Daron Yöndem

Windows® Internet Explorer 9

dikeyeksen

Internet Explorer 9

Internet Explorer 9

Daron Yöndem

dikeyeksen® 8

ISBN 978-605-61677-6-8
Publisher Certificate No: 19703

Managing Editor: Suat Özdemirci
Cover Designer: Cemile Kural
Interior Layout: Rükiye Ege
Marketing Manager: Ahmet Çıtır

dikeyeksen® **Yayın Dağıtım, Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti.**
Sultanahmet Binbirdirek Mh. Klodfarer Cad. No:39 B:7-8 Fatih İstanbul
Tel: 0 212 516 32 64 **Fax**: 0 212 516 21 55
e-posta: merkez@dikeyeksen.com
web: **www.dikeyeksen.com**

Daron Yöndem

Daron Yondem is the founder of Deveload Software – a UX company based in Turkey. He is a Microsoft Regional Director with a Gold Global Impact Award in year 2009 and 2010. Daron is an international speaker leading the INETA MEA, he is a Silverlight MVP and author of two ASP.NET AJAX books and an HTML5 book. He is passionate about UX and can host sessions everywhere anytime. He has more than 180+ sessions hosted in year 2010 including a full night of free Silverlight community training called SilverNight! You can follow his thoughts at *daron.yondem.com*

Preface

Internet Explorer 9 is not only changing the way you will be browsing the internet, it will change the way internet is presented to you. With the new features like GPU support, HTML5, fastest JavaScript engine, Seamless Windows 7 experience IE9 comes as the most secure and most productive browser for the industry and the consumers.

From a developers perspective; one of the most important features of IE9 is all around browser performance. IE9 gives us the full power beneath our fingers, hardware-accelerated text, video, and graphics. Script engine performance is just one part of the overall browser performance picture. “Chakra”, the new JavaScript engine in Internet Explorer 9, interprets, compiles, and executes code in parallel and takes advantage of multiple CPU cores, when available.

On top of all these IE9 brings you Improved Interoperability through Standards Support with Cascading Style Sheets, level 3 (CSS3), Data URI, Document object model levels 3 and 3, Saleable Vector Graphics(SVG) and HTML5 support including Geolocation, , video and audio Elements, canvas , Parsing Improvements and a Selection of HTLM5 APIs

Internet Explorer 9 also introduces Windows 7 Desktop Integration, with pinned sites, a feature with which you can integrate your websites with the Windows 7 desktop. A pinned site is more than just a shortcut; it is an enabler always on the screen enabling users to go multiple websites with a single click. With the innovation of hardware-driven rendering of graphics and text, Internet Explorer 9 uses the DirectX family of Windows application programming interfaces (APIs) to enables several advances for web developers.

Internet Explorer 9 provides the features, toolkits and robust

security features that make it faster, easier, and more secure for consumers, developers and IT pros to configure, deploy, and manage the browser as part of their life and corporate environment,

This book summarizes the power IE9 provides and enables us to use and deliver a more beautiful web by the examples.

I would like to thank Daron, for his ongoing support for developers by providing this brief but comprehensive study.

With such great features in our hands, the next generation web is in front of us.

Gökşin Bakır
Microsoft

Welcome to Windows Internet Explorer 9 developer and designer book! Both as a developer or as a designer I'm sure you are eager to know what's new or changing with the new version of Internet Explorer. Is it going better? Or worse? Will that make my life easier, or maybe make the products I build better? The book will not only give you the answer of the generic question "What's in it for me?" moreover we will have the chance to dig into the technical details and real world implementations as well.

I hope you enjoy reading the new HTML5, CSS3, Windows 7 and IE9 Love! features. If you have any questions or feedback you can reach me by daron@yondem.com and get some additional thoughts on *daron.yondem.com*

Daron Yöndem
Microsoft Regional Director
Silverlight MVP
INETA MEA VP

Contents

1

CSS3

In This Chapter

As some of you may recall, Internet Explorer 8 had full support for CSS2.1 but not for CSS3, which is what developers and designers were actually looking for. The great news with Internet Explorer 9 is that it is totally committed to support CSS3. In this section, you will see the implementation of many components of CSS3.

