# Creating Content with iAd JS

## Part 2—The iAd JS Framework

**Antoine Quint**
iAd JS Software Engineer
iOS Apps and Frameworks

## Agenda

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Prerequisites

### Understanding of web technologies

- HTML
- CSS
- JavaScript
- DOM APIs

### Familiarity with UIKit

## Introducing iAd JS

Motivations and features

iAd

## iAd Goals

- Emotion
- Interactivity

## Technology

- Media playback
- Expressive graphics
- Motion
- Touch

## Rich Media

## Rich Media Technologies

### Key features in WebKit on iPhone

HTML5

## Rich Media Technologies

### Key features in WebKit on iPhone

`<audio>`

HTML5

## Rich Media Technologies

### Key features in WebKit on iPhone

`<video>`

HTML5

## Rich Media Technologies

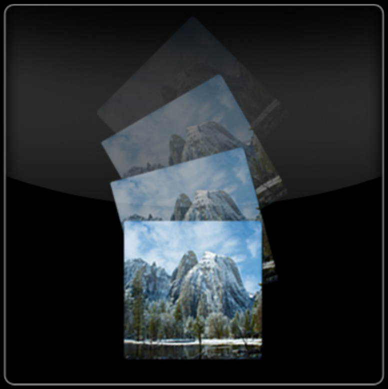
Key features in WebKit on iPhone

CSS3

HTML5

## Rich Media Technologies

### Key features in WebKit on iPhone



CSS Transforms 2D and 3D

HTML5 CSS3

# Rich Media Technologies

## Key features in WebKit on iPhone

CSS Transitions

HTML5

CSS3

## Rich Media Technologies

### Key features in WebKit on iPhone

CSS Animations

HTML5 CSS3

# Rich Media Technologies

## Key features in WebKit on iPhone

HTML5

CSS3

## Rich Media Technologies

### Key features in WebKit on iPhone

Touch Events

HTML5

CSS3

## Rich Media Technologies

### Key features in WebKit on iPhone

HTML5

CSS3

## Rich Media Technologies

### Key features in WebKit on iPhone

HTML5 CSS3

HTML5

## Rich Media

HTML5

## Rich Media

## Mini Apps

## Rich Media

HTML5

## Mini Apps

iAd

## iAd JS Goals

- Streamline creation of rich ads
- Ensure high performance

## iAd JS Features

- UIKit design and rich features
- Declarative programming model
- Modular and incremental building

## How?

## 100% Web Standards

No native code was hurt in the making of iAd JS

## Agenda

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Core JavaScript Enhancements

ADClass and ADObject

## JavaScript Limitations

### Inheritance

- No explicit support for traditional inheritance in JavaScript
- No concept of a super class and no `super` keyword

### Property Synthesis

- Synthesized properties are prevalent in UIKit design
- No built-in support in JavaScript syntax

### Property Change Notifications

## Core JavaScript Enhancements

| ADClass Utility | ADObject Base Class |
|---|---|
| **Inheritance**<br>The `superclass` property | **Subclassing**<br>The `callSuper()` method |
| **Synthesis**<br>The `synthetizedProperties` array | **Notifications**<br>Observe property changes |

Automated With Synthesis

## ADClass

### Inheritance

```
// specify the superclass
MyClass.superclass = ADObject;
```

```
function MyClass () {
  // ensure parent constructor is called
  this.callSuper();
}

// an instance method declaration
MyClass.prototype.someMethod = function () {
  // method code
};

// let ADClass process this class
ADClass(MyClass);
```

## ADClass

### Inheritance

```
// specify the superclass
MyClass.superclass = ADObject;

function MyClass () {
  // ensure parent constructor is called
  this.callSuper();
}

// an instance method declaration
MyClass.prototype.someMethod = function () {
  // method code
};

// let ADClass process this class
ADClass(MyClass);
```

## ADClass

### Inheritance

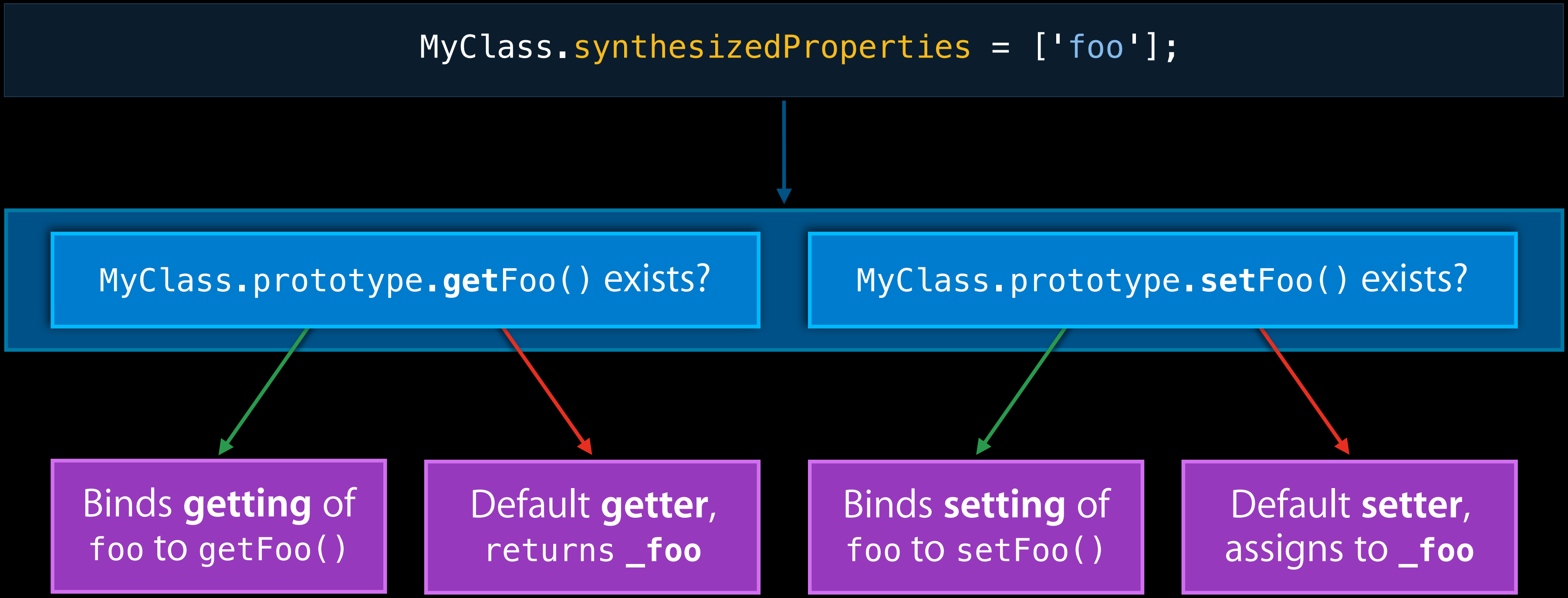
```
// specify the superclass
MyClass.superclass = ADObject;

function MyClass () {
  // ensure parent constructor is called
  this.callSuper();
}

// an instance method declaration
MyClass.prototype.someMethod = function () {
  // method code
};

// let ADClass process this class
ADClass(MyClass);
```

