Mobile Applications Professional Education Series

Brian Davidson
Research Network Operations Center

MOBILE WEB APPLICATIONS
DEVELOPMENT TOOLS
How to access the DevTools

To access the DevTools, open a web page or web app in Google Chrome. Then take one of the following actions:

- Select the **Chrome menu** at the top-right of your browser window, then select **Tools > Developer tools**.
- Right-click on any page element and select **Inspect element**.
How to access the DevTools

To access the DevTools, open a web page or web app in Google Chrome. Then take one of the following actions:

- Select the Chrome menu at the top-right of your browser window, then select Tools > Developer tools.
- Right-click on any page element and select Inspect element.
CONSOLE

Working with the console

The JavaScript `Console` provides two primary functions for developers testing web pages and applications:

- A place to log diagnostic information using methods provided by the `Console API`, such as `console.log()`, or `console.profile()`.
- A shell prompt where you can enter commands and interact with the document and the Chrome DevTools. You can evaluate expressions directly in the Console, and can also use the methods provided by the Command-Line API, such as `$()` command for selecting elements, or `profile()` to start.

```
> event.returnValue is deprecated. Please use the standard event.preventDefault() instead.
```

Copyright 2014 Russell J Clark
LANGUAGES & TECHNOLOGIES
Rendering Languages

• Hypertext Markup Language (HTML): used for writing web pages
• Cascading Style Sheets (CSS): stylistic info for web pages
• JavaScript: interactive and programmable web pages
Communication Languages and Technologies

• Asynchronous JavaScript and XML (Ajax): accessing data for web applications
• eXtensible Markup Language (XML): metalanguage for organizing data
• Javascript object notation (json): metalanguage for organizing data
Backend Languages

• PHP Hypertext Processor (PHP): dynamically create content on a standard web server
• Python: dynamically create content on a standard web server
• Javascript (Node.js): Create your own webserver and content that responds on the webserver
• Ruby (on Rails): Use default rails webserver (WEBrick) and dynamically content that is server
WEBSITES VS WEB APPS
Websites
Reactive Websites

When thinking about mobile, the first thought is to take the existing website and just make it "Reactive". Have the layout adjust so the existing site looks better on mobile.
Reactive Websites

Which is a good thing to do but doesn't mean it is the only thing that should be done in all cases.

For more, we need to look at what it means to visit a website.
Visiting a Website

When you request a webpage it sends a http request to the web server asking for that page

example.org
Web Server
Visiting a Website

GET / HTTP/1.1
Host: example.org
Connection: keep-alive
Cache-Control: no-cache
Pragma: no-cache
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US, en; q=0.8
Visiting a Website

example.org

Web Server

The server receives the messages processes it retrieving from disk and/or database as necessary
Visiting a Website

The server returns the response to the client

HTTP/1.1 200 OK

example.org

Web Server
Visiting a Website

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Thu, 16 Jan 2014 05:40:11 GMT
Etag: "359670651"
Expires: Thu, 23 Jan 2014 05:40:11 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (iad/1984)
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270
Visiting a Website

<!doctype html>
<html>
<head>
  <title>Example Domain</title>
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
    }
  </style>
</head>

Visiting a Website

<body>
<div>
<h1>Example Domain</h1>
<p>This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.</p>
</div>
</body>
<html>
Visiting a Website

The client receives the response, process the HTML, runs any Javascript and renders the page using CSS to modify the look and feel.
Visiting a Website (Simplified)

• Files are sent from server disk after processing
• Rendered on client
• Javascript runs on client to update the rendering
Visiting a Website (+Complexity)

Each resource used in the HTML must be requested from the web server.
Visiting a Website (+Complexity)

When you click on a link and load a new URL.
It starts all over

- Clears the screen
- Request new page from server
- Perform additional requests based on HTML
- Run Javascript and Render Page
Visiting a Website (+Complexity)
Web Apps

Problem

• Starting from scratch every link click is a waste of resources (network, cpu), slow, and causes interruptions to the user experience.

