

UIWebView

The Most Versatile Class in UIKit

Nick Dalton
CTO, Pervasent

Pervasant

- Consulting company specializing in mobile application development.
- In business 8 years.
- Focused on the iPhone the past 18 months.
- Recent projects:
 - Chipotle Mexican Grill Ordering
 - Family Guy Wallpapers
 - NFL Logos



When I give a lecture, I accept that people look at their watches, but what I do not tolerate is when they look at it and raise it to their ear to find out if it stopped.

- *Marcel Achard*

Sample Code

Available here:

<http://iPhoneIncubator.com>

Main	
UIWebView	
Load local resources	>
Data URI scheme image	>
Activity indicator	>
Transparent web view	>
Simple JavaScript	>
Call JavaScript function	>
Inject JavaScript function	>
Form data from web view	>
Results list - plain	>

Local Resources

- Load images, CSS and JavaScript from your application bundle.

← Main Load local resources

This HTML page uses CSS loaded from the app bundle. The image below is also loaded from the app bundle.



Local images can be loaded in CSS.



You can combine local resources with external resources, if you provide an absolute URL for the external resources. Here's an image loaded from an external web site.



LocalResourcesInWebViewController

```
NSString *basePath = [[NSBundle mainBundle] bundlePath];
```

```
NSURL *baseURL = [NSURL fileURLWithPath:basePath];
```

```
[webView loadHTMLString:htmlString baseURL:baseURL];
```

The HTML

```
<html>
<head>
<link href="Style.css" rel="stylesheet" type="text/css">
</head>
<body>
  <p>Local image:</p>
  

  <p>Local images can also be loaded in CSS:</p>
  <div id="myimage">&nbsp;</div>

  <p>Combine with external resources.</p>
  
</body>
</html>
```

When is this useful?

- In help files inside your application.
- Use images and CSS to make the HTML look better.

Data URI Scheme



- Use “data:” to embed image data inside HTML.

The HTML

`<p>This is a data encoded image.</p>`

```

```

When is this useful?

- If you want to download and cache a web page, it's *much* easier to deal with a single file, than one HTML and many image files.
- If you can't change baseURL to point to the application bundle, you can still include images using this technique.

Activity Indicator

- Display an activity indicator while HTML is loading.



Please be patient...



The large image is Courtesy of Apple Computer, Inc. Photographer: Gary Parker

ActivityIndicatorWebViewController

```
[activityIndicator startAnimating];
```

```
...
```

```
[webView loadHTMLString:htmlString baseURL:nil];
```

```
- (void)webViewDidFinishLoad:(UIWebView *)theWebView {  
    [activityIndicator stopAnimating];  
}
```

```
- (void)webView:(UIWebView *)theWebView  
didFailLoadWithError:(NSError *)error {  
    [activityIndicator stopAnimating];  
}
```

When is this useful?

- You should show an activity indicator anytime the application is accessing external HTML.
- Even with large amounts of HTML loaded from the application bundle, there can be a delay.

Transparent Web View

Main Transparent web view

- Display HTML over a background image.

**Text in a transparent
web view.**

**Text in a transparent
web view.**



TransparentWebViewController

```
webView1.backgroundColor = [UIColor clearColor];  
[webView1 loadHTMLString:htmlString baseURL:nil];
```

```
webView2.backgroundColor = [UIColor clearColor];  
webView2.opaque = NO;  
[webView2 loadHTMLString:htmlString baseURL:nil];
```

The HTML

```
<html>
  <head>
    <style type="text/css">
      body {
        background-color: transparent;
      }
    </style>
  </head>
  <body>
    <h1>Text in a transparent web view.</h1>
  </body>
</html>
```

When is this useful?

- Make your HTML screens look better. The standard white browser background is not that exciting.
- You can add background images using HTML/CSS. But a UIImageView is more efficient than loading the image through HTML.

Simple JavaScript

← Main The Meaning of Life

42

- Use JavaScript to access data in the web view.

SimpleJavaScriptWebViewController

```
- (void)webViewDidFinishLoad:(UIWebView *)theWebView {  
    self.title = [theWebView  
stringByEvaluatingJavaScriptFromString:@"document.title"];  
}
```

When is this useful?

- You can access any DOM property this way.
- Also works on web pages you load from external URLs.

JavaScript callback



Box width:

Click box to grow



- Call your Objective-C code from JavaScript in the UIWebView.

