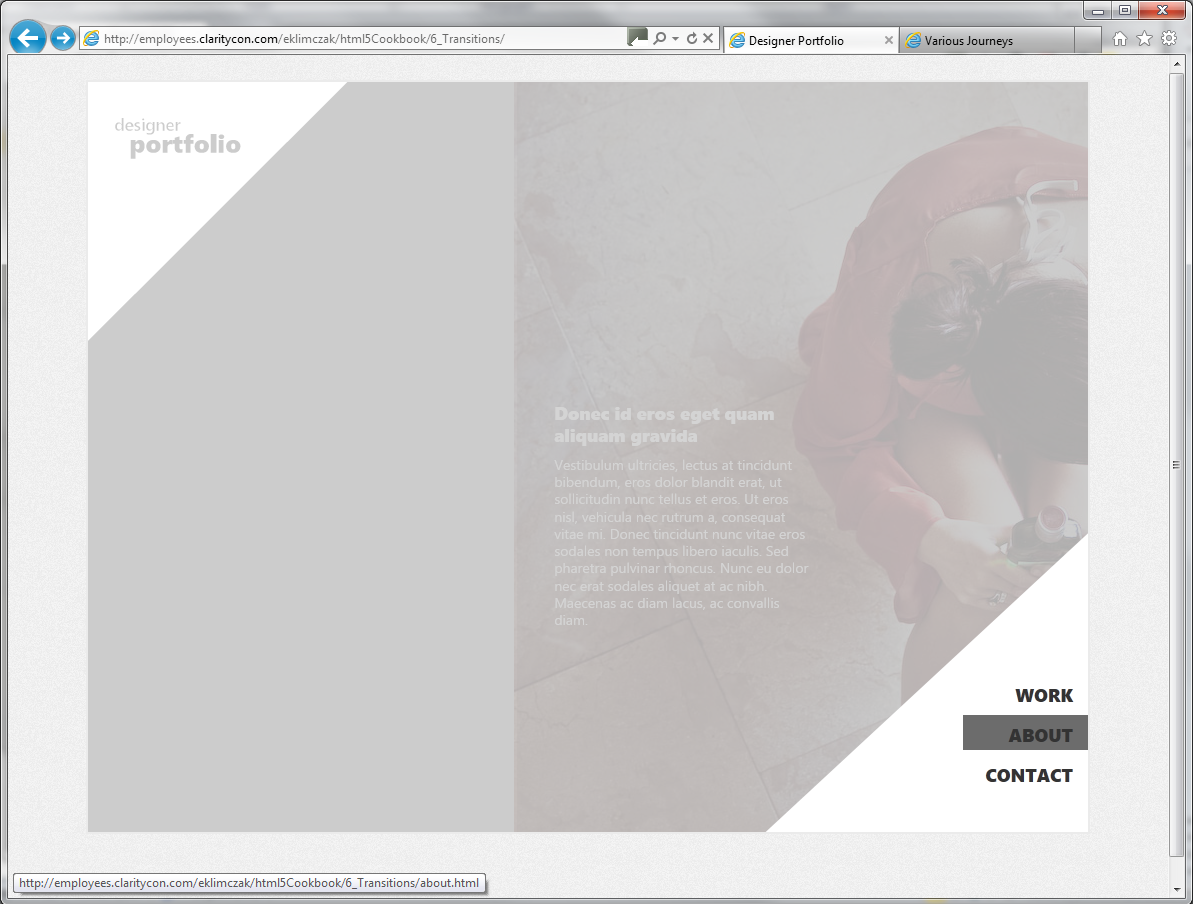
|  |
| --- |
| HTML5 Elements Cookbook |
| H5E Experiment: Transitions: Loading Content on Demand |
| DRAFT  Published 15 July, 2011 |
|  |

|  |
| --- |
|  |
| H5E Scout Team  Windows Web Partners  Microsoft Corporation  Microsoft Confidential |

# Executive Summary

Part of the aspirational experience of an application is the sense that the content within the app is available immediately, without download wait time or choppy performance between pages. With web pages, using HTML and JavaScript to loading pages dynamically in the background provides an app-like feel without incurring the overhead of loading all of the content simultaneously.



