

# iPhone™ and iPod® touch Programming: Handling Touch Interactions and Events for Mobile Safari™

Richard Wagner



**Wiley Publishing, Inc.**

Wrox Blox: *iPhone and iPod touch Programming: Handling Touch Interactions and Events for Mobile Safari* By Richard Wagner - ISBN: 978-0-470-26022-7  
Copyright 2008, Wiley Publishing, Inc.

This PDF is exclusively for your use in accordance with the Wrox Blox Terms of Service. No part of it may be reproduced or transmitted in any form by any means without prior written permission from the publisher. Redistribution or other use that violates the Wrox Blox Terms of Service or otherwise violates the U.S. copyright laws is strictly prohibited.

# Contents

<b>How iPhone and iPod touch Handle Events</b>	<b>1</b>
<b>Detecting an Orientation Change</b>	<b>3</b>
Changing a Style Sheet When Orientation Changes	6
Changing Element Positioning Based on Orientation Change	10
<b>Capturing Two-Finger Scrolling</b>	<b>13</b>
<b>Simulating a Drag-and-Drop Action</b>	<b>17</b>
<b>Trapping for Key Events with the On-Screen Keyboard</b>	<b>20</b>
<b>Summary</b>	<b>21</b>
<b>About Richard Wagner</b>	<b>22</b>

# iPhone and iPod touch Programming: Handling Touch Interactions and Events for Mobile Safari

An essential part of any Web 2.0 application is the ability to respond to events triggered by the user or by a condition that occurs on the client: the clicking of a button, the pressing of a key, the scrolling of a window. While the user interacts with an HTML element, the entire document, or the browser window, JavaScript serves as the watchful eye behind the scenes that monitors all of this activity taking place and fires off events as they occur.

With their touch screen interface, iPhone and iPod touch are all about direct interactivity with the user. As a result, you would expect any Mobile Safari application you create to be able to handle the variety of finger taps, flicks, swipes, and pinches that a user naturally performs as they interact with their mobile device. However, because of the current capabilities of Mobile Safari browser, you have to work with these interactions differently from what you might expect.

## How iPhone and iPod touch Handle Events

When working with touch interactions and events for iPhone and iPod touch, keep in mind that the finger is not a mouse. As a result, the traditional event model that web developers are so used to working with does not always apply in this new context. This is both good news and bad news for the application developer. The good news is that much of the touch interaction that iPhone and iPod touch are famous for is built right into Mobile Safari. As a result, you do not need to write any code to handle the basic touch interactions of the user. Flick-scrolling, pinching and unpinching, and one-finger scrolling are those sorts of user inputs that come for free. The bad news is that the developer is greatly constrained in his or her ability to work with the full suite of JavaScript events and override built-in behavior. As a result, certain user interactions that have become a staple to web developers now are impossible to utilize or require tricky, dumbed-down workarounds.

The general rule of thumb for Mobile Safari event handling is that no events trigger *until* the user's finger leaves the touch screen. The implications are significant:

- ❑ The `onmousedown` event handler fires only after a mouse up event occurs (but before `onmouseup` is triggered). As a result, the `onmousedown` event is rendered useless.
- ❑ The `onmousemove` event handler is unsupported. However, on rare occasions, our tests show that Mobile Safari may trigger an `onmousemove` event; thus developers should not assume that these handlers will never be called.
- ❑ `:hover` does not work.

In addition, you cannot trap for zoom events associated with the window. First, Mobile Safari provides no built-in event handler support for zooming out or zooming in. Second, you cannot perform a work-around by polling the window for width changes, since the width remains the same regardless of the current zoom factor.

You cannot trap for events associated with a user switching between pages in Mobile Safari. The `onfocus` and `onblur` events of the `window` object are not triggered when the focus moves off or on a page. Additionally, when another page becomes the active page, JavaScript events (including polling events created with `setInterval()`) are not fired. However, the `onunload` event of the `window` object is triggered when the user loads a new page in the current window.

Table 1 lists the events that are fully supported and unsupported. Table 2 identifies the events that are partially supported.

**Table 1: JavaScript Event Compatibility**

Supported events	Unsupported events
<code>formfield.onblur</code>	<code>document.onkeydown</code>
<code>formfield.onchange</code>	<code>document.onkeypress</code>
<code>formfield.onclick</code>	<code>document.onkeyup</code>
<code>formfield.onfocus</code> <code>formfield.onkeydown</code>	<code>form.onsubmit</code>
<code>formfield.onkeyup</code>	<code>formfield.ondblclick</code>
<code>formfield.onkeypress</code>	<code>formfield.onmouseenter</code>
<code>formfield.onmouseout</code>	<code>formfield.onmouseleave</code>
<code>formfield.onmouseover</code>	<code>formfield.onmousemove</code>
<code>formfield.onmouseup</code>	<code>formfield.onselect</code>
<code>form.onreset</code>	<code>window.oncontextmenu</code>
<code>window.onload</code>	<code>window.onerror</code>
<code>window.onmousewheel</code>	<code>window.onresize</code>
	<code>window.onscroll</code>