The HTML

```
<style type="text/css">
div#box
{
    background-color: blue;
    display: block;
    position: absolute;
    width: 50;
    height: 50;
    -webkit-transition-property: width;
    -webkit-transition-duration: 2.0s;
    -webkit-transition-timing-function: default;
}
</style>
<script type="text/javascript">
    function grow() {
        box = document.getElementById( "box" );
        var old_width = box.offsetWidth;
        box.style.width = old_width * 1.5;
        box.addEventListener( 'webkitTransitionEnd', function( event )
{ document.location = 'webkitTransitionEnd/' + box.style.width; }, false );
        }
    }
</script>
```

CallbackJavaScriptWebViewController

```
- (BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:
(NSURLRequest *)request navigationType:
(UIWebViewNavigationType)navigationType {
    BOOL shouldStartLoadWithRequest = YES;

    NSURL *url = [request URL];
    NSString *urlString = [url absoluteString];
    NSRange range = [urlString rangeOfString:@"webkitTransitionEnd"];
    if (range.location != NSNotFound) {
        NSString *value = [urlString substringFromIndex:range.location +
range.length + 1];
        boxWidthField.text = value;
        shouldStartLoadWithRequest = NO;
    }

    return shouldStartLoadWithRequest;
}
```

When is this useful?

- When you need notifications or data from the JavaScript/HTML layer.

Form Data

- Receive data from an HTML form.

[Main](#) Form data from web view

This is the UIWebView

These are UIView components

The received value is:

The HTML

```
<form name="input" action="form_submit" method="get">  
<input type="text" name="value" />  
<input type="submit" value="Submit" />  
</form>
```

DataFromWebViewController

```
- (BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:  
(NSURLRequest *)request navigationType:  
(UIWebViewNavigationType)navigationType {
```

```
    BOOL shouldStartLoadWithRequest = YES;
```

```
    NSURL *url = [request URL];  
    NSString *urlString = [url absoluteString];  
    NSRange range = [urlString rangeOfString:@"form_submit"];  
    if (range.location != NSNotFound) {  
        NSString *value = [self getParameterValueFromURL:url  
parameterName:@"value"];  
        textField.text = value;  
        shouldStartLoadWithRequest = NO;  
    }  
  
    return shouldStartLoadWithRequest;  
}
```

When is this useful?

- Another way to get data from the HTML world into Objective-C.
- Creating data entry screens on the iPhone is tedious. Doing it with HTML is often quicker and still provides a good user experience.

Handle mailto: links

- By default mailto: links open the Mail app.
- With this technique you can capture the tap on the link and use the in app email functionality in 3.0 instead.

← Main Handle mailto: links

If you have OS 3.x then this link will use the in app email functionality instead of opening the Mail app.

foo@bar.com

MailToWebViewController

```
- (BOOL)webView:(UIWebView *)aWbView shouldStartLoadWithRequest:
(NSURLRequest *)request navigationType:
(UIWebViewNavigationType)navigationType {
    NSURL *url = [request URL];
    NSString *scheme = [url scheme];
    if ([scheme isEqualToString:@"mailto:"]) {
        // Check for 2.0 or 3.0, and if 3.0, that the device can send email
        Class mailClass = (NSClassFromString(@"MFMailComposeViewController"));
        if (mailClass != nil) {
            if ([mailClass canSendMail]) {
                [self launchMailComposeViewController];
            } else {
                [self launchMailApp];
            }
        } else {
            [self launchMailApp];
        }
    }

    return NO;
} else {
    return YES;
}
}
```

When is this useful?

- This technique can be used to intercept and provide special handling of any type of URL.

Results List

- A flexible alternative to UITableView.
- Allows rich text and varying “row” heights.