## ADClass

### Inheritance

```
// specify the superclass
MyClass.superclass = ADObject;

function MyClass () {
  // ensure parent constructor is called
  this.callSuper();
}

// an instance method declaration
MyClass.prototype.someMethod = function () {
  // method code
};

// let ADClass process this class
ADClass(MyClass);
```

## Property Synthesis Process



```
MyClass.synthesizedProperties = ['foo'];
```

MyClass.prototype.getFoo() exists?
MyClass.prototype.setFoo() exists?
Binds getting of foo to getFoo()
Default getter, returns _foo
Binds setting of foo to setFoo()
Default setter, assigns to _foo

## Property Synthesis

### Custom setter

```
// specify properties to synthesize
MyClass.synthesizedProperties = ['foo'];

function MyClass () {
  // declare private ivar with convention “_” prefix
  this._foo = 0;
}

// method automatically called when .foo is set
MyClass.prototype.setFoo = function (foo) {
  // custom setter code
  this._foo = foo;
};

ADClass(MyClass);
```

## Property Synthesis

### Custom setter

```
// specify properties to synthesize
MyClass.synthesizedProperties = ['foo'];

function MyClass () {
  // declare private ivar with convention “_” prefix
  this._foo = 0;
}

// method automatically called when .foo is set
MyClass.prototype.setFoo = function (foo) {
  // custom setter code
  this._foo = foo;
};

ADClass(MyClass);
```

# Property Synthesis

## Custom setter

```
// specify properties to synthesize
MyClass.synthesizedProperties = ['foo'];

function MyClass () {
  // declare private ivar with convention “_” prefix
  this._foo = 0;
}

// method automatically called when .foo is set
MyClass.prototype.setFoo = function (foo) {
  // custom setter code
  this._foo = foo;
};

ADClass(MyClass);
```

## Property Synthesis

### Custom setter

```
// specify properties to synthesize
MyClass.synthesizedProperties = ['foo'];

function MyClass () {
  // declare private ivar with convention “_” prefix
  this._foo = 0;
}

// method automatically called when .foo is set
MyClass.prototype.setFoo = function (foo) {
  // custom setter code
  this._foo = foo;
};

ADClass(MyClass);
```

## Property Synthesis

### Custom getter

```
// specify properties to synthesize
MyClass.synthesizedProperties = ['foo'];

function MyClass () {
  // constructor code
}

// method automatically called when .foo is gotten
MyClass.prototype.getFoo = function () {
  var foo;
  // custom code to compute foo
  return foo;
};

ADClass(MyClass);
```

## Property Observing

### Default protocol

```
var controller = {
  myObject : new MyClass()
};

controller.init = function () {
  // make our controller observe myObject.foo
  this.myObject.addPropertyObserver('foo', this);
};

// controller must implement the ADPropertyChange protocol which defines the
// handlePropertyChange method as callback upon property change
controller.handlePropertyChange = function (observedObject, propertyName) {
  // handle foo property change
};
```

## Property Observing

### Default protocol

```
var controller = {
  myObject : new MyClass()
};

controller.init = function () {
  // make our controller observe myObject.foo
  this.myObject.addPropertyObserver('foo', this);
};

// controller must implement the ADPropertyChange protocol which defines the
// handlePropertyChange method as callback upon property change
controller.handlePropertyChange = function (observedObject, propertyName) {
  // handle foo property change
};
```

## Property Observing

### Default protocol

```
var controller = {
  myObject : new MyClass()
};

controller.init = function () {
  // make our controller observe myObject.foo
  this.myObject.addPropertyObserver('foo', this);
};

// controller must implement the ADPropertyChange protocol which defines the
// handlePropertyChange method as callback upon property change
controller.handlePropertyChange = function (observedObject, propertyName) {
  // handle foo property change
};
```

## Property Observing

### Custom callback

```
var controller = {
  myObject : new MyClass()
};

controller.init = function () {
  // make our controller observe myObject.foo
  this.myObject.addPropertyObserver('foo', this, 'fooDidChange');
};

// controller must implement the fooDidChange method which we defined as the
// custom callback in addPropertyObserver
controller.fooDidChange = function () {
  // handle .foo property change
};
```

## Property Observing

### Custom callback

```
var controller = {
  myObject : new MyClass()
};

controller.init = function () {
  // make our controller observe myObject.foo
  this.myObject.addPropertyObserver('foo', this, 'fooDidChange');
};

// controller must implement the fooDidChange method which we defined as the
// custom callback in addPropertyObserver
controller.fooDidChange = function () {
  // handle .foo property change
};
```

## Summary

### Core JavaScript enhancements

- Traditional **inheritance** with `superclass` and `callSuper()`
- Built-in **synthesis** with `synthesizedProperties`
- Automated **property change notifications** through synthesis

## Agenda

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Working with Views and Controls

ADView and ADControl

## Views

### ADView

- Based on UIKit’s `UIView`
- `ADView` is the base class for anything that renders on screen
- Wraps a DOM element and its subtree via the `layer` property
- Manual DOM manipulation possible with the `hostingLayer` property
- Transaction-based transition system
- HTML `body` accessible as `ADRootView.sharedRoot`

## View Instantiation

### Using JavaScript APIs

```
// create our ADScrollView instance
var scrollView = new ADScrollView();

// set up the viewable size
scrollView.position = new ADPoint(20, 20);
scrollView.size = new ADSize(window.innerWidth − 40, window.innerHeight − 40);

// let's only scroll vertically and without indicators
scrollView.verticalScrollEnabled = false;
scrollView.showsHorizontalScrollIndicator = false;

// add to the root view to display the scroll view
ADRootView.sharedRoot.addSubview(scrollView);
```

## View Instantiation

### Using JavaScript APIs

```
// create our ADScrollView instance
var scrollView = new ADScrollView();

// set up the viewable size
scrollView.position = new ADPoint(20, 20);
scrollView.size = new ADSize(window.innerWidth - 40, window.innerHeight - 40);

// let's only scroll vertically and without indicators
scrollView.verticalScrollEnabled = false;
scrollView.showsHorizontalScrollIndicator = false;

// add to the root view to display the scroll view
ADRootView.sharedRoot.addSubview(scrollView);
```

## View Instantiation

### Using JavaScript APIs

```
// create our ADScrollView instance
var scrollView = new ADScrollView();

// set up the viewable size
scrollView.position = new ADPoint(20, 20);
scrollView.size = new ADSize(window.innerWidth - 40, window.innerHeight - 40);

// let's only scroll vertically and without indicators
scrollView.verticalScrollEnabled = false;
scrollView.showsHorizontalScrollIndicator = false;

// add to the root view to display the scroll view
ADRootView.sharedRoot.addSubview(scrollView);
```