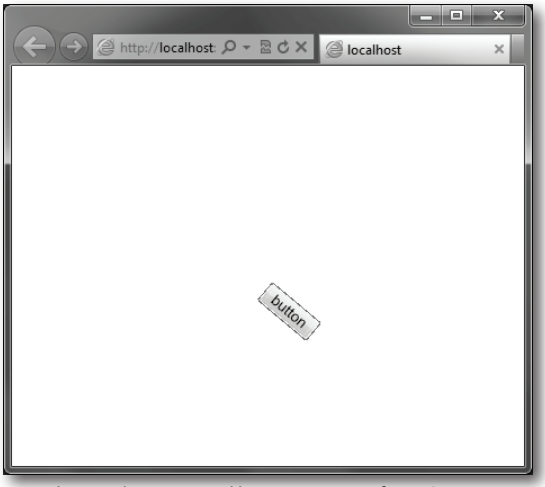
TIP It is important to remember that many CSS3 modules are still in the Working Draft or Last Call stages. Until they reach the Candidate Recommendation stage, they could change significantly. For more information visit ***www.w3.org/Style/CSS/current-work***

2D Transforms

2D Transformations can be applied to all HTML elements defined in the DOM. The current platform preview of IE9 supports two types of transformation properties. The first one is the -ms-transform property. The -ms-transform property can hold all the transformation information where the other -ms-transform-origin property defines the origin of all the transformations defined by the -ms-transform.

Below you can see a simple example where we define a CSS3 2D transformation for a simple HTML button.

```
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
  <title></title>
  <style type=”text/css”>
  .btn {
  -ms-transform: translate(200px, 200px) rotate(40deg);
  -ms-transform-origin: 50% 50%;
  }
  </style>
</head>
<body>
  <input id=”Button1” class=”btn” type=”button”
value=”button” />
</body>
</html>
```



A simple HTML button rotated by a CSS3 2D Transformation

TIP Because the CSS 2D Transforms module has not yet received Candidate Recommendation status from the W3C, both the transform and transform-origin properties must be used with the `-ms-` prefix to be recognized by Internet Explorer Platform Preview.

»The `translate(tx,ty)` function defines a 2D translation by the `vector [tx,ty]`

»The `translateX(tx)` function defines a translation by the given parameter in the x direction.

- The `translateY(ty)` function defines a translation by the given parameter in the y direction.
- The `scale(sx,sy)` function defines a 2D scale operation by the [sx,sy] vector that is described by the two parameters.
- The `scaleX(sx)` function defines a scale operation by using the [sx,1] vector, where `sx` is given as the parameter.
- The `scaleY(sy)` function defines a scale operation by using the [1,sy] vector, where `sy` is given as the parameter.
- The `rotate(angle)` function defines a 2D rotation by the angle specified in the parameter about the origin of the element, as defined by the transform-origin property.
- The `skewX(ax)` function defines a skew transformation along the x axis by the given angle.
- The `skewY(ay)` function defines a skew transformation along the y axis by the given angle.
- The `skew(ax,ay)` function defines a skew transformation along the x and y axes. The first angle parameter defines the skew on the x axis. The second angle parameter defines the skew on the y axis.

HINT Because the CSS 2D Transforms module has not yet received Candidate Recommendation status from the W3C, both the transform and transform-origin properties must be used with the `-ms-` prefix to be recognized by Internet Explorer Platform Preview.

Angle units	
deg	2

grad	Gradians
rad	Radians
turn	Turns
Time units	
ms	Milliseconds
s	Seconds

Improvements in Backgrounds & Borders

There is a long list of improvements for Backgrounds and Border specific CSS3 items. In this section, we will try to focus on some of the new functionalities that will be quite useful.

background-clip

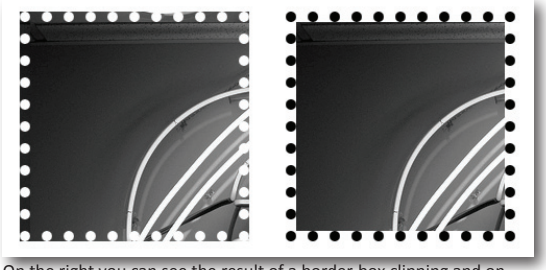
The background-clip property defines the background area of an object where the background painting can be done. There are 3 choices that you can apply:

- border-box: The background is painted within the border box.
- padding-box: The background is painted within the padding box.
- content-box:T he background is painted within the content box.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <style type="text/css">
```

```
    div {
        height:200px;
        width:200px;
        border:10px dotted white;
        background-image:url('images/back.jpg');
        background-clip:border-box;
    }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

In the example above, we have a simple `DIV` element with a dotted border and background image so that we can test how different properties will change the rendering of the browser.