← Main Results list - plain

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
foo, bar

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium
fu, bar

totam rem aperiam.

eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

ResultListPlainWebViewController

```
NSString *filePathSearchResultRow = [[NSBundle mainBundle]
    pathForResource:rowFileName ofType:@"html"];
NSString *searchResultRow = [NSString
    stringWithContentsOfFile:filePathSearchResultRow];

NSMutableString *htmlSearchResults = [[NSMutableString alloc] init];

NSArray *results = [self dummyResults];
for (NSDictionary *result in results) {
    NSString *text = [self highlightText:@"dolor" inText:[result
    objectForKey:@"text"]];
    [htmlSearchResults appendFormat:searchResultRow, CUSTOM_HTTP_SCHEME, [result
    objectForKey:@"id"], text, [result objectForKey:@"keywords"]];
}

NSString *filePathSearchResultPage = [[NSBundle mainBundle]
    pathForResource:pageFileName ofType:@"html"];
NSString *searchResultPage = [NSString
    stringWithContentsOfFile:filePathSearchResultPage];
NSString *html = [NSString stringWithFormat:searchResultPage, htmlSearchResults];

NSString *basePath = [[NSBundle mainBundle] bundlePath];
NSURL *baseURL = [NSURL fileURLWithPath:basePath];
[webView loadHTMLString:html baseURL:baseURL];
```

The Page HTML

```
<html>
<head>
<style type="text/css">
// iUI Style Sheet ...
</style>
</head>
<body >
<ul title="Search Results" selected="true">
%@
</ul>
</body>
</html>
```

The Row HTML

```
<li><a href="%@:%@">%@<br/><i>%@</i></a></li>
```

← Main Results list - plain

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
foo, bar

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium
fu, bar

totam rem aperiam.

eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

When is this useful?

- When you have a table with rows of varying heights and you don't want to worry about calculating the heights of each row.
- When you want to have rich text formatting in the cells.
- If you ever contemplate having one web view per UITableViewCell, then use a single UIWebView for the whole table instead.

Play YouTube Videos



- Play YouTube videos without leaving your application.

PlayYouTubeInWebViewController

```
[self embedYouTube:@"http://www.youtube.com/watch?v=WL2L\_Q1AR\_Q"  
frame:CGRectMake(20, 20, 100, 100)];
```

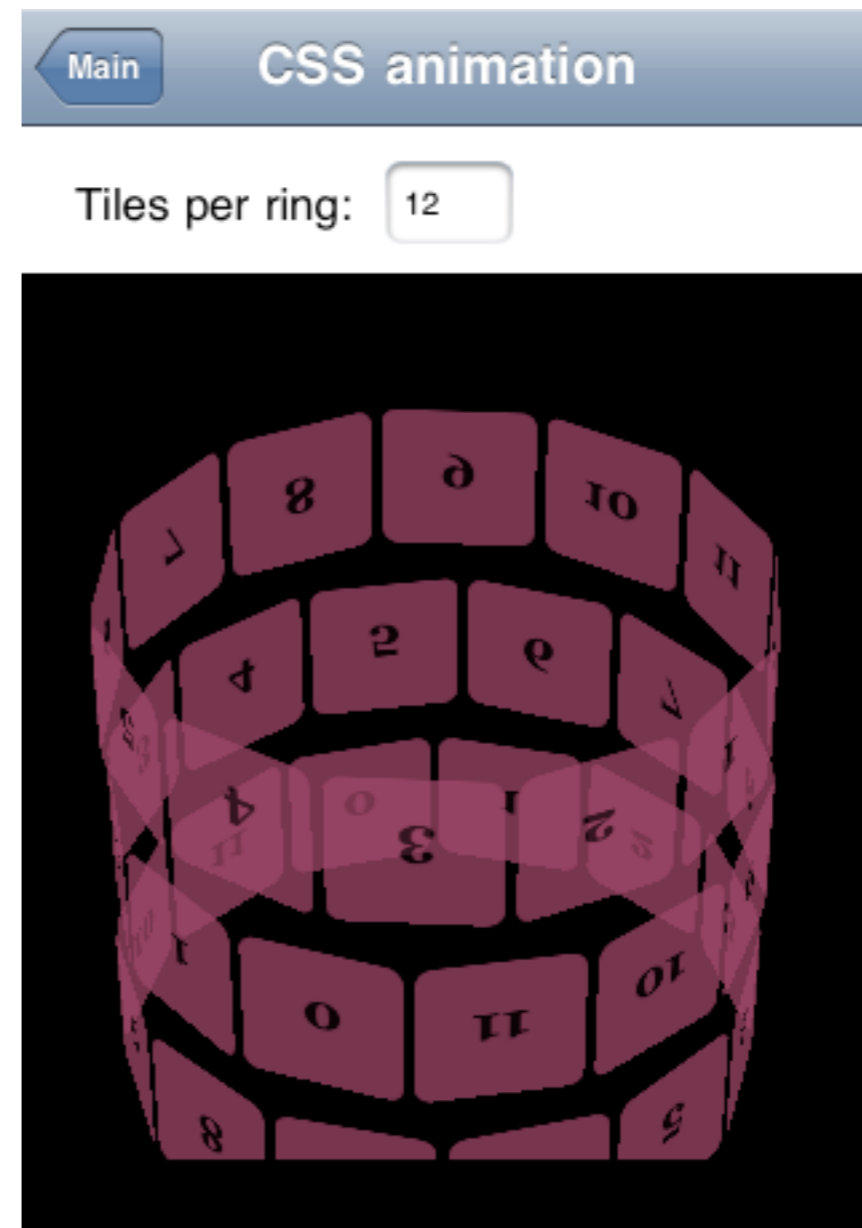
```
- (void)embedYouTube:(NSString *)urlString frame:(CGRect)frame {  
    NSString *embedHTML = @"\  
        <html><head>\  
        <style type=\"text/css\">\  
        body {\  
        background-color: transparent;\  
        color: white;\  
        }\  
        </style>\  
        </head><body style=\"margin:0\">\  
        <embed id=\"yt\" src=\"%@\" type=\"application/x-shockwave-flash\" \\  
        width=\"%0.0f\" height=\"%0.0f\"></embed>\  
        </body></html>";  
    NSString *html = [NSString stringWithFormat:embedHTML, urlString,  
frame.size.width, frame.size.height];  
    UIWebView *videoView = [[UIWebView alloc] initWithFrame:frame];  
    [videoView loadHTMLString:html baseURL:nil];  
    [self.view addSubview:videoView];  
    [videoView release];  
}
```