*The image above shows the content of a new page being loaded. The page is fading in while it is scrolled into position.*

## Scope

This document describes an experiment conducted by the H5E scout team using Clarity Consulting. Our objective is to test the limits of HTML5 solving real-world partner questions. This document assumes an existing knowledge of JavaScript and jQuery. This document does not supersede any requirements or instructions provided by the IE team.

## Keywords

Dynamic loading, History API, jQuery, JavaScript

## Contact us

To contact us for questions or support, please email Chewy Chong ([ChewyC](mailto:ChewyC?subject=HTML5%20Cookbooks)). Feedback is welcome.

Contents

Executive Summary 2

Scope 2

Keywords 2

Contact us 2

Dynamically Loading Pages: Overview 4

Video 4

Walkthrough 4

How do I build this with HTML5? 6

Primary files in this solution 6

Step 1: Hi-Jack the onclick event on the links 7

Step 2: Listen for popstate/hashchange 7

Step 3: Use XHRequest (via $.ajax) to get the contents 7

Extra: 8

Complete Code 9

Conclusions and Recommendations 11

Resources 11

Relevant Web sites and specifications 11

Microsoft Resources 11

Appendix A: About H5E 12

What is an HTML5 Elements Cookbook? 12

Contact us 12

Copyright and trademark information 12

Document Revision History 13

# Dynamically Loading Pages: Overview

To better understand how HTML5-based features can be used to provide aspirational experiences, the H5E team has categorized these experiences into fundamental elements.