## View Instantiation

### Using JavaScript APIs

```
// create our ADScrollView instance
var scrollView = new ADScrollView();

// set up the viewable size
scrollView.position = new ADPoint(20, 20);
scrollView.size = new ADSize(window.innerWidth - 40, window.innerHeight - 40);

// let's only scroll vertically and without indicators
scrollView.verticalScrollEnabled = false;
scrollView.showsHorizontalScrollIndicator = false;

// add to the root view to display the scroll view
ADRootView.sharedRoot.addSubview(scrollView);
```

## View Instantiation

### Generated markup

```
<!-- this is the view’s layer property -->
<div class="ad-scroll-view ad-view"
     style="-webkit-transform: translate(20px, 20px);
            width: 280px; height: 440px;">
  <!-- this is the view’s hostingLayer property -->
  <div class="hosting-layer">
    <!-- scrollable content -->
  </div>
  <!-- other private views specific to ADScrollView -->
  <div class="ad-scroll-indicator ad-view horizontal indicator-default">…</div>
  <div class="ad-scroll-indicator ad-view vertical indicator-default">…</div>
</div>
```

## Why so much code?

## I’d rather use markup!

## The Case for a Declarative Approach

### Ease of authoring

- Easier to style content with a known markup structure
- Separation of logic and content is just good sense

### Performance

- Fewer manipulations of DOM tree
- Less time spent rendering

## View Instantiation

### Using declarative layer

```
<!-- the root view, usually body -->
<body class="ad-root-view">

  <!-- the scrollview -->
  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       style="left: 20px; top: 20px; right: 20px; bottom: 20px;">
    <!-- the hosting layer -->
    <div class="hosting-layer">
      <!-- scrollable content -->
    </div>
  </div>

</body>
```

## View Instantiation

### Using declarative layer

```
<!-- the root view, usually body -->
<body class="ad-root-view">

  <!-- the scrollview -->
  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       style="left: 20px; top: 20px; right: 20px; bottom: 20px;">
    <!-- the hosting layer -->
    <div class="hosting-layer">
      <!-- scrollable content -->
    </div>
  </div>

</body>
```

## View Instantiation

### Using declarative layer

```
<!-- the root view, usually body -->
<body class="ad-root-view">

  <!-- the scrollview -->
  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       style="left: 20px; top: 20px; right: 20px; bottom: 20px;">
    <!-- the hosting layer -->
    <div class="hosting-layer">
      <!-- scrollable content -->
    </div>
  </div>

</body>
```

## View Instantiation

### Using declarative layer

```
<!-- the root view, usually body -->
<body class="ad-root-view">

  <!-- the scrollview -->
  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       style="left: 20px; top: 20px; right: 20px; bottom: 20px;">
    <!-- the hosting layer -->
    <div class="hosting-layer">
      <!-- scrollable content -->
    </div>
  </div>

</body>
```

## View Instantiation

### Using declarative layer

```
<!-- the root view, usually body -->
<body class="ad-root-view">

  <!-- the scrollview -->
  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       style="left: 20px; top: 20px; right: 20px; bottom: 20px;">
    <!-- the hosting layer -->
    <div class="hosting-layer">
      <!-- scrollable content -->
    </div>
  </div>

</body>
```

## Views

## Controls

## Controls

### ADControl

- Subclass of `ADView`
- Provide advanced and automatic touch tracking
- Extends built-in touch events with more granularity

## Extended Touch Events

### Mouse events

- Mouse events are relative to a target element
  - `mouseover`
  - `mouseout`
  - `mousemove`
  - `mousedown`
  - `mouseup`
  - `click`

## Extended Touch Events

### Base touch events

- Built-in touch events provide raw touches
  - `touchstart`
  - `touchmove`
  - `touchend`

### Tracking touches relative to a given element is hard

## Extended Touch Events

### iAd JS control events

- Controls **analyze** raw touches and trigger additional touch events
  - `controlTouchDragEnter`
  - `controlTouchDragExit`
  - `controlTouchUpInside`
  - and more…

## Related Session

| | |
|---|---|
| Adding Touch and Gesture Detection to Web Pages on iPhone OS | Nob Hill<br>Wednesday 2:00PM |

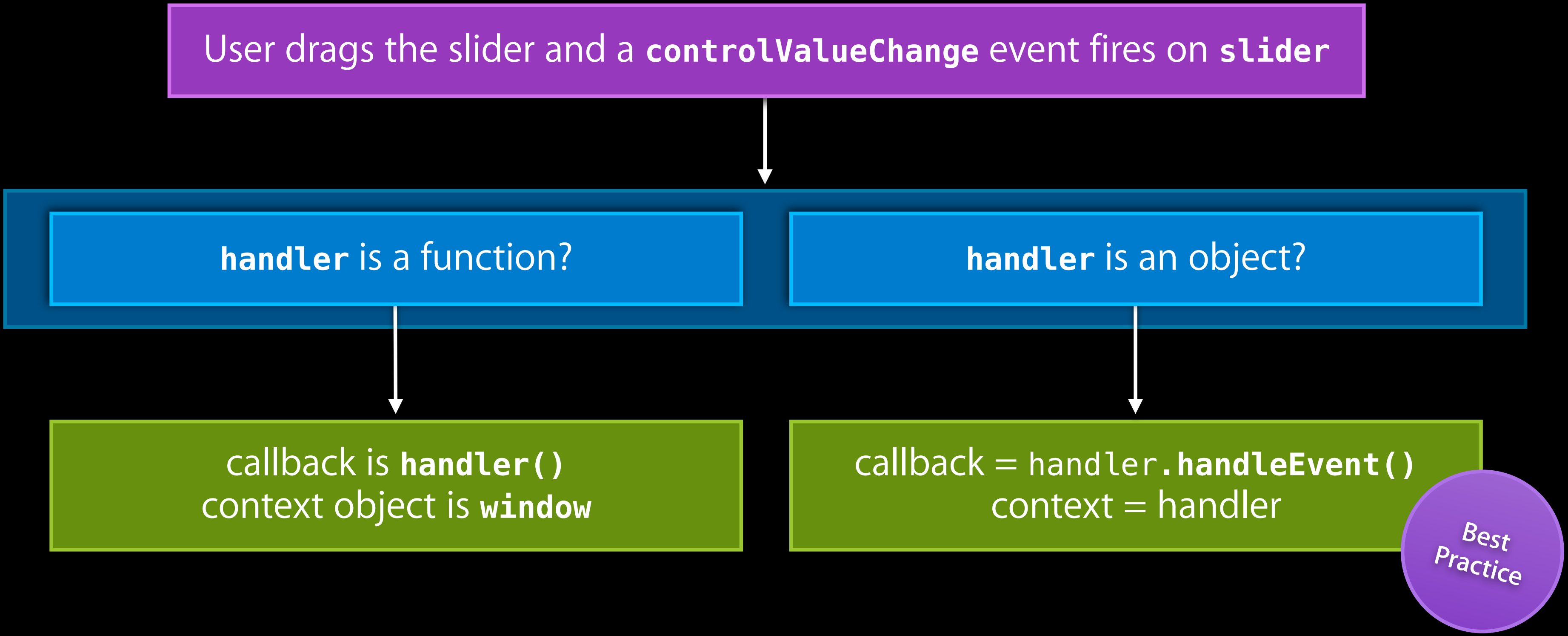
## Actions

### Using DOM events

- UIKit’s action-target mechanism is one-to-one
- Model in web development is one-to-many with DOM events
- A `controlValueChange` is dispatched when a control value changes