When is this useful?

- When you want to play YouTube videos without leaving your application.

CSS Animation

- Mobile Safari implements a lot of fun CSS animations and transitions.
- This demo also passes in data from Objective-C to the JavaScript code.



PosterCircleWebViewController

```
- (IBAction)start:(id)sender {
```

```
    NSString *jsFunctionCall = [NSString  
stringWithFormat:@"init(%@);", numTilesField.text];
```

```
    [webView stringByEvaluatingJavaScriptFromString:  
jsFunctionCall];
```

```
    ...
```

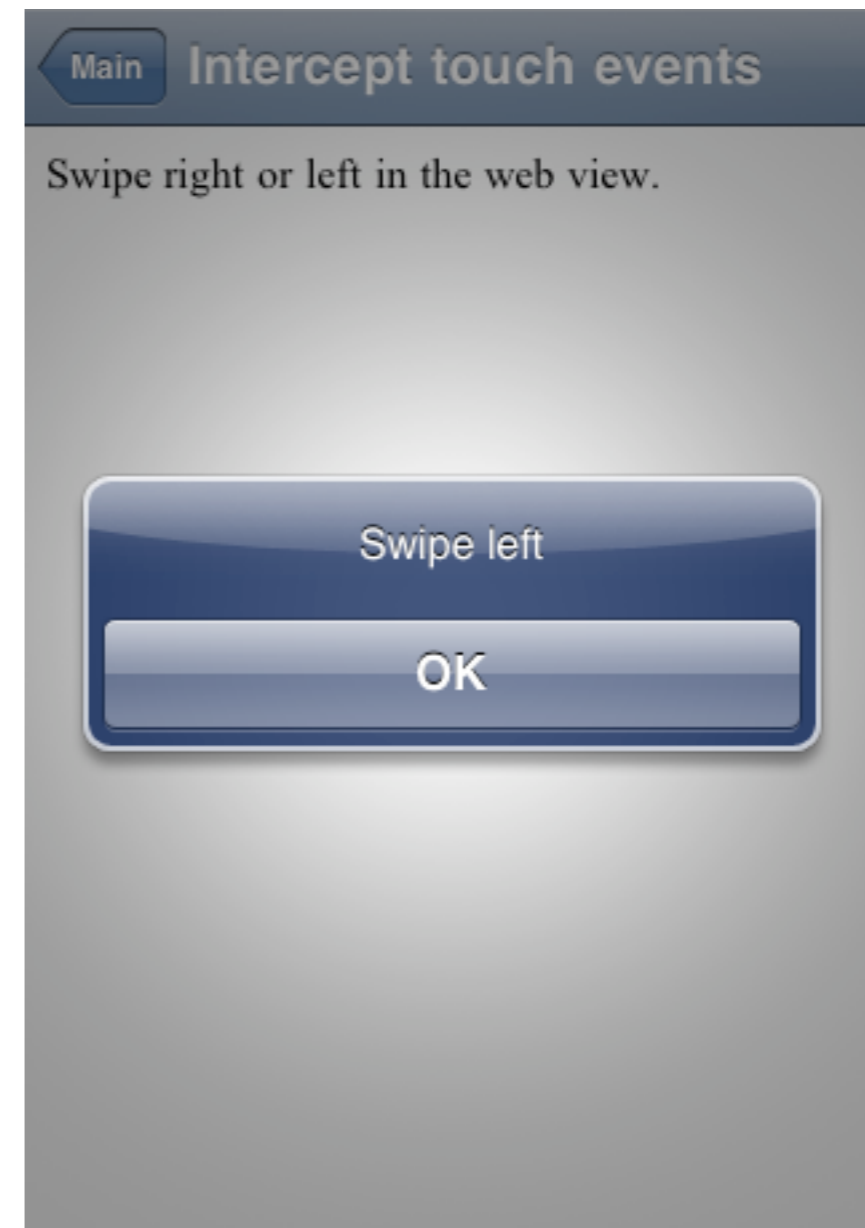
```
}
```

When is this useful?

- CSS animations are hardware accelerated and run surprisingly fast.
- Not a replacement for OpenGL.

Intercept Touches

- UIWebView is like a black hole that eats touch events.
- This technique allows you to detect swipes and other gestures.



SnoopWindow

```
- (void)sendEvent:(UIEvent *)event {