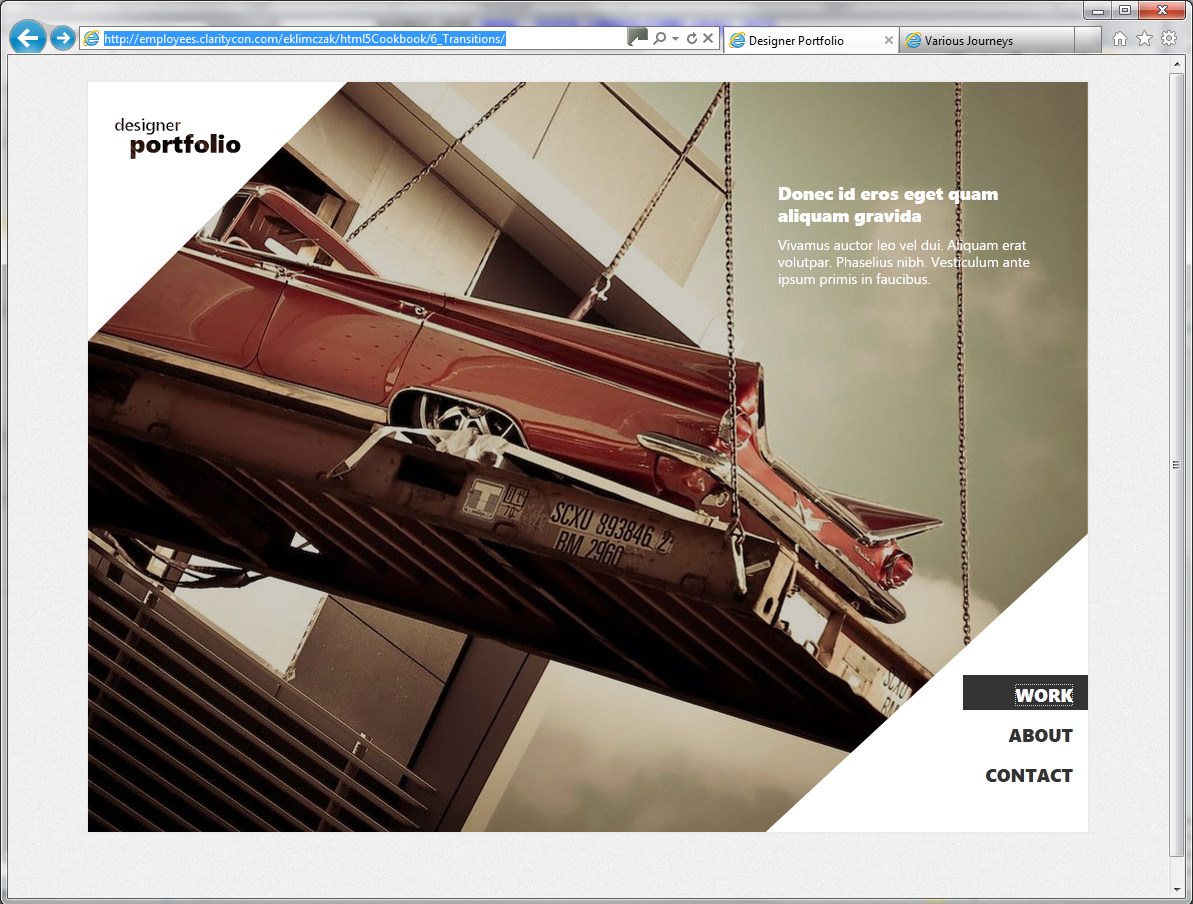
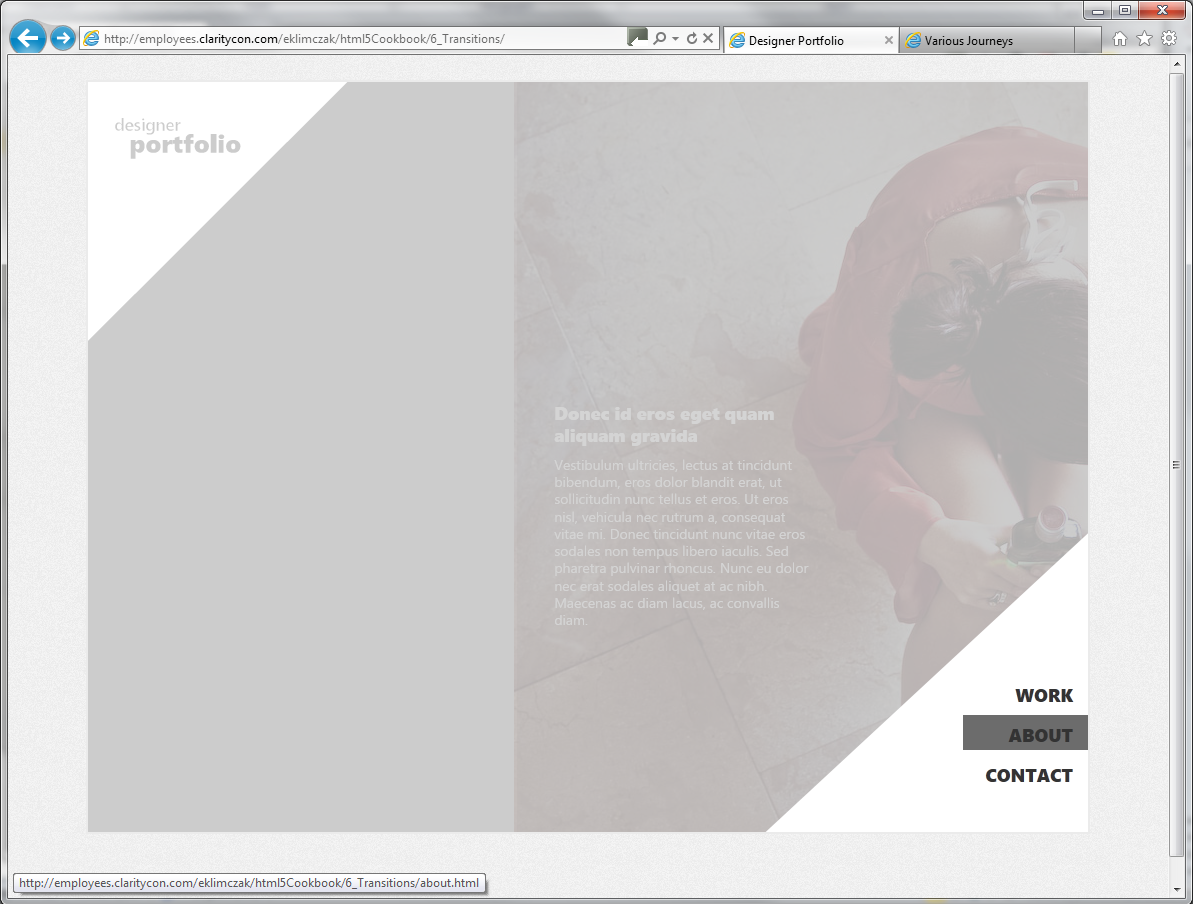
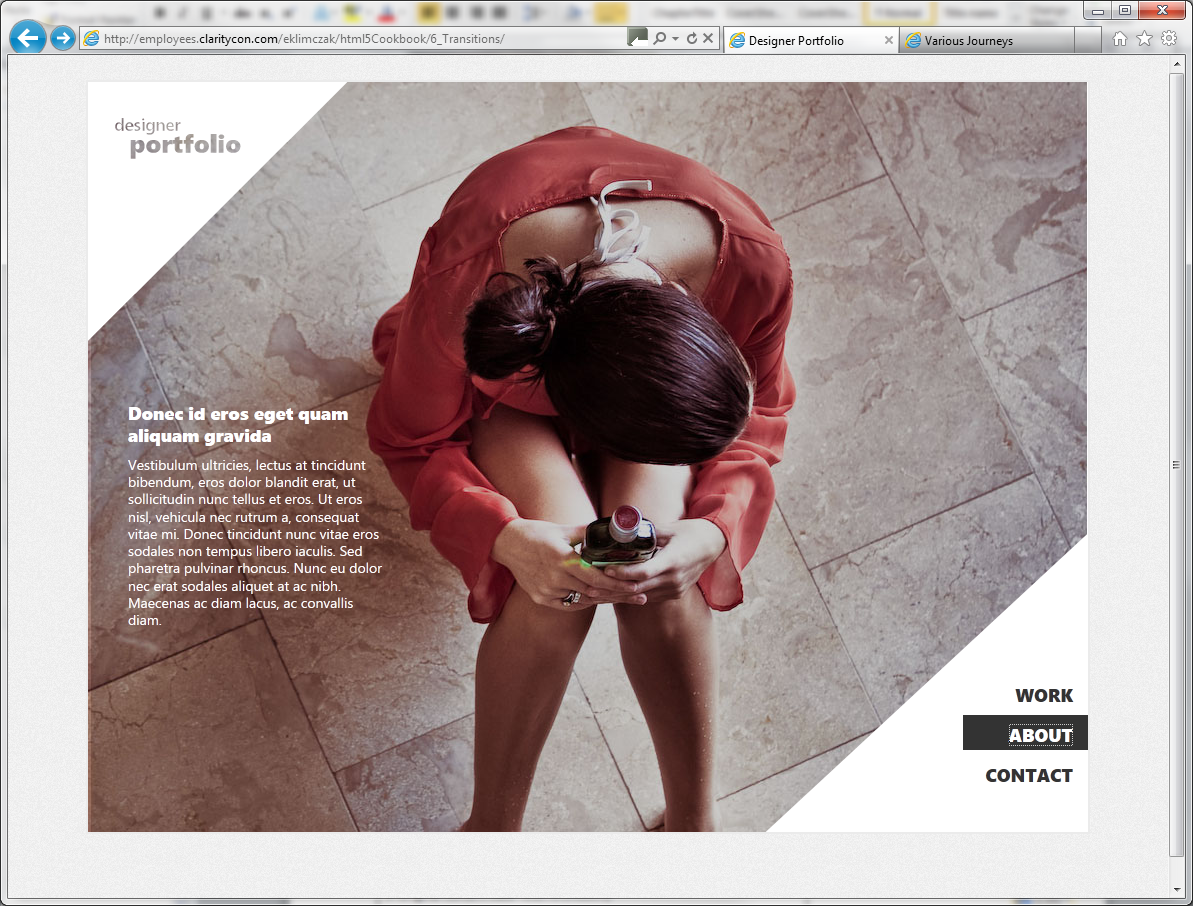
Dynamically loading pages in the background enables us to create an app-like feel for web applications and sites, but without the overhead of loading all of the content simultaneously.

