A Guide to Beginning and Rockin' with PhoneGap

0. Get DUNS number first, then AppleDev account
To put an app on an iOS device (not the app store, just a physical iPhone), you need to have a developer account. To get a dev account, you can get one as an individual quickly but as a company (which I recommend), you need to get a DUNS number (which proves to Apple that you are a real business). This process is free and takes 30 days or $300 and takes 5 days. It is a rip-off but unavoidable. You will need INC entity tax docs, bank acct info, etc. Have someone start this process now. Get the ball rolling. This was numbered "0" intentionally to emphasize the need to do it now.
Get DUNS number here: https://iupdate.dnb.com/iUpdate/viewiUpdateHome.htm
Apple iOS Dev: https://developer.apple.com/programs/ios/

1. Build.PhoneGap + hydration
This allows you to code your html app on your desktop, test it first in your browser as a webpage, then upload a zip (or git pull) and build it in all the mobile native apps, all online. No messy SDKs for everything setup. No Xcode, or java Android SDK needed. It's cool. But a little tricky at first. They also have a tech called Hydration which will update the app on your phone when you rebuild new code using Build.PhoneGap. This is their version of TestFlight (mentioned below). Use it for testing and beta only. Remove (uncheck hydration in settings) for production.
http://build.phonegap.com/
http://phonegap.com/
You can then also put your project in GitHub, hook it up to this and press one button to pull and build your latest code. Cool potatoes.

2. UI Frameworks
I'm not saying don't use jQueryMobile, but that framework was designed for the web, not to be wrapped in PhoneGap. If you are going to use it, know exactly why and build a custom build with only the elements needed for your app. There are tons of people who say that all the bad opinions of PhoneGap are due to them using jQM and that it's slow, not PG. Any future apps I do with PG will not use jQM and will use some of these below.
Read this: http://sintaxi.com/you-half-assed-it
Top Coat (from Adobe) http://topcoat.io/
PageSlider http://coenraets.org/blog/2013/03/hardware-accelerated-page-transitions-for-mobile-web-apps-phonegap-apps/
Zurb Foundation http://foundation.zurb.com/
Bootstrap : http://twitter.github.io/bootstrap/
Bartender : http://www.stokkers.mobi/valuables/bartender.html
FastClick (solves delay problem on mobile; already included in jquerymobile) https://github.com/ftlabs/fastclick
Zepto : http://zeptojs.com/
Zepto page transitions : https://github.com/dgileadi/zepto-page-transitions
Fixed footer : http://css-tricks.com/snippets/css/fixed-footer/
Fixed footer example : http://www.flashjunior.ch/school/footers/fixed.cfm

3. Create your app in one html page
Having all of your 'views' or pages in one html page will save you tons of headaches involving cross-browser scripting, ajax errors etc. Since all of these views are just DIVs anyway, and get shown or hidden, it's much easier. Plus the whole app gets wrapped and built anyway, so separate pages do not buy you any download performance speed. Your native 'app' is all the resources it comes with, html, images, js libs, css - combined.

4. TestFlight
Super cool and free way to distribute your app to people without needing their physical iPhone. It's all done remotely and works very well. Their Dashboard UI doesn't match my mental model of how things are organized, but it works great and is very popular in the iOS crowd.
https://testflightapp.com/

5. CSS and Meta Tricks - Research and know about these.
They could/will save you days of UI debugging.

```
 -webkit-touch-callout: none;
 -webkit-user-select: none;
 -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
 -webkit-backface-visibility: hidden;  /*android nav flicker solution, could make android crash */
 html, body {height: 100%;}

<meta name="viewport" content="user-scalable=no, width=device-width, initial-scale=1">
<meta name="format-detection" content="telephone=no">
```

6. Adobe PhoneGap Optimization videos and articles
http://tv.adobe.com/watch/max-2013/optimizing-phonegap-applications/
http://stevengill.github.io/presentations/slides/max2013/index.html#/7 (slides from above screencast)
http://coenraets.org/blog/2013/05/top-10-performance-techniques-for-phonegap-applications/
article: http://www.mikedellanoce.com/2012/09/10-tips-for-getting-that-native-ios.html?m=1

7. Documentation is good most of the time, but scattered, support is spotty. Use these:
Watch the updates on this blog, which only get communicated here : http://build.phonegap.com/blog
v2.6 docs : http://docs.phonegap.com/en/
Latest version is 2.7 on build.pg

8. Don't try to do weird iframe cookie reads or other unnatural cross-domain web tricks.
If you can do it in Chrome/Safari on your desktop, then you are cool.

9. Consider UX
Wireframe all screens with functionality, view transition and error scenario callouts. I am using Moqups now, free and easy. https://moqups.com/

How iOS/Android navbars differ per platform (bottom or top conventions).
How many app icons and pixel density versions there are. (when you get to this let me know and I have other resources that will save you tons of time)
Think about the app like a responsive website. There are so many device screen-sizes, your UI needs to handle anything. But of course, you could and probably should consider having media-query breakpoints that resize text, UI elements etc that match the most common mobile devices out there.
With the PG app, you will need to handle the 'airplane mode' case, where your app doesn't fail, but alerts the user that there is no network access etc. This will be the first thing Apple tries and rejects your app for.

10 How to Brake the App Up into Manageable Pieces
Phase 1 : Simple phase 1, build it as a web app in webkit, make it work in desktop webkit browser. I would even put it on a secret public hidden part of a web server and test it on your phones.

Phase 2 : iOS only, use TestFlight to selected 'special' customers for feedback

Phase 3 : Fix stuff, add Android, put it in the iOS AppStore and go through that whole process.

11. Maintenance, future support.
Who knows this new tech? There will suddenly be a new mobile stack for someone to maintain.
Who's maintaining and testing it when the mobile companies release a new OS version?