Solution

• Don't go back to the server for the whole new page, just go back for whatever you wanted changed.
Web Apps
Web Apps

Keynote for iCloud is a great example of a Web App. This presentation was made in keynote for iCloud. When I move between slides there is no network activity at all, the activity is only done when I make changes that need to be saved to the cloud or pull in changes made by collaborators from the cloud.
Web Apps
Visiting A Web App

The core part of a WebApp is AJAX. Javascript can talk to the Web Server to just send and receive the needed data and not have to reload whole page.
PROVIDING DATA TO APP
Why Have an API?

If the data is stored on disk or in a database. Why don't I just connect to the disk/database directly and get rid of the API?
Why Have an API?

If the client wanted to connect the client directly to the database, it would have to have the username and password to login. Furthermore all the logic that deals with the database would have to be downloaded by every client.
Why Have an API?

But Javascript source code is readable by any user in their browser.
Why Have an API?

A malicious person could look up the credentials and connect directly and perform his own actions

SELECT (username, password) FROM users
DELETE FROM users
Why Have an API?

If all the database code exist client side, then it’s harder to update the database structure as all clients will need to receive the updates or they could fail working and cause corruption.
Why Have an API?

Question:
• Could I do this if it was native instead of javascript?

Answer:
• Native code can always still be decompiled and the password still has to be stored on the client so it can still be retrieved
Providing Data To Your Applications

• There are many different ways to architect your application to leverage external data
• One of the most popular approaches today is to use a RESTful API implemented on a highly available network server
• So what is a RESTful API?
REST Leverages Standard Web Technologies

- **Web Server**: supports remote access using standard HTTP queries – e.g. Apache, IIS
- **HTTP**: the HyperText Transfer Protocol, designed for web sites but has become the *lingua franca* of network services
- **Web Programming Languages**: various languages like PHP, Python, Ruby, Java that are used to build web services
- **Databases**: provide the backend data storage such as MySQL, MongoDB, Oracle, etc
WEB APP DEVELOPMENT WORKFLOW
Web App Development Workflow

- **Come up with idea and use cases**
- Figure out type of application it is
- Design the Frontend (UI)
- **Choose a Frontend Framework based on needs**
- Program the Frontend (UI) in HTML
- Figure out what data needs to be stored
- Figure out what API calls you need based on use cases and UI and data
- Program the Backend (API)
- Program the Bridge (Javascript) between the Frontend (HTML) & Backend (API)
Mobile Application Types

• **Standalone App**: completely self-contained, no external data, never use the network, e.g. calculator, many single player games

• **Rich Client**: most features are local, some external data provided when available, e.g. address book

• **Rich Internet Application**: most features require remote data access to be useful, e.g. FaceBook, Voice Calling

• **Thin Client**: most all resources are remote, only the UI is local, e.g. Google search
Design the Frontend (UI)

Drag & Drop

Style

Tips & Tricks
Share this Mockup with others by sharing the URL.
Everyone who knows the URL can make changes. Changes will be synchronized live. To test this, open the URL in two different browser windows.

Double-click widgets to add or change text labels.

Upload

Select PNG

Upload
Choose a Framework

- jQuery Mobile
- Ionic
- The M Project
- famo.us
- Framework7
- Onsen UI
- CocoonJS
- Junior
- Jo
- Openui5
- LoopBack by StrongLoop
- Sencha Touch
- Web Starter Kit
- ChocolateChipUI
- Ratchet
- ImpactJS
- UI for PhoneGap
- Kendo UI
## Choose a Framework (Advanced)

### IDEs & Integrated Frameworks/Platforms
- Intel XDK
- Monaca
- AppGyver
- RhoMobile
- Appcelerator Titanium
- MoSync
- Marmalade

### Platforms
- PhoneGap
- Crosswalk
- Telerik Platform
- Trigger.io
Web App Framework Principles

• Don’t create multiple html pages
  – Must reload the DOM and reinitialize javascript every time

• Use fragment URLs to switch contexts so that forward and back navigation can be tracked, and allows easy switching

• Use templating technology to reliably create UI elements