Delving into the

Mobile Web Framework

Eric Bollens
ebollens AT oit.ucla.edu
Mobile Web Framework Architect
UCLA Office of Information Technology

December 14, 2011

Overview

1. Design Principles
2. Architectural Patterns
3. Building for Degradation

Design Principles

for a mobile experience

Keep It Simple

- Markup should be simple and compatible
  - XHTML MP 1.0
  - WCSS
- Mobile users want information fast
  - Minimize scrolling
  - Avoid excess decoration
  - Short text and icons
- Do not clutter the screen

Task-Oriented Content

- Get the user quickly to where they want
  - Minimize the number of pages to complete a task
  - Keep the user focused on the current task
- Don’t create a mini version of a desktop site
- Reconsider movement around the site

Consider the Context

- Small screen
  - Tightened focus over less real estate
- Touch interface
  - Multi-touch and gestures
  - Different sort of interactive experience
- Mobility
  - Locational awareness
  - Different goals

Demo

from desktop to mobile

Architectural Patterns

that minimize mobile development costs

Architectural Goals

- Encapsulation
- Layers
- Reusability
- Business logic integrity
- Interface consistency

Approaches

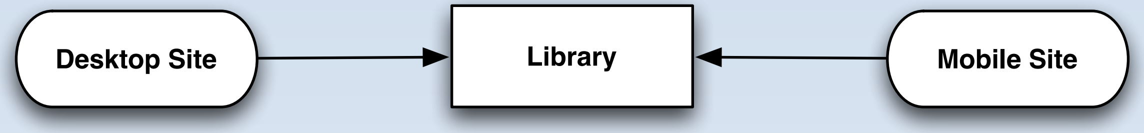
- Shared Libraries
- Model-view-controller (MVC)
- Service-oriented architecture (SOA)

Shared Library

- Concept
  - Libraries of functions and/or objects
  - Separate desktop, tablet and mobile apps
- Properties
  - Reuses objects to accomplish the same task
  - Consistency if the library is used and maintained
  - Requires homogeneous environment

Shared Library

- Good use cases
  - Decorators
  - Session and state management
  - Data setters and getters



Shared Decorator Library

- Object that encapsulates some element
  - Methods permute the content of the element
  - Render method generates the actual output
- Use case:
  - Instantiate the decorator
  - Modify element attributes
  - Add contained entities
  - Render to produce actual HTML output

Shared Decorator Library

- MWF provides two decorator sets
  - HTML Decorators
  - Site Decorators
- Site decorators create MWF entities.
- MWF entities are semantic HTML.
  - Can style them in a desktop manner as well.
  - Can simply define different CSS for desktop.

Shared Decorator Library

- Using the .menu-full decorator directly

```
$decorator = Site_Decorator::menu_full(array(), array('class'=>'main-menu'));

$decorator->set_title('Menu')

$decorator->add_item('Item 1', '#1')

$decorator->add_item('Item 2', '#2')

$decorator ->add_item('Item 3', '#3')

$decorator->add_item('Item 4', '#4')

$decorator->add_item('Item 5', '#5')

$decorator->render();
```

Shared Decorator Library

- Encapsulate the .menu-full decorator

```
class Front_Page_Menu_Full_Site_Decorator extends Menu_Full_Site_Decorator {
    public function __construct($title = false, $params = array())  {
        parent::__construct($title, $params);
        $this->set_param('class', 'main-menu menu-full');
        $this->set_title('Menu');
        $this->add_item('Item 1', '#1');
        $this->add_item('Item 2', '#2');
        $this->add_item('Item 3', '#3');
        $this->add_item('Item 4', '#4');
        $this->add_item('Item 5', '#5');
    }
}
```

Shared Decorator Library

- Using the shared library

```
$menu = new Front_Page_Menu_Full_Site_Decorator();
echo $menu->render();
```