## Video

A video of this demo can be seen at the following: <http://www.vimeo.com/26427651>

## Walkthrough

This demonstration includes the following experience:

1. The user clicks a link in the site navigation panel.;  
   
2. The new page loads in the background and smoothly slides into view.  
   
3. The new page is loaded and displayed.  
   

# How do I build this with HTML5?

There are multiple approaches to creating this experience using HTML5-related technologies. In this case, this HTML5 Experiment focuses on the following:

* Step 1: Hi-jack the event occurring on a link click
  + Update URL, fallback to hash.
* Step 2: Add a listener for the page change onpopstate/onhashchange
  + Get the new URL
* Step 3: Use XHRequest to get the contents of the new page
  + Load the content into a div and slide into place.

***NOTE****: Many of the HTML5 Experiments are still under development. Our initial target is to build prototypes that work on current HTML5-supported browsers and tablet devices, including the iPad. The experiments do not aim for full cross-browser support at this stage, but we will likely build in graceful degradation in future updates.*

To ensure that users have a similar cross-browser experience, the following table describes the compatibility of the solutions in this document:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| HTML5 Logo**HTML5 Feature** | IE6.0 | IE7.0 | IE8.0 | IE9.0 | IEPP | Chrome11 | | Chrome12 | Safari4.x | Safari5.x | Firefox3.6 | Firefox4.x | Firefox5.x | Opera11 |
| History API |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| onhashchange |  |  |  |  |  |  | |  |  |  |  |  |  |  |
| $.ajax |  |  |  |  |  |  | |  |  |  |  |  |  |  |
|  | | | | | | | Full Support | | | | | | |  |
|  | | | | | | | Supported with Shim | | | | | | |  |
|  | | | | | | | No current support | | | | | | |  |

***NOTE****: In general, shims are not incorporated into the HTML5 Experiments at this stage. If a shim or polyfill is required for cross-browser support, see* [*http://browserexperiments.com*](http://browserexperiments.com) *for details on shim implementation.*

## Primary files in this solution

### Source Location

<https://github.com/molant/BrowserExperiments/tree/master/cookbook/6_Transitions>

### Sample Location

<http://employees.claritycon.com/eklimczak/html5Cookbook/6_Transitions/>

## Step 1: Hi-Jack the onclick event on the links

Using jQuery for event delegation, we can hijack the action of every click on a link. In this example we exclude non-relative domains (links to external sites). To exclude these domains, use the following:

// Listen for any anchor tag call

$('a:not([href^=http])').live("click", function(e){

// prevent the default reloading the page

e.preventDefault();

…

});

Instead we call HTML5’s **history.pushstate** or otherwise attach a hashtag to the url.

// Listen for any anchor tag call

$('a:not([href^=http])').live("click", function(e){

// prevent the default reloading the page

e.preventDefault();

// Does the browser support popstate?

if("history" in window && !!history.pushState){

// Goodbye hashchange, you weren't great at deeplinking and SEO but you did create a nice UX.

// Is this a change in location?

history.pushState( {}, this.title, this.href );

// Safari doesn't update our search string

$(window).trigger('popstate');

return false;

}

window.location.hash = this.href;

});

## Step 2: Listen for popstate/hashchange

Listen for these events triggered, and get the URL that was requested.

// Listen out to the new change in

$(window).bind("hashchange popstate", function(){

// Log the State

var url = ( window.location.hash.replace(/\#/,'') || window.location.pathname );

});

## Step 3: Use XHRequest (via $.ajax) to get the contents

Here the url variable from step two is the target of our Ajax request. On success we are putting the content into a temporary div and then filtering the content of that and inserting it into the page.

// Make a call to grab the new page

$.ajax({

url: url,

type: "get",

dataType: "html",

success: function (html) {

// Insert the content into the page

var $div = $('<div></div>');

$div.get(0).innerHTML = html;

$('#page\_content').html( $div.first().html() );

},

error: function () {

}

});

## Extra:

### Mimicking the hashchange

The following snippet provides backwards support for mimicking the hashchange event.

