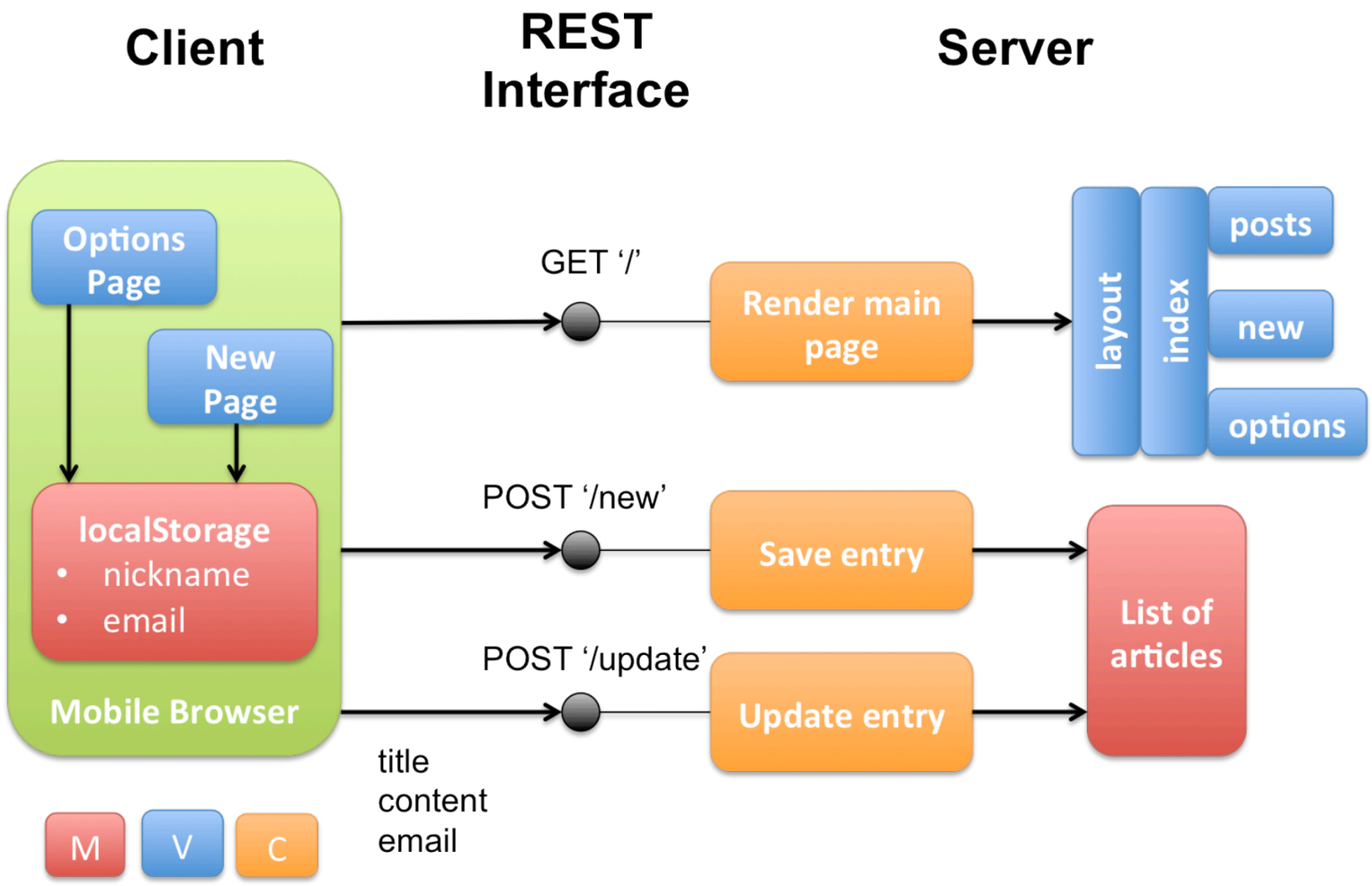
Add Options for storing:

- Name
- E-mail

The personal data is stored in the localStorage

When posting a new entry the name and email are also included.

Design with localStorage



Add Options to the Blog htmljQueryStorageBlog

index.haml

```
...
%div(data-role="page" id="options")

  =haml :options

  =haml :footer
```

options.haml

```
%div(data-role = "header" data-position="inline")
  %a(href = "#home" data-icon="delete" data-rel="back") Cancel
  %h1 Options
  %a(href = "#home" data-icon="check" data-theme="b" id="saveOpt" data-rel="back") Save

%div(data-role="content")
  %form
    %input(type = "text" name="name" id="optName" placeholder="Name")
    %input(type = "email" name="email" id="optEmail" placeholder = "Email")
```

new.haml

```
%form
  %input(type = "text" name="title" id="title" placeholder="Post Title" required)
  %textarea(name="content" rows = "5" placeholder = "Content" id="content")
  %button#postEntry(type = "submit") Post
```

Add Options to the Blog htmljQueryStorageBlog

JS

```
$(function() {
  // Assign the options from the local storage
  $("#optName").val(localStorage.name);
  $("#optEmail").val(localStorage.email);

  $("#postEntry").bind("click", function() {
    e.preventDefault();
    $("#new .invalid").removeClass("invalid")
    if ($("#new :invalid").length) { $("#new :invalid").addClass("invalid"); return false; }
    var msg = {};
    msg['title'] = $('#title').val();
    msg['content'] = $('#content').val();
    msg['email'] = localStorage.email;
    msg['name'] = localStorage.name;
    $.post('/new', msg, function() { $.mobile.changePage("/", {reloadPage: true}); });
  });

  ...
  //Save the options in the local storage
  $("#saveOpt").bind("click", function() {
    $("#options .invalid").removeClass("invalid")
    if ($("#options :invalid").length) {
      $("#options :invalid").addClass("invalid");
      return false;
    }
    localStorage.name = $("#optName").val();
    localStorage.email = $("#optEmail").val();
  })
})
```

Manifest File

- The manifest file tells the browser what files to cache locally for offline usage (HTML, CSS, Javascript, images, video, etc.)
- Every single resource that you want to cache must be explicitly listed (no wildcards).
- The files listed in the manifest will be loaded from the local disk even if the browser is online.
- If the page is loaded from the application cache (either online or offline), the browser will load the assets ONLY from the cache and ignore external resources (unless they are specified in the NETWORK section)

Cache size

- Safari desktop browser (Mac and Windows) have no limit
- Mobile Safari had 10MB limit and with iOS6 it has been increased to 25MB
- Chrome has a 5MB limit
- Android browser has no limit
- Firefox desktop has no limit
- Opera’s cache limit can be managed by the user, but has a default size of 50MB

Manifest File htmljQueryOfflineBlog

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

```
CACHE MANIFEST
# 30 January 2012
/index.html
/jquery.js
/main.css
/logo.png

NETWORK:
/api

FALLBACK:
/about /offline-about.html
```

```
mime type: text/cache-manifest
```

- Recommended file extension is `.appcache`
- Only HTTP GET requests are cached.
- The NETWORK section contains resources that are always fetched from the network
- The FALLBACK section contains resources that are fetched from the network only if the browser is online. Otherwiser the fallback version is used.

Manifest File - Rules

- When a page is loaded the browser will check the manifest on the server. If it has changed, the browser will redownload all the files listed in the manifest, update the cache and fire the event `applicationCache.onUpdateReady`
- The browser retains the cached assets until either the user clears the cache or you trigger an update.
- Triggering an update requires that the content of the manifest file changes (not just the assets).
- Any document referencing the manifest file is cached

Good explanation is available here

Manifest File - Network Whitelisting

- Once the manifest file is in place, the browser will route all the requests throught that file (both online or offline).
- If there is no rule in the manifest file that his matching a request, the request will fail.
- In the NETWORK section, the rule '*' whitelist all the requests that are not matching the rules in the CACHE section

```
CACHE MANIFEST
# 30 January 2012
/index.html
/jquery.js
/main.css
/logo.png

NETWORK:
/api
*

FALLBACK:
/about /offline-about.html
```

Making life easier with Manifesto htmljQueryOfflineManifestoBlog

- Manifesto automatically generate the manifest file by listing all files in the specified directories and subdirectories
- By default it includes `/public`
- It also computes a hash of the files's contents, so that if a file is changed the manifest file is automatically invalidated.

```
get '/manifest.appcache' do
  headers 'Content-Type' => 'text/cache-manifest'
  Manifesto.cache
end
```

```
CACHE MANIFEST
# Generated by manifesto
(http://github.com/johntopley/manifesto)
# Hash: 48c5fbdc2a7627233a5f9741fec32a77
/jquery.mobile.structure-1.0.1.min.css
/jquery.mobile.structure-1.0.1.css
/jquery.mobile-1.0.1.min.js
/jquery.mobile-1.0.1.min.css
/jquery.mobile-1.0.1.js
/jquery.mobile-1.0.1.css
/jquery-1.7.1.min.js
/images/icons-36-white.png
/images/icons-36-black.png
/images/icons-18-white.png
/images/icons-18-black.png
/images/ajax-loader.png
/error.png
```

How to refresh the list of posts

- The main HTML page of the web app is also cached (because it is referencing the manifest file)
- The page will be refreshed only when the manifest file expires in the cache of the browser and is changed on the server.

=> Dynamic updates require a new design for our blog

The applicationCache interface

```
window.applicationCache  //Main object to access the app cache
window.applicationCache.status //Status
window.applicationCache.onupdateready = function (e) {...};  //Callback
window.applicationCache.update();  //Force cache update
window.applicationCache.swapCache();  //Swap to new cache
```

State of the cache:

- 0: UNCACHED
- 1: IDLE
- 2: CHECKING
- 3: DOWNLOADING
- 4: UPDATEREADY
- 5: OBSOLETE

Event callbakcs:

- onchecking
- ondownloading
- onupdateready
- onobsolete
- oncached
- onerror
- onnoupdateready
- onprogress

Browser connectivity

Mobile apps must detect if the browser is online or offline to provide the best user experience:

- disabling those parts of the UI that can only work online
- caching offline actions
- synchronizing the offline changes with the server

Detecting connectivity with the manifest htmljQueryOfflineDetectBlog

layout.haml

```
$(function() {
  function testOnline() {
    $.getJSON('ping', function(data) {
      $('.status').html(data);
    });
  }
//Test at first page load
testOnline();
//Test a timer for infinite testing
window.setInterval(testOnline, 10000);
```

header.haml

```
%div(data-role="header")
  %h1.status
  %a(href="#options" data-icon="gear" class="ui-btn-right" data-transition="slidedown") Options
```

Detecting connectivity with the manifest htmljQueryOfflineDetectBlog

manifest.appcache

```
...
FALLBACK:
/ping /ping-offline.json
```

ping-offline.json

```
"Offline"
```

server.rb

```
get '/manifest.appcache' do
  headers 'Content-Type' => 'text/cache-manifest' # Must be served with this MIME type
  cache_control :no_cache #Disable caching of manifest file, only for demo purposes
  Manifesto.cache << "FALLBACK:\n /ping /ping-offline.json\n"
end
get '/ping' do
  content_type :json
  "Online".to_json
end
```

IndexedDB

Storage mechanism for unstructured data objects (NoSQL)

Key-value storage where values can be complex structure objects.

Good performance (better than localStorage)

Asynchronous API (not blocking UI thread) and synchronous for web workers

Based on a transactional model

Support for data migrations