    UITouch *touch = [[event allTouches] anyObject];
    UIView *touchView = [touch view];

    if (touchView && [touchView isDescendantOfView:webView]) {
        //
        // touchesBegan
        //
        if (touch.phase==UITouchPhaseBegan) {
            startTouchPosition = [touch locationInView:self];
            startTouchTime = touch.timestamp;
        }

        //
        // touchesEnded
        ///
        ...

        [super sendEvent:event];
    }
}
```

SnoopWindow

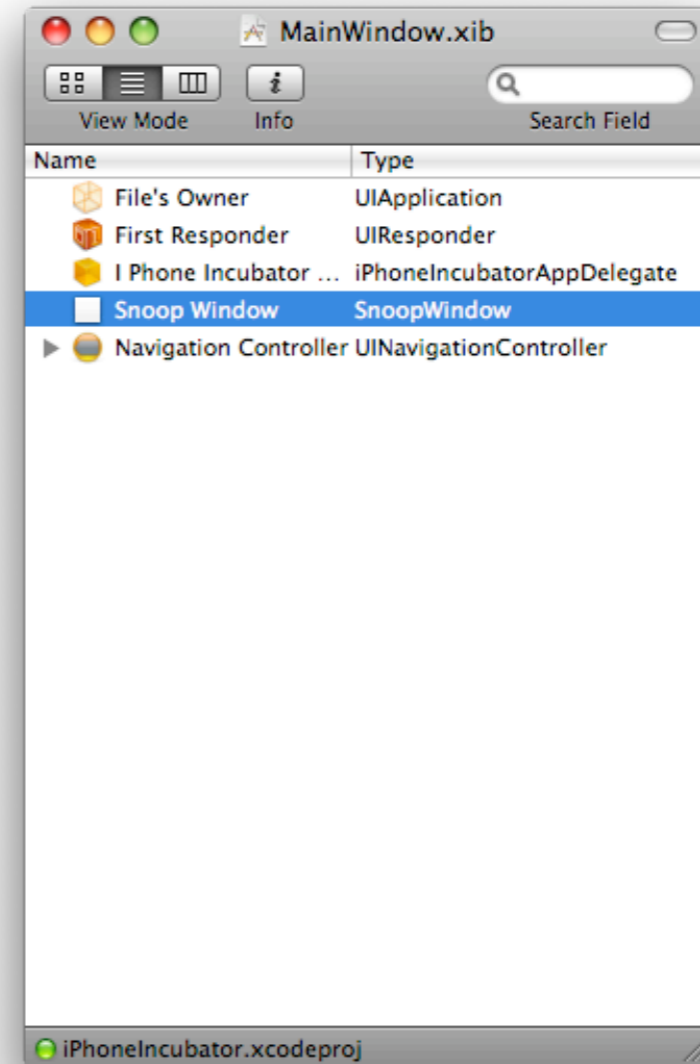
```
// It appears to be a swipe.  
if (startTouchPosition.x < currentTouchPosition.x) {  
    DLog(@"swipe right");  
    [[NSNotificationCenter defaultCenter]  
postNotificationName:NOTIFICATION_SWIPE_RIGHT  
object:touch];  
} else {  
    DLog(@"swipe left");  
    [[NSNotificationCenter defaultCenter]  
postNotificationName:NOTIFICATION_SWIPE_LEFT  
object:touch];  
}
```

SwipeWebViewController

```
    iPhoneIncubatorAppDelegate *applicationDelegate =  
    (iPhoneIncubatorAppDelegate *)[[UIApplication sharedApplication]  
    delegate];  
    SnoopWindow *snoopWindow = (SnoopWindow *)applicationDelegate.window;  
    snoopWindow.webView = webView;  
  
    [[NSNotificationCenter defaultCenter] addObserver:self  
        selector:@selector(swipeLeft:)  
        name:NOTIFICATION_SWIPE_LEFT  
        object:nil];  
}  
  
- (void)swipeLeft:(NSNotification *)notification {  
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:nil  
        message:@"Swipe left"  
        delegate:nil  
        cancelButtonTitle:@"OK"  
        otherButtonTitles:nil];  
  
    [alert show];  
    [alert release];  
}
```

MainWindow.xib

- Replace UIWindow with SnoopWindow in MainWindow.xib



When is this useful?

- When you want to detect any touch events from a web view.
- Examples:
 - Swipe to go to the next/previous page.
 - Create new gestures to interact with your screen.

Pinch to Resize Text

- Processing a multi-touch gesture like pinch can use the same mechanism as for swipes.
- This example resizes the text of the UIWebView using CSS in response to pinching.

Main Pinch to resize text

Use the pinch gesture to increase and decrease the size of this text.

SnoopWindow

```
//  
// touchesMoved  
//  
if (touch.phase==UITouchPhaseMoved) {  
    if ([[event allTouches] count] > 1) {  
        CGPoint currentPosition1 = [[allTouches objectAtIndex:0] locationInView:self];  
        CGPoint currentPosition2 = [[allTouches objectAtIndex:1] locationInView:self];  
  
        CGFloat currentFingerDistance = CGPointDist(currentTouchPosition1,  
currentTouchPosition2);  
        CGFloat previousFingerDistance = CGPointDist(previousTouchPosition1,  
previousTouchPosition2);  
  
        if (fabs(currentFingerDistance - previousFingerDistance) > ZOOM_DRAG_MIN) {  
            NSNumber *movedDistance = [NSNumber numberWithFloat:currentFingerDistance -  
previousFingerDistance];  
  
            if (currentFingerDistance > previousFingerDistance) {  
                DLog(@"zoom in");  
                [[NSNotificationCenter defaultCenter] postNotificationName:NOTIFICATION_ZOOM_IN  
object:movedDistance];  
            } else {  
                DLog(@"zoom out");  
                [[NSNotificationCenter defaultCenter]  
postNotificationName:NOTIFICATION_ZOOM_OUT object:movedDistance];  
            }  
        }  
    }  
}
```

ZoomWebViewController

```
- (void)zoomIn:(NSNotification *)notification {
    [self setFontSize:fontSize + 0.1];
    [webView setNeedsDisplay];
}

- (void)zoomOut:(NSNotification *)notification {
    [self setFontSize:fontSize - 0.1];
    [webView setNeedsDisplay];
}

- (void)setFontSize:(CGFloat)size {
    if (size >= FONT_SIZE_MIN && size <= FONT_SIZE_MAX) {
        fontSize = size;

        NSString *jsFunctionCall = [NSString
stringWithFormat:@"document.body.style.fontSize = %f + 'em';", fontSize];
        NSLog(@"jsFunctionCall: %@", jsFunctionCall);
        NSLog(@"jsFunctionCall response: %@", [webView
stringByEvaluatingJavaScriptFromString:jsFunctionCall]);
    }
}
```

When is this useful?

- This particular example is a bit contrived, but the technique can be used to handle multi-touch events that normally just end up in the UIWebView “black hole”.

Thank You!

www.Pervasive.com
iPhoneIncubator.com

or search for

iphone development blog
iphone developer nick

Session Feedback: m.360idev.com