// Hashchange event doesn't get triggered in all browsers to make it compatible

// For IE 6,7

(function(){

if( !history.pushState && ( !("onhashchange" in window) || ($.browser.msie&&$.browser.version==='7.0') ) ){

var hash = window.location.hash;

setInterval(function(){

if(window.location.hash!==hash){

$(window).trigger('hashchange');

hash = window.location.hash;

}

}, 500);

}

})();

## Complete Code

(function () {

// Listen for any anchor tag call

$('a:not([href^=http])').live("click", function (e) {

// prevent the default reloading the page

e.preventDefault();

// Does the browser support popstate?

if ("history" in window && !!history.pushState) {

// Goodbye hashchange, you weren't great at deeplinking and SEO but you did create a nice UX.

// Is this a change in location?

history.pushState({}, this.title, this.href);

// Safari doesn't update our search string

$(window).trigger('popstate');

return false;

}

window.location.hash = this.href;

});

// Listen out to the new change in

$(window).bind("hashchange popstate", function () {

// Get the new url

var url = (window.location.hash.replace(/\#/, '') || window.location.pathname);

// Add the page

$('#page\_title').html(url);

// Just load test\* pages

if(!url.match(/test/)){

$('#page\_content').html("");

return;

}

// Make a call to grab the new page

$.ajax({

url: url,

type: "get",

dataType: "html",

success: function (html) {

// Insert the content into the page

var $div = $('<div></div>');

$div.get(0).innerHTML = html;

$('#page\_content').html( $div.first().html() );

},

error: function () {

}

});

});

// Hashchange event doesn't get triggered in all browsers to make it compatible

// For IE 6,7

(function () {

if (!history.pushState && (!("onhashchange" in window) || ($.browser.msie && $.browser.version === '7.0'))) {

var hash = window.location.hash;

setInterval(function () {

if (window.location.hash !== hash) {

$(window).trigger('hashchange');

hash = window.location.hash;

}

}, 500);

}

})();

})();

# Conclusions and Recommendations

* **Scalable**: Yes
* **Easy to implement**: relatively
* **Failover:**  yes
* **SEO:** Yes, however sharing the hash is not.
* **Plumbing**: relatively easy, can be implemented partially on a site.

# Resources

## Relevant Web sites and specifications

|  |  |
| --- | --- |
| Demo | <http://browserexperiments.com/History/simple.html> |
| History.js shim demo | <http://browserexperiments.com/History/> |
| Clarity consulting demo | <http://employees.claritycon.com/eklimczak/html5Cookbook/6_Transitions/> |

## Microsoft Resources

|  |  |
| --- | --- |
| H5E Primary Contact | Chewy Chong ([ChewyC](mailto:%20chewyc?subject=HTML5%20Experiments%20Cookbooks)) |
| H5E Development Contact | Anton Molleda Quintana ([v-anmoll](mailto:v-anmoll?subject=HTML5%20Experiments%20Cookbooks)) |

# Appendix A: About H5E

## What is an HTML5 Elements Cookbook?

Each HTML5 Elements Cookbook reflects a case study of an aspirational experience that is provided by a native or component-based application. The HTML5 Experiments that are conducted by the H5E team use HTML5 and related technologies to replicate these experiences. Our primary objective is to learn from these experiments to determine if an HTML5 alternative to component-based or native implementations is both possible, and practical. Each Cookbook provides a description of the element and technical details of the HTML5 replication of that feature. We also include recommendations on whether it makes sense to pursue this approach.

## Contact us

If you need assistance with technical solutions or have a best practice to share, please contact us by sending email to Chewy Chong ([ChewyC](mailto:chewyc?subject=H5E%20Cookbooks%20and%20Documentation)).

## Copyright and trademark information

This document is Microsoft Confidential.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

 2011 Microsoft Corporation. All rights reserved.

# Document Revision History

|  |  |  |
| --- | --- | --- |
| **Reviser** | **Date** | **Revisions** |
| **i-andods** | 14 July 2011 | Initial draft |
| **v-jgeige** | 15 July 2011 | Light edit |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |