Understanding Responsive Web Design (RWD) & Environment Aware Component Design

Version: 2013.11.21

Contents

- Contents
- Checklist
  - Planning
- Responsive Web Design Overview
  - What is responsive design?
    - When should responsive design be considered?
    - Architectural Strategies
    - Key Ingredients of Responsive Design
    - Prescriptive vs. Responsive design
    - Determining Major Breakpoints
  - Cost reduction strategies for responsive web design
- Beyond Responsive Design
  - Environment Awareness
  - Environment-aware Conditional Loading
  - Device-Awareness and Feature Detection
  - IBM Worklight Features for Environment Optimization
    - Environment Optimization
    - Skin

Checklist

Planning

- ❑ Make sure to include design, development and test work items for apps that need to operate in more than one orientation.
- ❑ Use responsive design early in the project lifecycle. Trying to retrofit responsive techniques late in a project is very expensive.
- ❑ Define a cost reduction strategy and plan for minimizing costs due to supporting multiple devices, operating systems.
- ❑ Start with small form factor screen designs, and work outward toward larger form factors—this is “mobile-first” responsive design. Starting with large form factors and going the other direction incurs large refactoring costs.
- ❑ Do not waterfall the design process – responsive design requires designer and developer collaboration.
- ❑ Determine your breakpoints early in the project
- ❑ When determining supported breakpoints, make sure to consider vertical breakpoints in addition to horizontal breakpoints.
- ❑ Make sure to design for orientation changes

Responsive Web Design Overview

What is responsive design?

It is an essential technique for adjusting application content to the wide variations in screen sizes and different orientations.

Figure 1

- Minimum screen resolution cause clipped viewport
- Devices becoming smaller *and* larger simultaneously
    - Small: Mobile devices, smart phones
    - Large: Web gets into Game Console, TV
    - Somewhere in between: Tablet
- We are designing for more:
    - Devices
    - Input types
    - Resolutions
    - Orientations
- Reduce development costs
- 30% ~ 50% less coding (as compared to pixel perfect approaches)

Example: Application Layouts on iPhone vs. iPad form factors

Figure 2

When should responsive design be considered?

Responsive design pattern should be considered even in the early phase of User Interface/ Experience (UX) design

- Design UX component that can be easily expand or subtracted. Etc. List, Grid
- Design/Wireframe one flow (use case) for multi-form factor (phone -> tablet)

Architectural Strategies

Multi-channel rendering

- Can facilitate the rendering of multiple channels via Responsive Design, Feature Detection, and the Worklight Optimization framework.

Mobile First

- Build on form factor (phone or tablet) as first channel.
- Extend to support other devices and form factors

Responsive Architecture

- Responsive Architecture is a piece of architecture and its inhabitants to mutually influence and inform each other
- Responsive Design is the technique of designing web sites/apps that are not only more flexible, but that can adapt to the media that renders them.

Key Ingredients of Responsive Design

- Flexible size (including fonts)
- Fluid layout
- Responsive images and media
- CSS3 Media queries for responsive layout

Prescriptive vs. Responsive design

Prescriptive design, the traditional design approach which uses absolute pixel positioning in design mockups targets specific device form factors without considering multi-channel or multi form factors.

Using prescriptive design usually ends up with developers creating functional components for each device type or form factors, instead of reusing most of the existing components with Responsive Design and thinking about building components that must have flexibility when used in different environments and container sizes.