- Mobile and desktop sites both define CSS
  - CSS handler covers the mobile site
  - Different CSS file for the desktop site
- Can consolidate to one invoking script:
  - Redirect script or User_Agent call

Demo

Working with Decorators

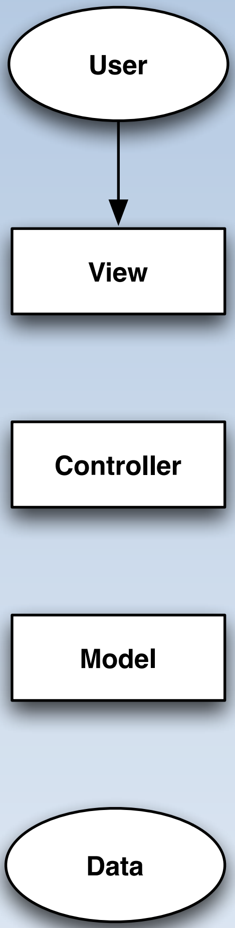
From Decorators to Views

- A decorator may use multiple decorators
  - Many site decorators are tag composites
  - A page decorator could also be a composite
  - UC San Diego has a Java-based page decorator
- Multi-element decorators are views
  - Pass a set of data into an encapsulating object
  - Object renders output based on data

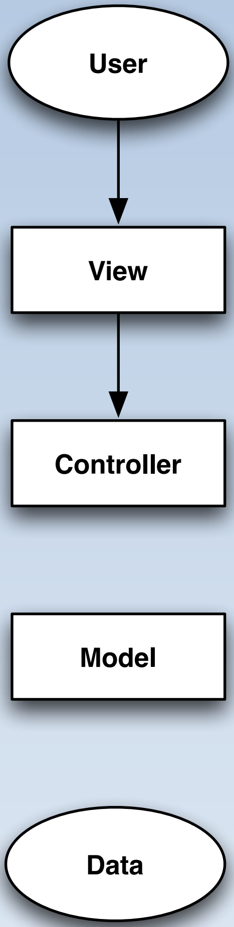
Model-View-Controller

- Model
  - Manages, mediates and manipulates data
- View
  - Encapsulates the user interface
- Controller
  - Bridges model and view with business logic

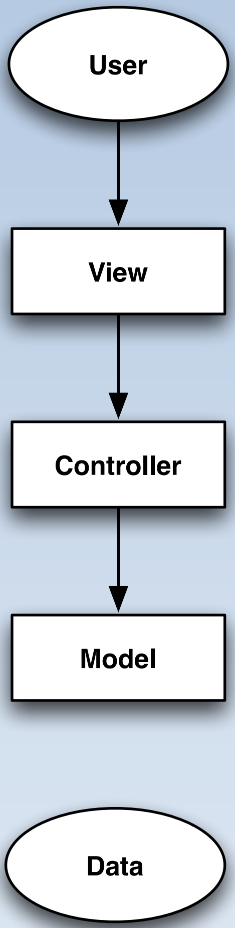
Model-View-Controller Layers



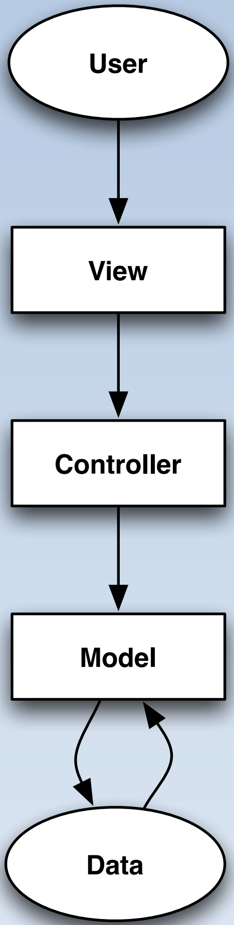
Model-View-Controller Layers



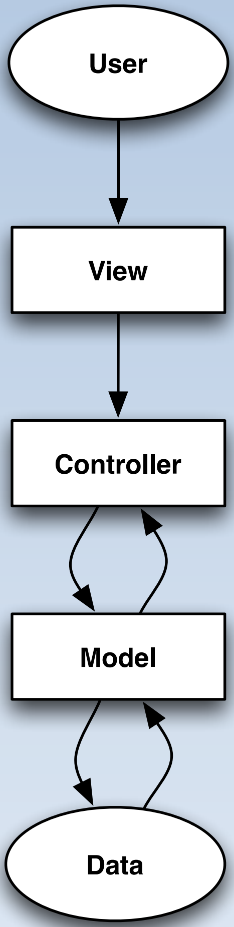
Model-View-Controller Layers



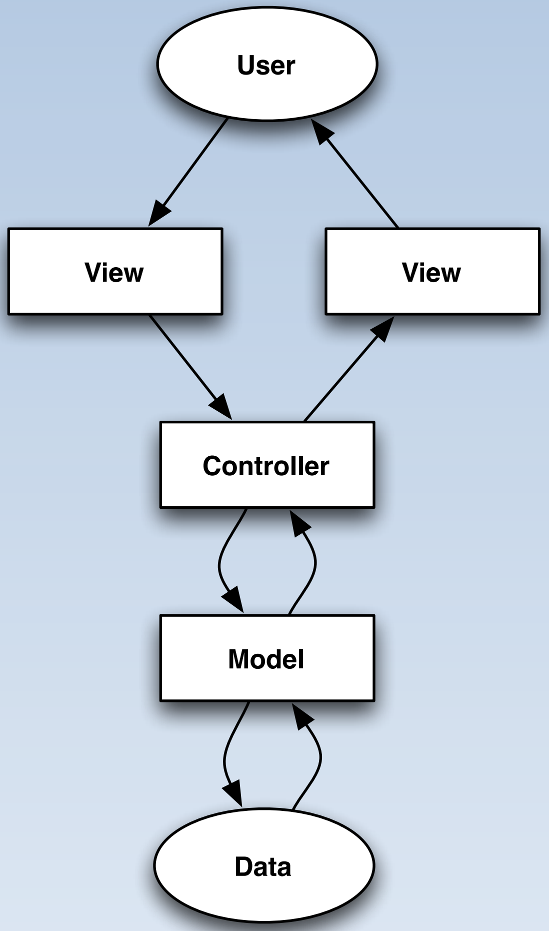
Model-View-Controller Layers



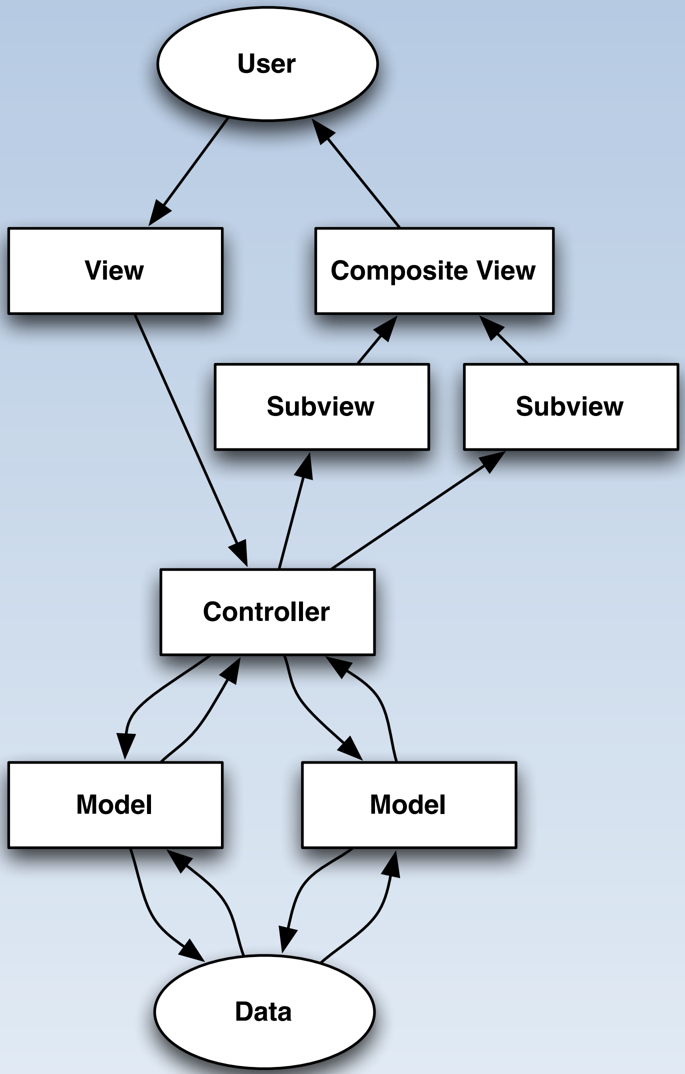
Model-View-Controller Layers



Model-View-Controller Layers



Model-View-Controller Layers



MVC for Mobile Web Apps

- Can build one app that supports:
  - Desktop
  - Tablet
  - Mobile
- MVC approach:
  - One set of controllers and models
  - Different composite views for mobile & desktop
  - Reuse subviews in different composite views

Demo

MVC in Practice

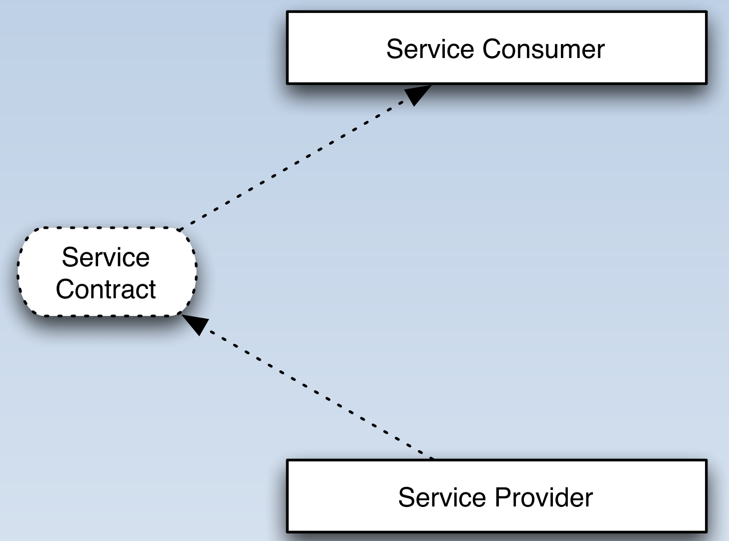
From MVC to SOA

- Views separate rendition from
  - Business logic
  - Data models
- Going a step further:
  - One entity handles business logic & data models
  - Another entity handles rendition
- Basis of service-oriented architecture

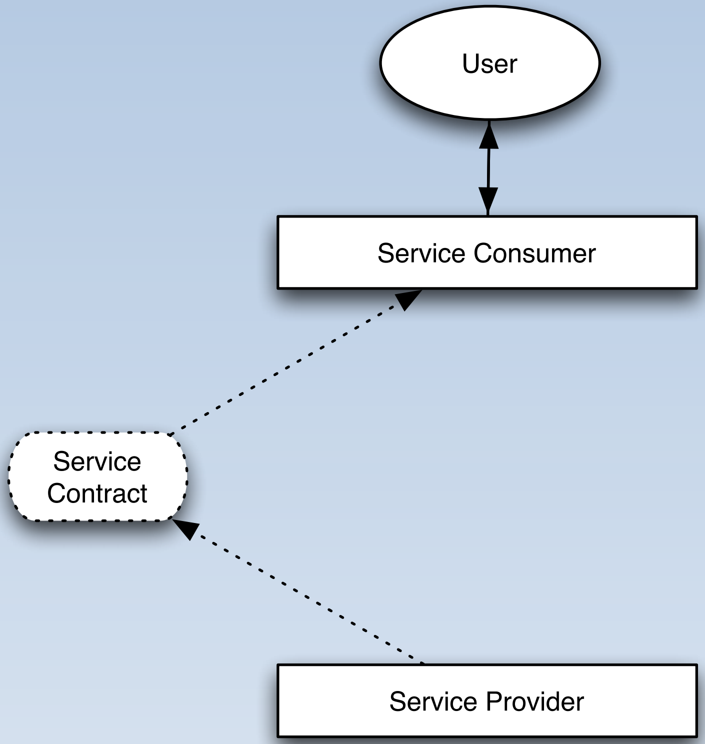
Service-Oriented Architecture

- Service provider
  - Exposes business logic through service interfaces
  - Mediates & manipulates data based on services
- Service consumer
  - Invokes services provided by the service provider
- Service definition
  - Contract between provider & consumer

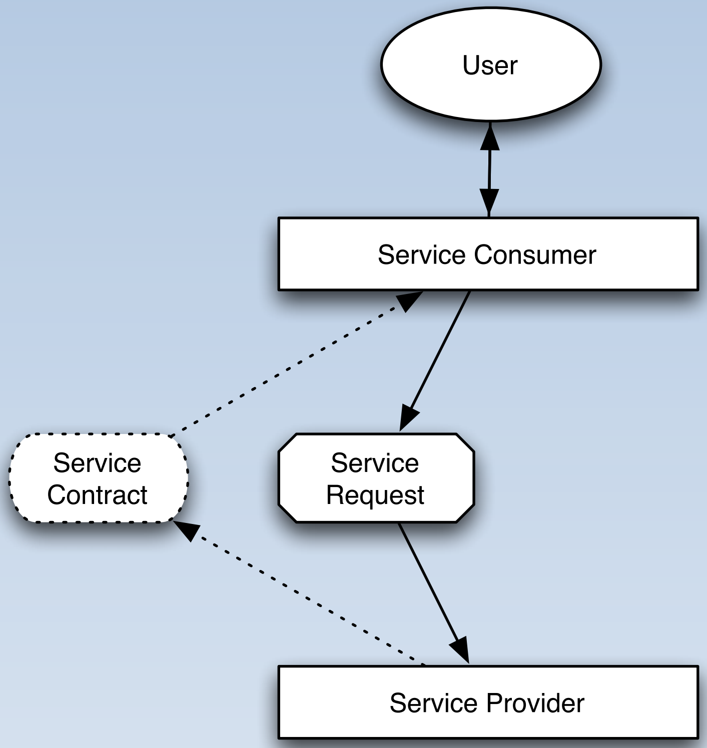
Service-Oriented Architecture



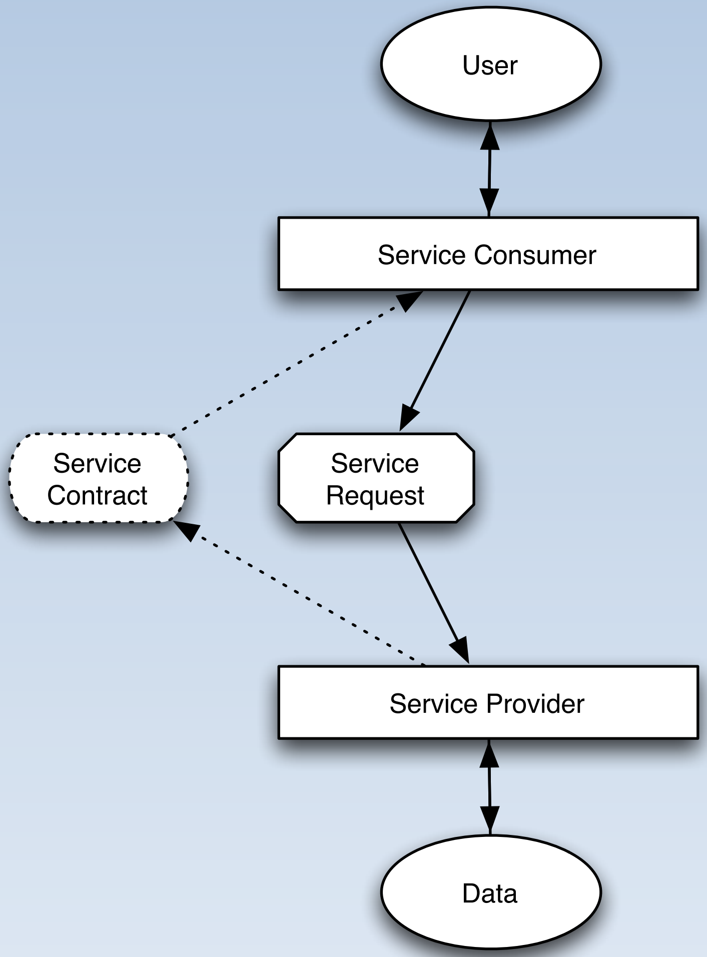
Service-Oriented Architecture



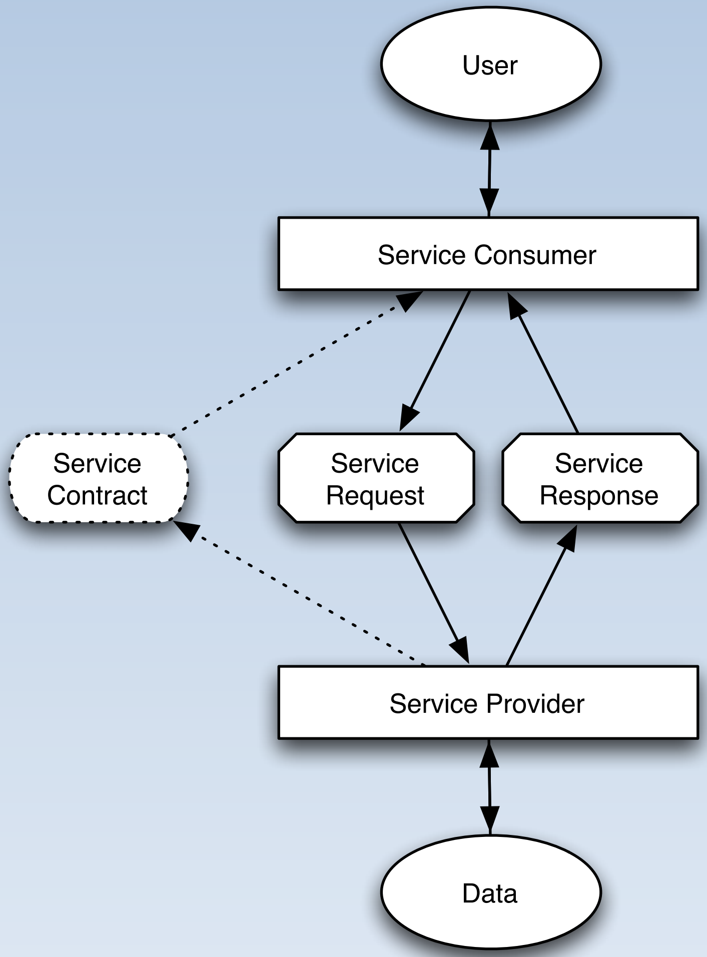
Service-Oriented Architecture



Service-Oriented Architecture



Service-Oriented Architecture



Service-Oriented Architecture

- Service contract
- Loose coupling
- Abstraction
- Reusability
- Autonomy
- Ganularity
- Statelessness

SOA through Web Services

- Often implemented through web services
- Two common modern protocols:
  - SOAP
  - REST

SOA through Web Services

SOAP	REST
Transport neutral	HTTP Transport
Message-driven	HTTP URI/Request-driven
XML	XML, JSON, HTML, etc.
Complex definition	Simple definition
Verbose semantics	Limited semantics
Larger payload	Minimized payload
Must parse for AJAX	Can avoid parsing for AJAX

SOA for Mobile Web Apps

- Define business logic behind web services
- Client calls web services to perform actions
- An analogy back to MVC:
  - Client ~ View
  - Service ~ Controller + Models

SOA for Mobile Web Apps

- Design decisions:
  - REST or SOAP?
  - XML, JSON or HTML?
  - Thick or thin client?
  - Server or browser rendering?

SOA for Mobile Web Apps

- Design decisions:
  - **REST** or SOAP?
  - XML, **JSON** or HTML?
  - Thick or **thin** client?
  - **Server** or browser rendering?

SOA for Mobile Web Apps

- REST
  - Simple and easy to implement
  - Uses HTTP requests and responses
  - Allows XML, JSON, HTML, etc.
- JSON
  - Smaller payload than XML
  - No parsing required for Javascript

SOA for Mobile Web Apps

- Thin Client
  - Reusability
  - Business logic integrity
  - Focus client on presentation
- Server Rendering
  - Not all user agents allow Javascript or AJAX
  - Supplement with AJAX where possible

Building for Degradation

while using MWF and new HTML 5 Technologies

The Situation

- Not all phones
  - have the same features
  - provide access to the same features
  - provide the same access to the same features
- The goal:
  - Use top-end features when available
  - Still remain usable for low end devices
  - Avoid writing two applications

The Situation

- CSS 3
  - Gradients
  - Transitions
- HTML 5
  - Forms and Input Types
  - Semantic Entities
- Javascript
  - DOM Writing
  - AJAX
- Device APIs
  - Audio
  - Video
  - Geolocation
  - Compass
  - Accelerometer
  - Storage
  - Camera
  - Web Sockets

Degradation in MWF

- Handlers load styles/scripts in three tiers:
  - Basic
  - Standard
  - Full
- Degradation further prevalent in:
  - Geolocation
  - Transitions
  - Images

Degrading with CSS 3

- Presentational and cascades
  - Build up from WCSS definitions to CSS 3 definitions
  - If CSS 3 definitions aren't accepted, falls back
- A few simple degradations:
  - Rounded corners can degrade to square
  - Gradient can degrade to median value
  - Transitioning areas can degrade to blocks

Degrading with CSS 3

- Only load where it is allowed:
  - WCSS: Basic
  - CSS 2.1: Standard
  - CSS 3: Full
- This reduces payload size and validation concerns for devices in the classification

Degrading with HTML 5

- HTML 5 introduces new semantics
- Rather than use new entities directly:
  - Use classes on XHTML MP 1.0 elements
  - Transform to HTML 5 elements where supported
- MWF Forms API includes this approach:
  - https://github.com/ucla/mwf/wiki/Roadmap%3A-Framework-v1.2%3A-Forms

Degrading with Javascript

- Live DOM writes not supported universally
  - Degrades by showing what is visible on load
  - Use DOM write to change state of elements
- AJAX not supported universally
  - AJAX should be a plus, not a necessity
  - Define <a href…> to a new page
  - Override default with AJAX request
  - Web service can serve to AJAX and new pages

Degrading with Device APIs

- Audio/Video
  - Pre-HTML 5 semantics don’t include tags
  - Degrade to other viable players
  - Least common denominator is error message
  - MWF will eventually have an Audio/Video API
- Geolocation
  - MWF has abstraction layer for HTML 5 & Gears
  - GPS failure is treated similarly to no GPS

Degrading with Device APIs

- Compass & Accelerometer
  - Degradation similar to Geolocation
  - API will be added to MWF
- Storage
  - Specifications shifting rapidly so use caution
  - Abstraction layer to handle shifting support
  - State-saving can be offloaded via AJAX
  - API will be added to MWF

Thank You

for listening