# DOM Event Handling

```
slider.addEventListener('controlValueChange', handler, false);
```



User drags the slider and a **`controlValueChange`** event fires on **`slider`**

**`handler`** is a function?

**`handler`** is an object?

callback is **`handler()`**
context object is **`window`**

callback = `handler`**`.handleEvent()`**
context = handler

Best Practice

## Control States

- Selected
- Enabled
- Highlighted
  - Synchronized with touch tracking
  - State is reflected in CSS

## Building a Custom Control

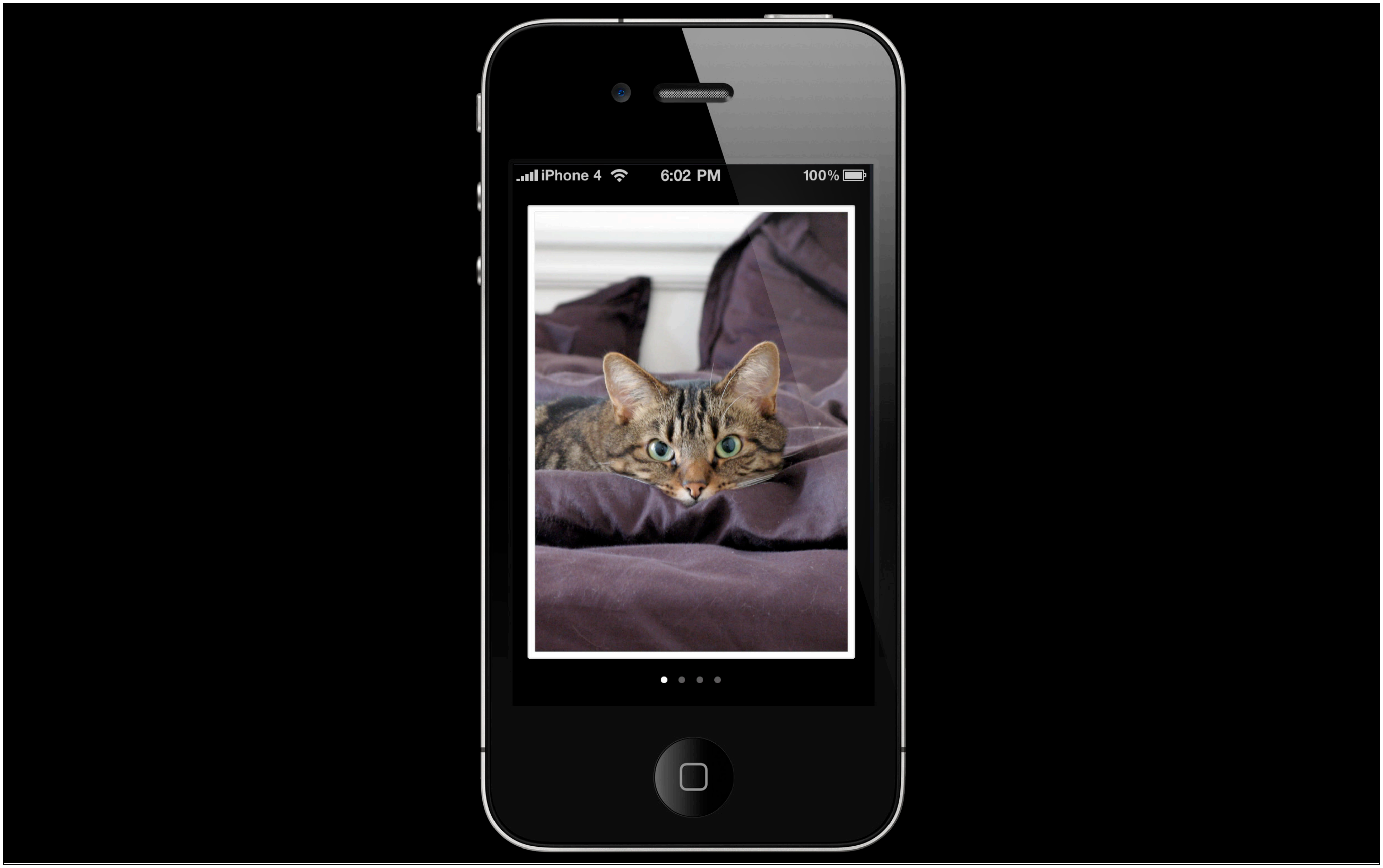
### Markup

```html
<div class="ad-control my-custom-control"></div>
```

### Style

```css
/* default state for control */
.ad-control.my-custom-control {
  background-color: white;
  color: red;
}

/* highlighted state for control */
.ad-page-control.my-custom-control.highlighted {
  background-color: red;
  color: white;
}
```

## Code Sample


iPhone 4
6:02 PM
100%

## Markup

```
<body class="ad-root-view">

  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       ad-paging-enabled="true">
    <div class="hosting-layer">
      <img src="stig1.png">
      <img src="stig2.png">
      <img src="stig3.png">
      <img src="stig4.png">
    </div>
  </div>

  <div class="ad-page-control"
       ad-number-of-pages="4"
       ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>

</body>
```

## Markup

```
<body class="ad-root-view">

  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       ad-paging-enabled="true">
    <div class="hosting-layer">
      <img src="stig1.png">
      <img src="stig2.png">
      <img src="stig3.png">
      <img src="stig4.png">
    </div>
  </div>

  <div class="ad-page-control"
       ad-number-of-pages="4"
       ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>

</body>
```

## Markup

```
<body class="ad-root-view">

  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       ad-paging-enabled="true">
    <div class="hosting-layer">
      <img src="stig1.png">
      <img src="stig2.png">
      <img src="stig3.png">
      <img src="stig4.png">
    </div>
  </div>

  <div class="ad-page-control"
       ad-number-of-pages="4"
       ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>

</body>
```

## Markup

```
<body class="ad-root-view">

  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       ad-paging-enabled="true">
    <div class="hosting-layer">
      <img src="stig1.png">
      <img src="stig2.png">
      <img src="stig3.png">
      <img src="stig4.png">
    </div>
  </div>

  <div class="ad-page-control"
       ad-number-of-pages="4"
       ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>

</body>
```

## Markup

```
<body class="ad-root-view">

  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       ad-paging-enabled="true">
    <div class="hosting-layer">
      <img src="stig1.png">
      <img src="stig2.png">
      <img src="stig3.png">
      <img src="stig4.png">
    </div>
  </div>

  <div class="ad-page-control"
       ad-number-of-pages="4"
       ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>

</body>
```

## Markup

```
<body class="ad-root-view">

  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       ad-paging-enabled="true">
    <div class="hosting-layer">
      <img src="stig1.png">
      <img src="stig2.png">
      <img src="stig3.png">
      <img src="stig4.png">
    </div>
  </div>

  <div class="ad-page-control"
       ad-number-of-pages="4"
       ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>

</body>
```

## Markup

```
<body class="ad-root-view">

  <div class="ad-scroll-view"
       ad-vertical-scroll-enabled="false"
       ad-shows-horizontal-scroll-indicator="false"
       ad-paging-enabled="true">
    <div class="hosting-layer">
      <img src="stig1.png">
      <img src="stig2.png">
      <img src="stig3.png">
      <img src="stig4.png">
    </div>
  </div>

  <div class="ad-page-control"
       ad-number-of-pages="4"
       ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>

</body>
```

## Layout

```
/* size the scroll view */
.ad-scroll-view {
  width: 320px;
  height: 421px;
}
```

```
/* position and size the page control */
.ad-page-control {
  top: 422px;
  width: 320px;
  height: 38px;
}
```

## Layout

```
/* size the scroll view */
.ad-scroll-view {
  width: 320px;
  height: 421px;
}

/* position and size the page control */
.ad-page-control {
  top: 422px;
  width: 320px;
  height: 38px;
}
```

## Scrollable Content Layout

```
/* explicitly size the hosting layer */
.hosting-layer {
  width: 1280px;
  padding-left: 16px;
}

/* amount of x spacing between images */
.hosting-layer > img {
  margin-right: 28px;
}

/* no margin for last image */
.hosting-layer > img:last-of-type {
  margin-right: 0;
}
```

## Scrollable Content Layout

```
/* explicitly size the hosting layer */
.hosting-layer {
  width: 1280px;
  padding-left: 16px;
}

/* amount of x spacing between images */
.hosting-layer > img {
  margin-right: 28px;
}

/* no margin for last image */
.hosting-layer > img:last-of-type {
  margin-right: 0;
}
```

## Scrollable Content Layout

```
/* explicitly size the hosting layer */
.hosting-layer {
  width: 1280px;
  padding-left: 16px;
}

/* amount of x spacing between images */
.hosting-layer > img {
  margin-right: 28px;
}

/* no margin for last image */
.hosting-layer > img:last-of-type {
  margin-right: 0;
}
```

## Controller

```
var controller = {};

controller.init = function () {
  // get a reference to our page control
  this.pageControl = ADRootView.sharedRoot.pageControls[0];
  // get a reference to our scroll view
  this.scrollView = ADRootView.sharedRoot.scrollViews[0];
  // wire up the scroll view delegate to be our controller
  this.scrollView.delegate = this;
};

function init () {
  // init our controller
  controller.init();
}

// call init() as soon as the page is fully loaded
window.addEventListener('DOMContentLoaded', init, false);
```

## Controller

```
var controller = {};

controller.init = function () {
  // get a reference to our page control
  this.pageControl = ADRootView.sharedRoot.pageControls[0];
  // get a reference to our scroll view
  this.scrollView = ADRootView.sharedRoot.scrollViews[0];
  // wire up the scroll view delegate to be our controller
  this.scrollView.delegate = this;
};

function init () {
  // init our controller
  controller.init();
}

// call init() as soon as the page is fully loaded
window.addEventListener('DOMContentLoaded', init, false);
```

## Controller

```
var controller = {};

controller.init = function () {
  // get a reference to our page control
  this.pageControl = ADRootView.sharedRoot.pageControls[0];
  // get a reference to our scroll view
  this.scrollView = ADRootView.sharedRoot.scrollViews[0];
  // wire up the scroll view delegate to be our controller
  this.scrollView.delegate = this;
};

function init () {
  // init our controller
  controller.init();
}

// call init() as soon as the page is fully loaded
window.addEventListener('DOMContentLoaded', init, false);
```

## Controller

```
var controller = {};

controller.init = function () {
  // get a reference to our page control
  this.pageControl = ADRootView.sharedRoot.pageControls[0];
  // get a reference to our scroll view
  this.scrollView = ADRootView.sharedRoot.scrollViews[0];
  // wire up the scroll view delegate to be our controller
  this.scrollView.delegate = this;
};

function init () {
  // init our controller
  controller.init();
}

// call init() as soon as the page is fully loaded
window.addEventListener('DOMContentLoaded', init, false);
```

## Controller

```
var controller = {};

controller.init = function () {
  // get a reference to our page control
  this.pageControl = ADRootView.sharedRoot.pageControls[0];
  // get a reference to our scroll view
  this.scrollView = ADRootView.sharedRoot.scrollViews[0];
  // wire up the scroll view delegate to be our controller
  this.scrollView.delegate = this;
};

function init () {
  // init our controller
  controller.init();
}

// call init() as soon as the page is fully loaded
window.addEventListener('DOMContentLoaded', init, false);
```

## Scroll View Delegation

```
// scroll view has snapped to the nearest page without deceleration
controller.scrollViewDidEndScrollingAnimation = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// scroll view has scrolled to a new location following a deceleration animation
controller.scrollViewDidEndDecelerating = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// syncs the page control to show the currently visible page in the scroll view
controller.syncPageControlToScrollView = function () {
  this.pageControl.currentPage = Math.round(
    this.scrollView.contentOffset.x / this.scrollView.size.width);
};
```

## Scroll View Delegation

```
// scroll view has snapped to the nearest page without deceleration
controller.scrollViewDidEndScrollingAnimation = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// scroll view has scrolled to a new location following a deceleration animation
controller.scrollViewDidEndDecelerating = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// syncs the page control to show the currently visible page in the scroll view
controller.syncPageControlToScrollView = function () {
  this.pageControl.currentPage = Math.round(
    this.scrollView.contentOffset.x / this.scrollView.size.width);
};
```

## Scroll View Delegation

```
// scroll view has snapped to the nearest page without deceleration
controller.scrollViewDidEndScrollingAnimation = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// scroll view has scrolled to a new location following a deceleration animation
controller.scrollViewDidEndDecelerating = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// syncs the page control to show the currently visible page in the scroll view
controller.syncPageControlToScrollView = function () {
  this.pageControl.currentPage = Math.round(
    this.scrollView.contentOffset.x / this.scrollView.size.width);
};
```

## Scroll View Delegation

```
// scroll view has snapped to the nearest page without deceleration
controller.scrollViewDidEndScrollingAnimation = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// scroll view has scrolled to a new location following a deceleration animation
controller.scrollViewDidEndDecelerating = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// syncs the page control to show the currently visible page in the scroll view
controller.syncPageControlToScrollView = function () {
  this.pageControl.currentPage = Math.round(
    this.scrollView.contentOffset.x / this.scrollView.size.width);
};
```

## Scroll View Delegation

```
// scroll view has snapped to the nearest page without deceleration
controller.scrollViewDidEndScrollingAnimation = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// scroll view has scrolled to a new location following a deceleration animation
controller.scrollViewDidEndDecelerating = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// syncs the page control to show the currently visible page in the scroll view
controller.syncPageControlToScrollView = function () {
  this.pageControl.currentPage = Math.round(
    this.scrollView.contentOffset.x / this.scrollView.size.width);
};
```