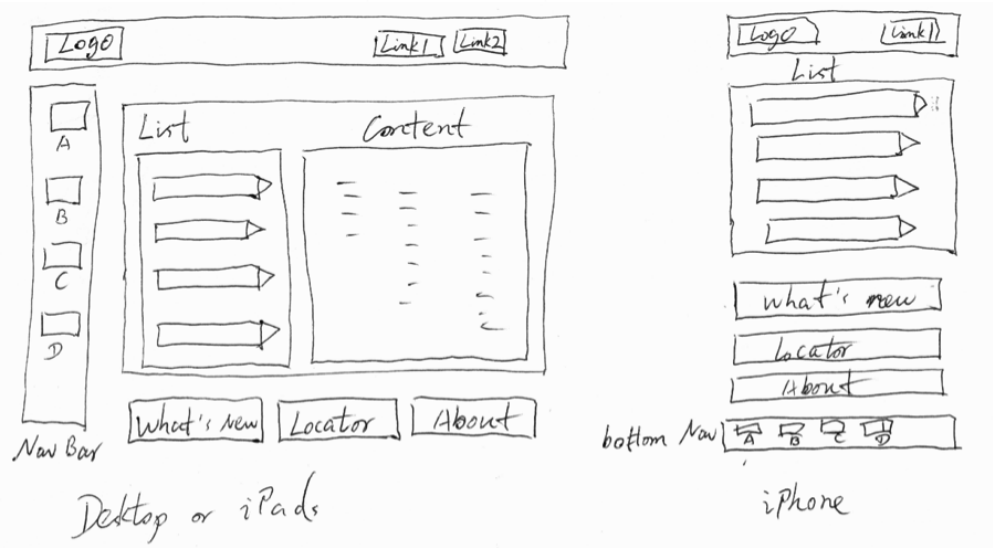
Determining Major Breakpoints

It is important early on in a project to decide upon the major "breakpoints" that your app (or company's apps) will need to support. For example, here are some horizontal breakpoints that typically are used:

- Smartphone width ***<480px***
- Tablets portrait width ***<768px***
- Tablets landscape width ***>768px***
- Desktop ***>1024px***

When determining supported breakpoints, make sure to consider vertical breakpoints in addition to horizontal breakpoints.

Example during rough mockup activity



Cost reduction strategies for responsive web design

It is important that you develop a business strategy around responsive design to minimize costs over time. Starting with different form factors and working toward other form factors in subsequent iterations can have dramatic differences in costs, depending on if you start small and work to larger or vice versa.

There are many dimensions that incrementally affect costs that may not be initially apparent when developing mobile apps across multiple device types and channels. Figure 1 describes the most common cost factors.

When targeting many device types or channels (commonly referred to as “multichannel”), there are many cost factors include designing, developing and testing the application to be flexible enough to accommodate a wide variety of form factors, orientations, themes, device-specific behaviors or UI metaphors, accessing device capabilities not available in the browser alone, and coding native screens or components.

The goal of this section is to illustrate one possible approach of many for how costs can be minimized over time. Your projects may not have the luxury of starting in this order, but you will still need to define a plan for how your projects (and in other projects across your company) can reduce costs following similar approaches that fit your business case.

Figure 4 – Cost Dimensions

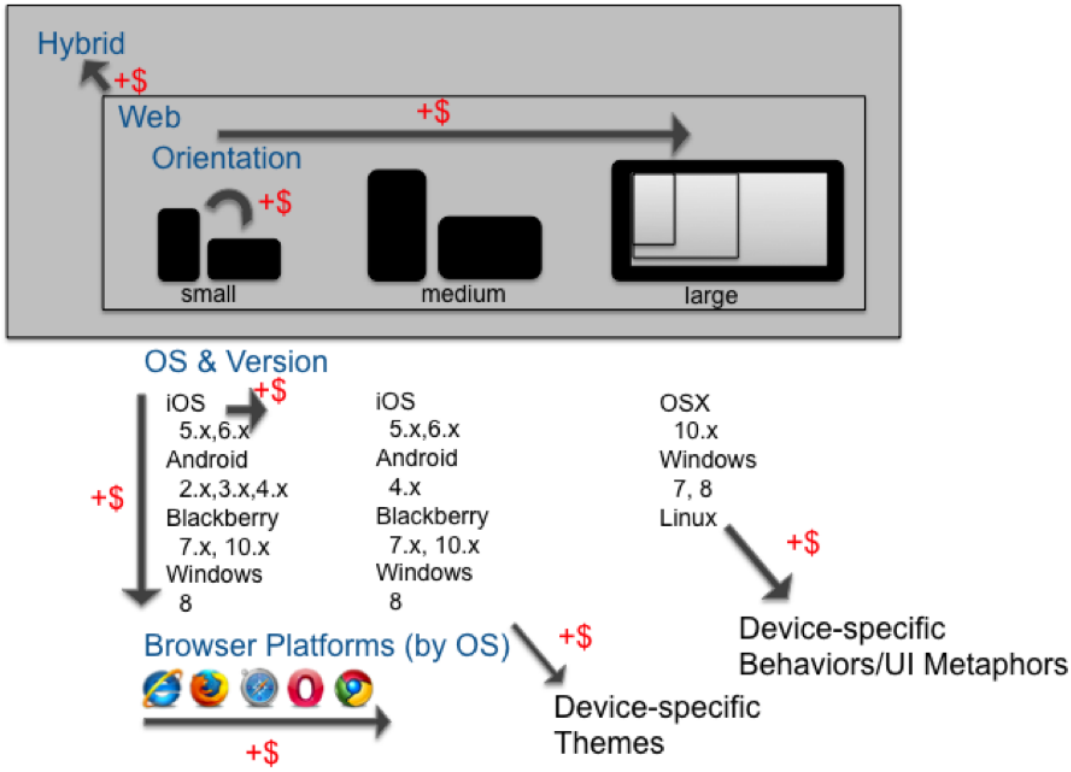
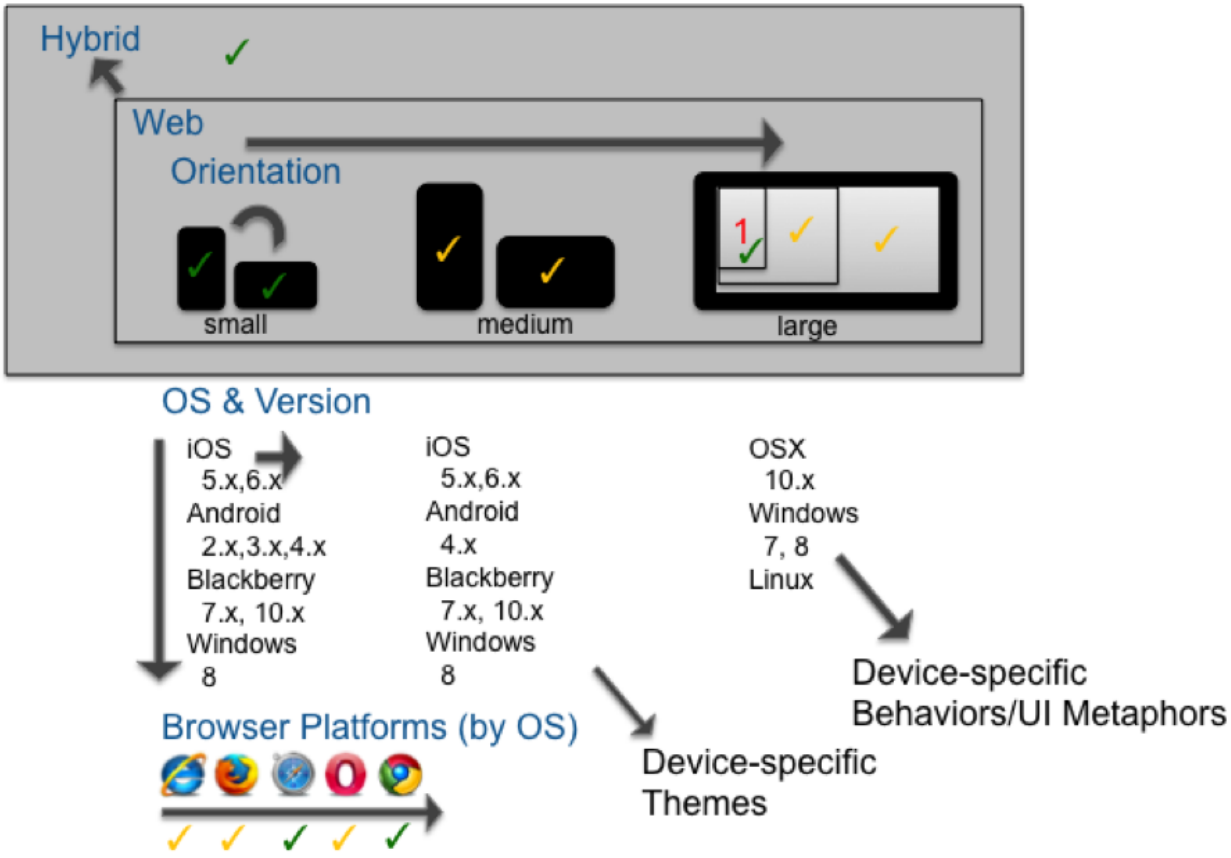


Figure 5 illustrates a lowest cost/broadest reach strategy. This approach requires starting with designing for small form factor devices and desktop hosted web browser at smaller (phone-approximate) sizes, webkit-based devices first but with flexible CSS layouts. This allows the app design on the smartphone to be fully functional and flexible sizing to be tested on all form factors (by resizing the desktop browser manually), until more form-factor appropriate designs can be implemented in subsequent iterations.

Figure 5 – Lowest Cost/Broadest Reach Approach

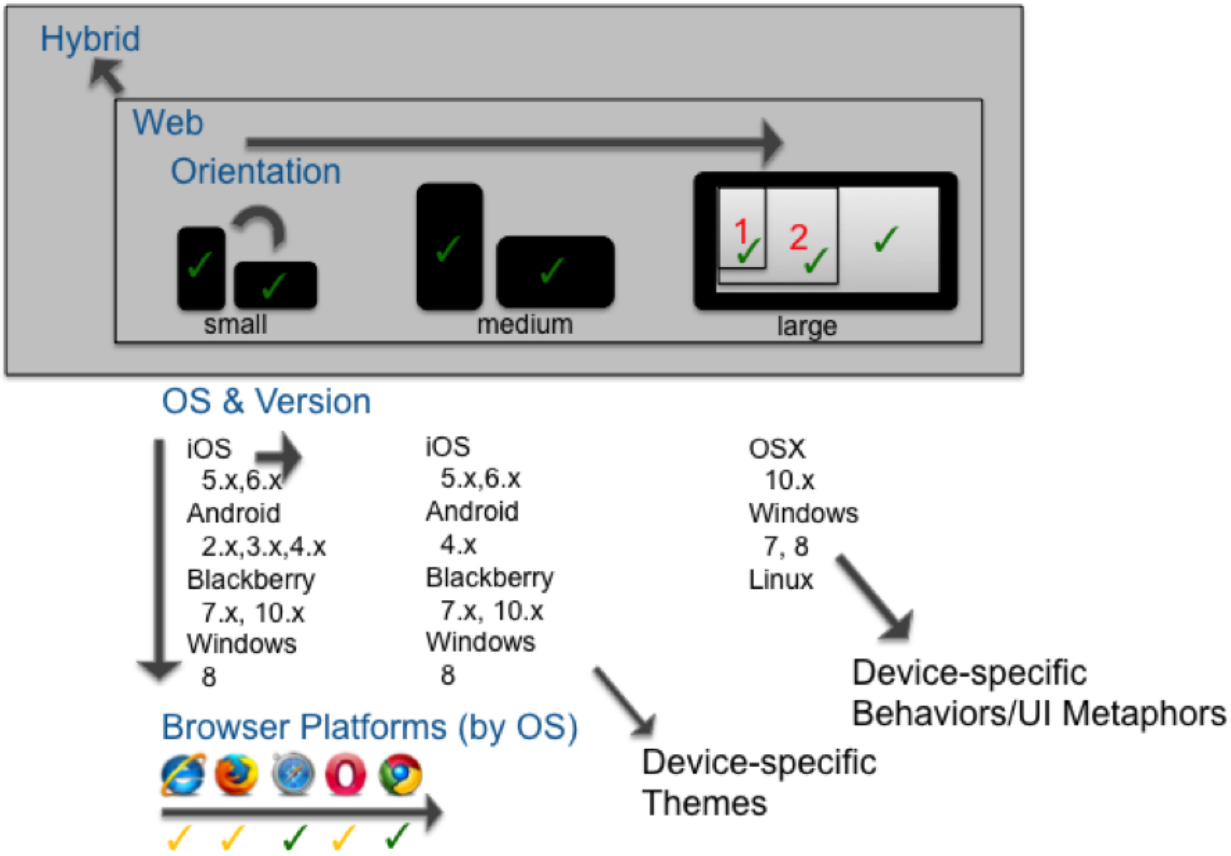


Subsequent iterations can then address design differences at larger form factors. This is where the term, “Mobile First Design” originates. As design moves toward larger form factors, aggregation is used reusing components designed for smaller form factors, reducing the costs that would be associated with decomposing larger user interfaces designs if working in the opposite direction (from larger to smaller form factors).

To further reduce costs, this approach can use a single cross-device unified “theme”, rather than device-specific themes. After web form factors are targeted, application features requiring device capabilities are added using hybrid bridge technologies such as Apache Cordova

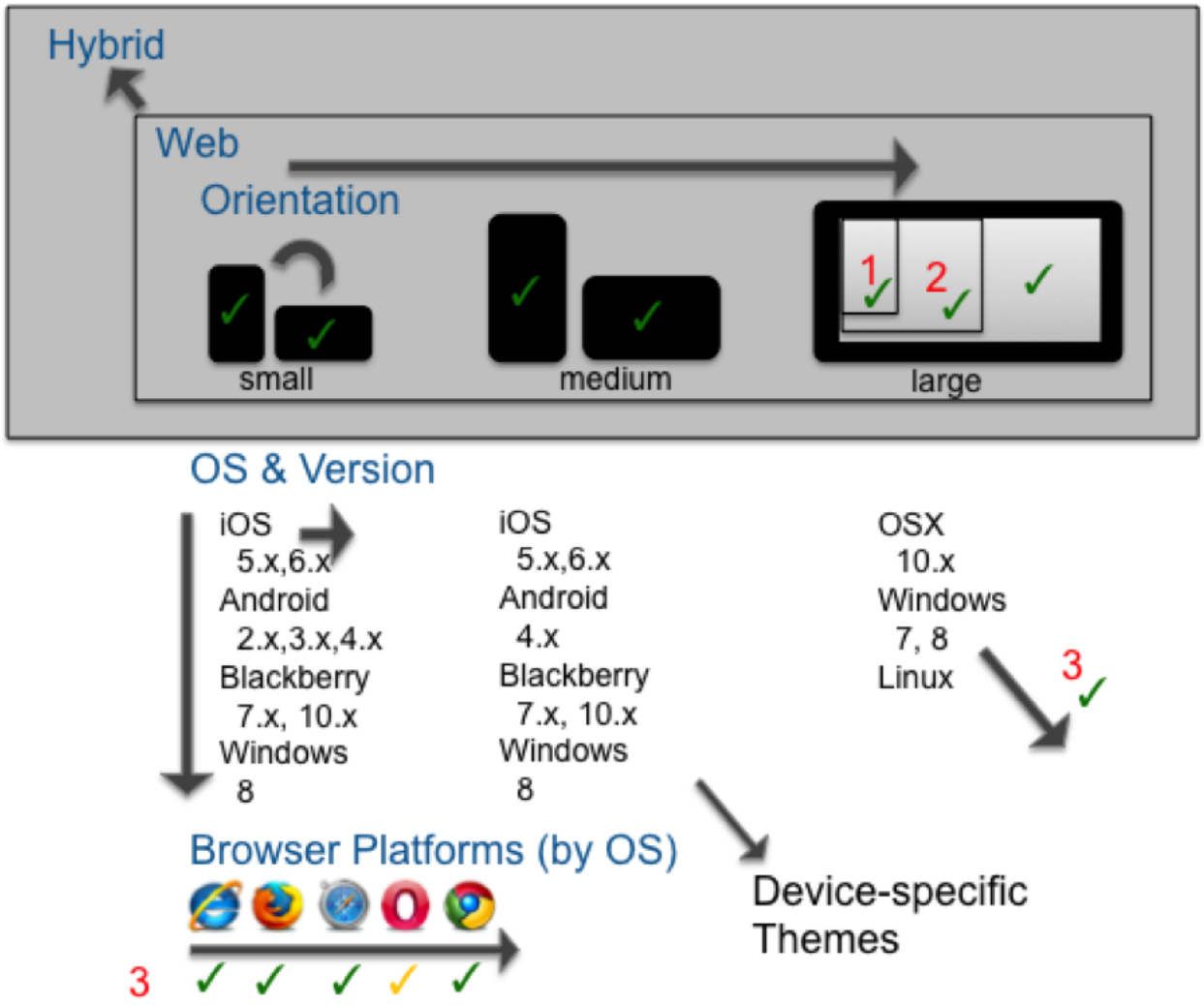
(Phonegap).

Figure 6 – Addressing multiple orientations in different form factors



Finally, Figure 7 illustrates how if you plan to add additional operating system-specific UI behaviors, this will add an additional cost dimension to your project.

Figure 7 – Device specific behavior and tweaks



Beyond Responsive Design

- Responsive Design can't solve all the multi-channel challenges.
    - Dramatic different design for Web vs. Mobile channel.
    - Need for different landing page URL
    - Vendor device model-specific UI needs
- Alternatives to Responsive Design
    - facade for component with different implementations on different devices
    - Server side dynamic CSS using SASS

Environment Awareness

Types of problems that require environmental awareness

- How to design user interfaces that can dynamically adapt to varying capabilities available on devices?
  e.g.
    - Input methods (Keyboard, Touch, Mouse, Headtrack, etc.)
    - Capabilities (SVG vs. Canvas, Camera/Video, Secure storage)

- How to allow different UI controls on different device types?
- How allow different UI controls to be used in different form factors, with very different behaviors?
  eg. Popover dialogs for tablet, slide-in overlays for phone

- How to you keep JavaScript code loaded limited to only what's needed by a particular environment?

Environment-aware Conditional Loading

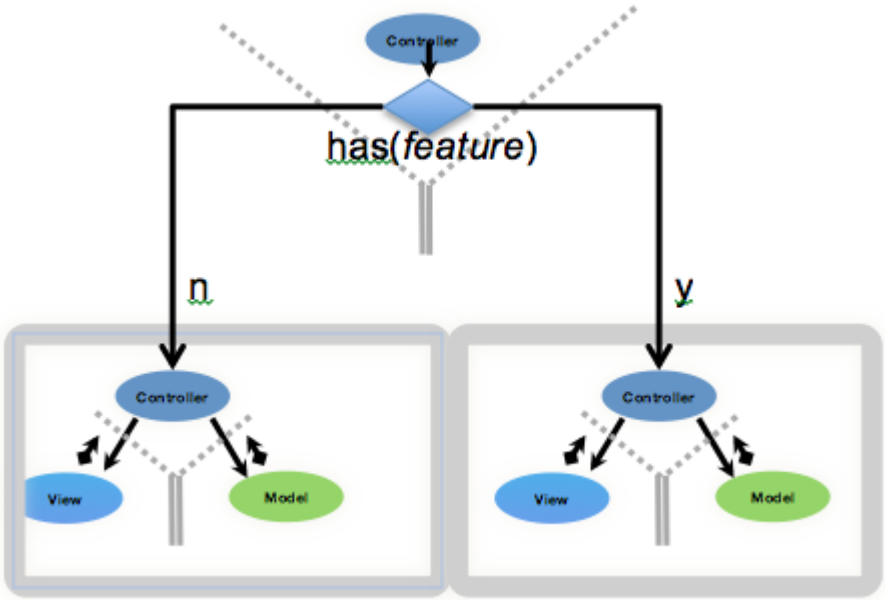
When using MVC patterns, the controller can perform feature detection, then dynamically load an appropriate child controller module (or switch between views already loaded).

This allows the controller to:

- Swap in different controls for different device types (Android/iOS/etc.)
- Swap in different controls for different form factors

A similar technique can be used to dynamically load feature modules for certain types of interaction, only when needed (eg. mouse or touch support)

Device-Awareness and Feature Detection



- Process/Render different content based on device type and feature detection
- Inspecting the User-agent header property of HTTP request
  Mozilla/5.0 (iPhone; CPU iPhone OS 5_0_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Mobile/9A405
  - Device
  - OS
  - Browser

Existing frameworks and API's simplify the task of feature detection:

- has.js
- Dojo – dojo.has, dojox.mobile.deviceTheme
- Modernizr
- IBM Worklight - WL.Client.getEnvironment

IBM Worklight Features for Environment Optimization

There are times when a combination of feature detection and responsive design techniques are still not sufficient, such as overcoming deficiencies in browsers on specific device models or OS versions. In these cases, IBM Worklight's Environment Optimization & Skin capabilities can be used.

Environment Optimization

- The Worklight application structure is divided into environment folders (iPhone, Android, BlackBerry, and so on).
- Each environment folder contains the resources (CSS, JS, images, and so on) that are relevant for that specific environment.

- Customization can be made for a specific environment

Skin

- Skins provide support for multiple form factors in a single executable file for devices of the same OS family.
- Skins are a sub-variant of an environment.
- Skins are packaged together in one app.
- The decision on which skin to use is made automatically at runtime