## Scroll View Delegation

```
// scroll view has snapped to the nearest page without deceleration
controller.scrollViewDidEndScrollingAnimation = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// scroll view has scrolled to a new location following a deceleration animation
controller.scrollViewDidEndDecelerating = function (theScrollView) {
  this.syncPageControlToScrollView();
};

// syncs the page control to show the currently visible page in the scroll view
controller.syncPageControlToScrollView = function () {
  this.pageControl.currentPage = Math.round(
    this.scrollView.contentOffset.x / this.scrollView.size.width);
};
```

## Page Control Event Handling

```
<div class="ad-page-control"
     ad-number-of-pages="4"
     ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>
```

```
controller.pageControlValueChanged = function (event) {
  // get new scroll view content offset from page control’s current page
  var x = this.pageControl.currentPage * this.scrollView.size.width;
  // update the scroll view’s content offset with an animation
  this.scrollView.setContentOffsetAnimated(new ADPoint(x, 0), true);
};
```

## Page Control Event Handling

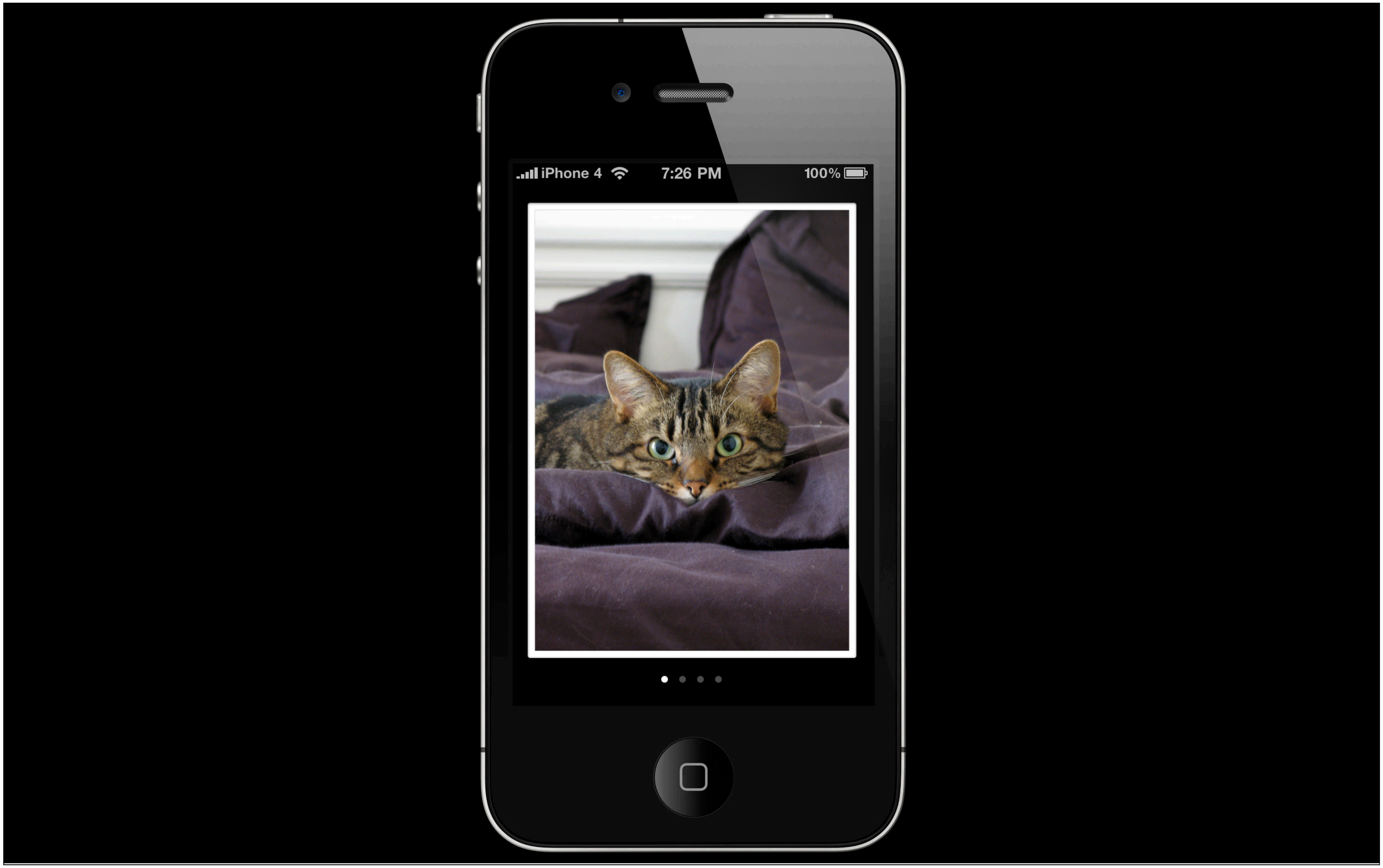
```
<div class="ad-page-control"
     ad-number-of-pages="4"
     ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>
```

```
controller.pageControlValueChanged = function (event) {
  // get new scroll view content offset from page control’s current page
  var x = this.pageControl.currentPage * this.scrollView.size.width;
  // update the scroll view’s content offset with an animation
  this.scrollView.setContentOffsetAnimated(new ADPoint(x, 0), true);
};
```

## Page Control Event Handling

```
<div class="ad-page-control"
     ad-number-of-pages="4"
     ad-onControlValueChange="controller.pageControlValueChanged(event)"></div>
```

```
controller.pageControlValueChanged = function (event) {
  // get new scroll view content offset from page control’s current page
  var x = this.pageControl.currentPage * this.scrollView.size.width;
  // update the scroll view’s content offset with an animation
  this.scrollView.setContentOffsetAnimated(new ADPoint(x, 0), true);
};
```


iPhone 4
7:26 PM
100%

## Summary

### Views and controls

- Two techniques
  - **Programmatic**—using the JS APIs
  - **Declarative**—using strictly HTML & CSS

### There’s more…

Scroll Views

Toolbars

Search Bars

Page Controls

Flow Views

Switches

Progress Views

Sliders

Navigation Bars

Picker Views

Tab Bars

Carousel Views

Table Views

Buttons

Segmented Controls

Rating Controls

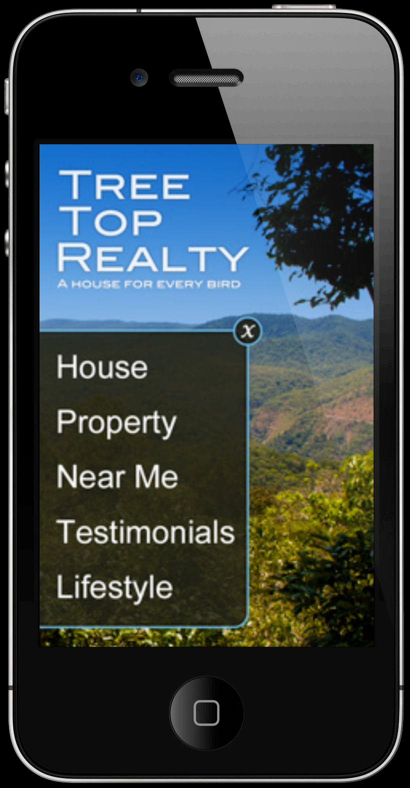
## Agenda

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. **Working with Views and Controls**
4. Using View Controllers

## Working with View Controllers

ADViewController

## Modular Architecture



Screen One
Photos

Top Screen
Menu

Screen Two
Maps

## Modular Architecture

### Performance considerations

- Loading too much initial content strains the network
- Too many elements in the tree bogs down rendering performance

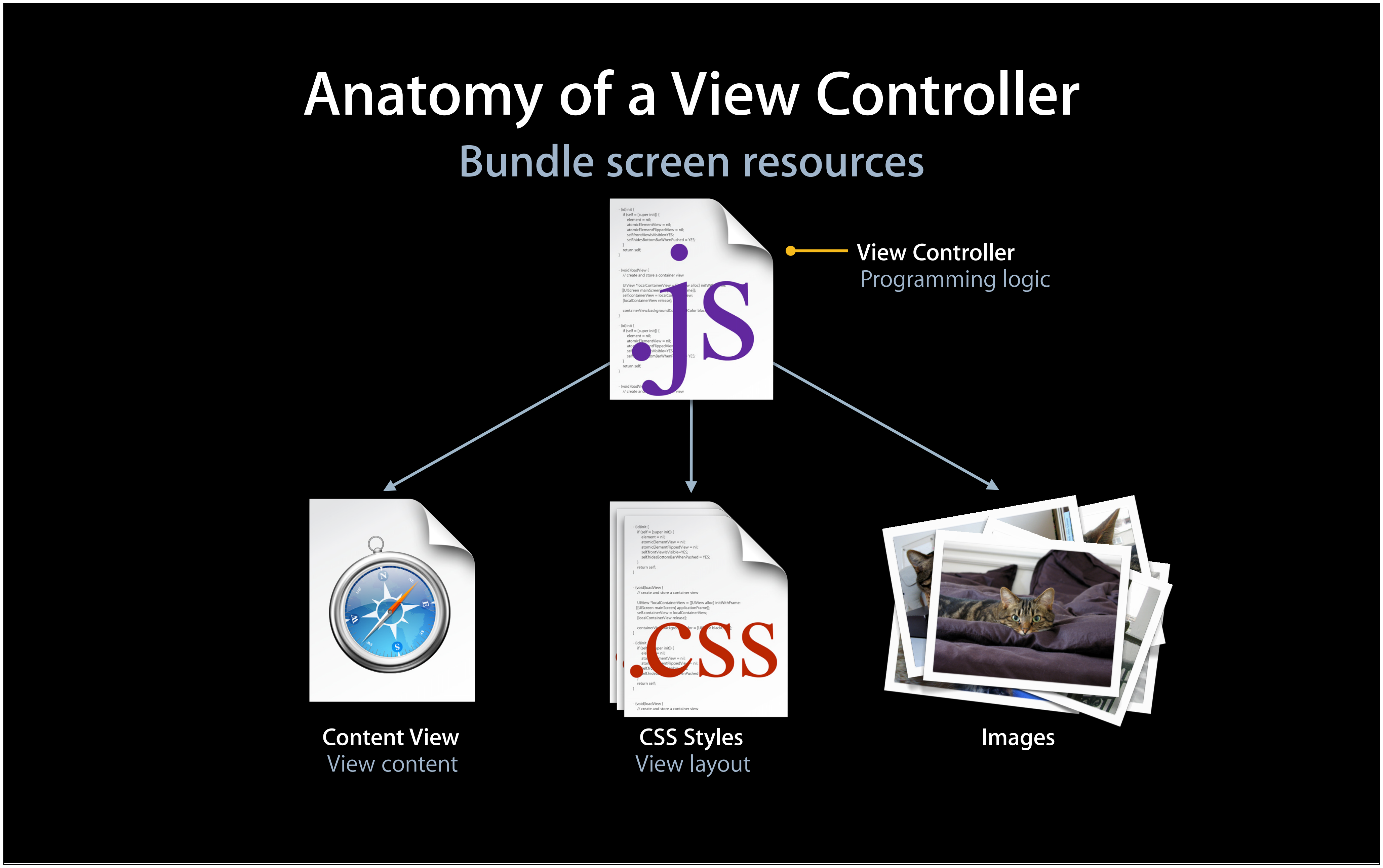
## Incremental Download

## Incremental Display

## View Controllers

### Modular architecture

- Abstract loading of a number of resources
- Add and remove views as needed



# Anatomy of a View Controller

## Bundle screen resources

## ADViewController

### Configuration

```
MenuController.superclass = ADViewController;
```

```
function MenuController () {
  this.callSuper({
    id : 'menu',
    requiredFileURIs: {
      contentView : 'views/menu.html',
      stylesheets : ['css/menu.css'],
      images: ['images/stig1.png',
               'images/stig2.png',
               'images/stig3.png']
    }
  });
};
```

## ADViewController

### Configuration

```
MenuController.superclass = ADViewController;

function MenuController () {
  this.callSuper({
    id : 'menu',
    requiredFileURIs: {
      contentView : 'views/menu.html',
      stylesheets : ['css/menu.css'],
      images: ['images/stig1.png',
               'images/stig2.png',
               'images/stig3.png']
    }
  });
};
```

## ADViewController

### Configuration

```
MenuController.superclass = ADViewController;

function MenuController () {
  this.callSuper({
    id : 'menu',
    requiredFileURIs: {
      contentView : 'views/menu.html',
      stylesheets : ['css/menu.css'],
      images: ['images/stig1.png',
               'images/stig2.png',
               'images/stig3.png']
    }
  });
};
```

## ADViewController

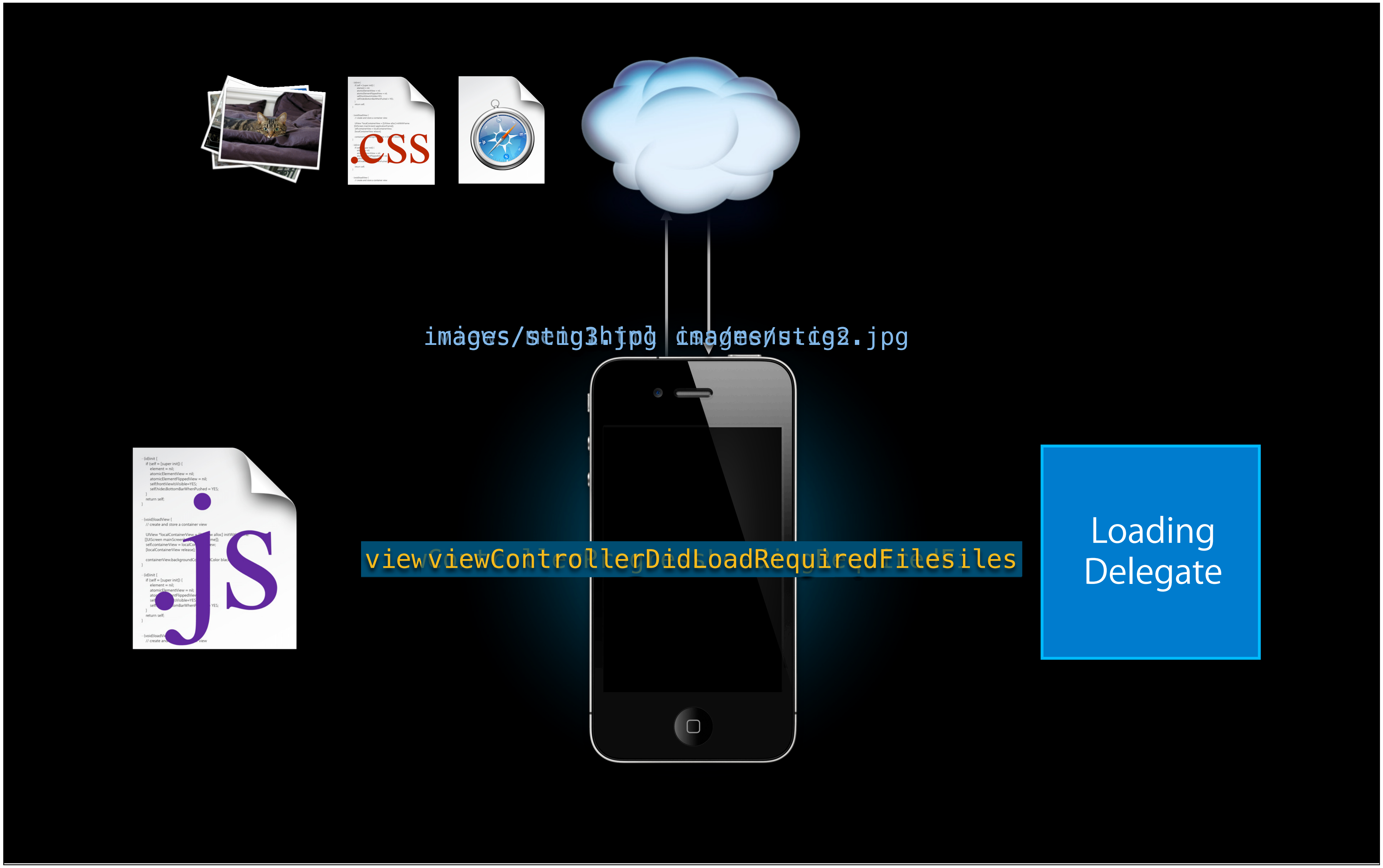
### Configuration

```
MenuController.superclass = ADViewController;

function MenuController () {
  this.callSuper({
    id : 'menu',
    requiredFileURIs: {
      contentView : 'views/menu.html',
      stylesheets : ['css/menu.css'],
      images: ['images/stig1.png',
               'images/stig2.png',
               'images/stig3.png']
    }
  });
};
```

## ADViewController

### Configuration

```
MenuController.superclass = ADViewController;

function MenuController () {
  this.callSuper({
    id : 'menu',
    requiredFileURIs: {
      contentView : 'views/menu.html',
      stylesheets : ['css/menu.css'],
      images: ['images/stig1.png',
               'images/stig2.png',
               'images/stig3.png']
    }
  });
};
```


.css
.js
images/sting3.jpg
images/sting2.jpg
viewControllerDidLoadRequiredFiles
Loading Delegate

## ADViewController

### Configuration

```
MenuController.superclass = ADViewController;

function MenuController () {
  this.callSuper({
    id : 'menu',
    requiredFileURIs: {
      contentView : 'views/menu.html',
      stylesheets : ['css/menu.css'],
      images: ['images/stig1.png',
               'images/stig2.png',
               'images/stig3.png']
    }
  });
};
```

## View Controller Unique Identifier

### The `id` property

- Uniquely identifies a view controller among all others
  - `ADViewController.instances.menu;`
- Eases CSS matching
  - `contentView` uses identifier as-is for its `id` attribute
  - `view` uses identifier with `-container` suffix for its `id` attribute

## Declarative Features

## More Convenience

## View Controllers

### Common Programming Tasks

- Obtaining references to objects in the content view
- Responding to user interaction triggered in content view
- Transitioning between screens

## Outlets

- Automatically reference any view or element in content view
  - `<div ad-outlet="title"></div>`
- Store references in `outlets` property
  - `this.outlets.title`

## Actions

- Automatically register a callback in the context of the view’s controller
  - `<div ad-action="playAudio"></div>`
  - `MyController.prototype.playAudio()`

## Transitions

- Automatically trigger transition to new view controller by its `id`
    - `<div ad-transitions-to="maps"></div>`
    - Abstracts loading and removing views in one synchronized transaction

## Summary

### View controllers

- Automatic loading of screen resources bundle
- Incremental screen display with automated transitions
- Standardized common coding tasks
- More declarative features
- Less code, fewer common errors

## Summary

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Summary

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Summary

1 Motivations and Features of iAd JS

2 Core JavaScript Enhancements

3 Working with Views and Controls

4 Using View Controllers

## Summary

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Summary

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Summary

1. Motivations and Features of iAd JS
2. Core JavaScript Enhancements
3. Working with Views and Controls
4. Using View Controllers

## Rich Media

HTML5

## Mini Apps

## 100% Web Standards

iAd

## More Information

**Vicki Murley**
Safari Technologies Evangelist
vicki@apple.com

**Download**
iAdDeveloper Package
http://developer.apple.com/iAd

**Documentation**
iAd JS Reference Library
http://developer.apple.com/iphone/iad/prerelease/library/navigation/

**Apple Developer Forums**
http://devforums.apple.com

## Related Sessions

### iAd

| | |
|---|---|
| Creating Content with iAd JS, Part I | Marina<br>Thursday 9:00AM |
| Integrating Ads with iAd (Repeat) | Pacific Heights<br>Friday 9:00AM |

## Related Sessions

### Web technologies

| | |
|---|---|
| Delivering Audio and Video Using Web Standards, Part I | Nob Hill<br>Friday 10:15AM |
| Delivering Audio and Video Using Web Standards, Part 2 | Nob Hill<br>Friday 11:30AM |

| | |
|---|---|
| CSS Effects, Part I: UI Elements and Navigation | Marina<br>Tuesday 3:15PM |
| CSS Effects, Part 2: Galleries and 3D Effects | Marina<br>Tuesday 4:30PM |
| Adding Touch and Gesture Detection to Web Pages on iPhone OS | Nob Hill<br>Wednesday 2:00PM |

## Labs

| | |
|---|---|
| Safari on iPhone OS Lab | Internet & Web Lab B<br>Thursday 2:00PM |

## Q&A

WWDC10