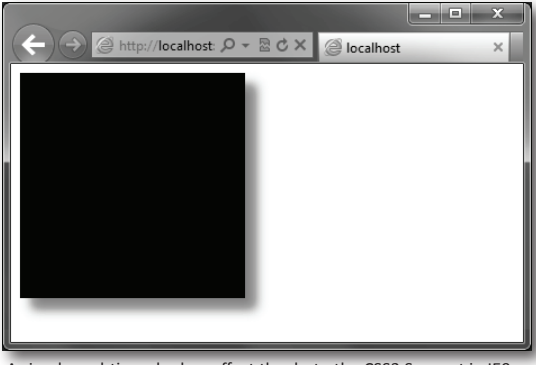
On the right you can see the result of a border-box clipping and on the left a padding-box clipping

BOX-SHADOW

This is arguably one of the most requested features in CSS and Internet Explorer. For example, how difficult is it to put a background image as a background for a dynamic content filled element in order to get a shadow effect? With Internet

Explorer 9, there is a simple solution:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <style type="text/css">
    div
    {
        height:200px;
        width:200px;
        background-color:Black;
        box-shadow: 10px 10px 15px #888;
    }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```



A simple real-time shadow effect thanks to the CSS3 Support in IE9

The box-shadow property gets four parameters. The first two define how far the shadow will be according to the x and y axis. The third one defines how blurry the shadow will be and finally the fourth one defines the color of the shadow itself.

Layering Multiple Background Images

The new background properties with CSS3 support can offer you different layers of background!

```
body
{
    background-image: url(images/back.jpg),
url(images/back2.jpg);
    background-position: top left, top right;
    background-repeat:no-repeat, repeat;
}
```

As the code above shows, we are still using the old CSS properties, but this time the value we assign can be multiple values separated by commas. For programmers, this can look like defining an array of items. The only rule you need to follow is the order of parameters and values, these should always be the same so that the exact values can be assigned to the appropriate background. Below you can find an example for the syntax.

```
body
{
    background-image: Background1, Background2;
    background-position: ValuesForBackground1, Va-
luesForBackground2;
    background-repeat: ValuesForBackground1, Valu-
esForBackground2;
}
```

Two background images assigned to the same element. The first one does not repeat but the second one does.

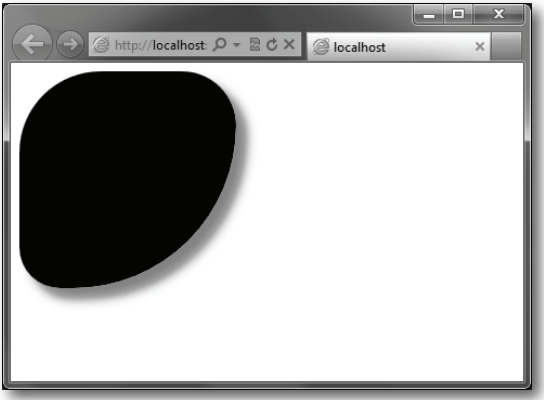
Border-radius

Creating elliptic corners was always a hard job for web developers especially in content management systems. The content has always been dynamic and there was no way to draw elliptic corners except by sending partial images from the server for each corner. Many have been waiting for this `border-radius` implement for a long time and now we have it.

```
div
{
      height:200px;
      width:200px;
      background-color:Black;
      box-shadow: 10px 10px 15px #888;
```

```
    border-radius: 100px 66.66px 200px 50px;
}
```

The example above uses both the `box-shadow` property and the `border-radius` as well. You can specify four different radius values for each corner, or provide one radius and all of the corners will be applied to that general assignment.



An amazing DIV with custom border-radius implementation!

New ways of defining Colors

Color is usually a crucial element of a great design. This includes both web design and web animation. With Internet Explorer 9 comes the alpha channel/opacity support and HSL, HSLA color models. Let's see what improvements we have in this area and how Internet Explorer 9 enhances the way that we use and manipulate colors on our website and applications.

RGBA: Red, Green, Blue and Alpha

The difference between RGB and RGBA is the additional alpha channel. In RGBA we have the regular Red, Green and Blue values which we can define as percentage values or integers, and we have the new alpha channel where we can define the transparent color. The alpha channel can be assigned a value between 0.0 (completely transparent) and 1.0 (completely opaque)

```
div
{
     height:200px;
     width:200px;
     background-color: rgba(255,0,0,0.5);
}
```

In the above example, the background-color is defined by a 50% transparent red color. This will cause the elements of color to be mixed with the parent background-color. Just as a reminder, remember that RGB values support hexadecimal notation, RGBA values do not.

HSL & HSLA

Internet Explorer 9 supports `hue-saturation-lightness` (HSL) color values. In the HSL color model, "hue" is defined as the indicated color's angle on the color wheel (for instance, red is 0 or 360, green is 120, blue is 240, and so on). "Saturation" and "lightness" are expressed as percentages. For example, the following CSS declaration defines a red background like this:

```
div
    {
```

```
    height:200px;
    width:200px;
    background-color: hsl(0,100%,50%);
}
```

Like the RGB/RGBA support, Internet Explorer 9 also extends the HSL color model with an alpha channel. As with the RGBA model, the alpha channel is defined by a value between 0.0 and 1.0.

```
div
{
    height:200px;
    width:200px;
    background-color: hsla(0,100%,50%,0.5);
}
```

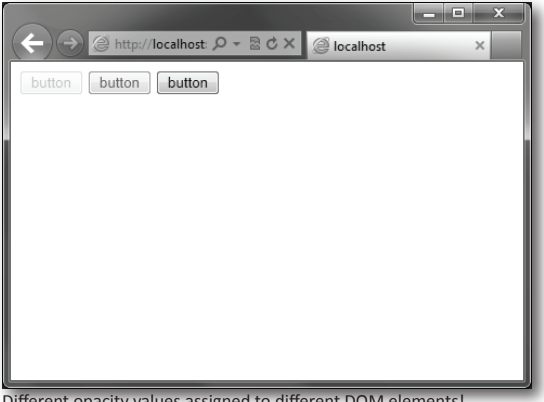
The Magic Opacity

It is like a dream come true! Have you ever needed to make any of the DOM Elements slightly transparent? With the opacity keyword support it is possible to define transparent elements.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <style type="text/css">
    .btn
    {
        opacity: 0.2;
    }
    </style>
</head>
<body>
    <input id="Button1" class="btn" type="button"
value="button" />
```

```
    <input id="Button2" class="btn"
style="opacity:0.6;" type="button" value="button"
/>
    <input id="Button3" type="button"
value="button" />
</body>
</html>
```

In the code above we have three buttons with different opacity values. The first one is assigned with the CSS class named btn which defines opacity of 0.2. The second button has its own style definition inline telling the browser to render the button with opacity of 0.6. Finally, the last button has no opacity definition at all, so it will all be visible.



Different opacity values assigned to different DOM elements!

TIP **Color Namings**
With Internet Explorer 9, you can use the transparent keyword in all the properties accepting color values. Additionally all the named colors are still available. You can find a list of named colors on this web site: ***http://samples.msdn.microsoft.com/workshop/samples/author/dhtml/colors/ColorTable.htm***

Using Custom Fonts

The text content is a big part of a web site. Being able to control the typography is a crucial part of good and successful web design. Typography control was always a big focus point for CSS, but the lack of an interoperable web font format made our lives as designers and developers much harder. With the enhancements in IE9, it is easier to deploy and use custom web fonts for dynamic text content in web site. Moreover, IE9 adds support for the **Web Open Font Format** (WOFF) and raw fonts as well.

```
@font-face
{
    font-family: CustomFont;
    src: url(images/font.ttf)
        format('truetype');
}
```

In order to define a custom font to use with CSS, we use the font-face rule. In the font-face definition, we specify a name to the font and the URL to download the font. The name of the font you specify should be used in the rest of your CSS codes as the target font name. Finally, you will need to tell the browser what type of font file you are using. In our example

above we specify the font format as “truetype”. Another full example of a custom font implementation is below;

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <style type="text/css">
        @font-face
        {
            font-family: CustomFont;
            src: url('images/FelbridgeOTS-Conden-
sed.woff');
        }
    </style>
</head>
<body>
<p style="font-family: CustomFont">Some sample
text!</p>
</body>
</html>
```

Internet Explorer 9 is still compatibly with **Embedded OpenType** (EOT). Additionally IE9 added support for the Web Open Font Format (WOFF), which repackages sfnt-based font files (TrueType, OpenType, and Open Font Format fonts) by compressing each table individually using a ZIP compression format.

Designing For Different Screens And Devices

With the range of devices and screens today it is more important to be able to cater a design to different resolutions or environments. Until now we had the media rule in CSS and we were already able to define a target media type like “screen”

or “print”. With Internet Explorer 9, we have a strong support for media queries as well.

Media queries enable you to scope a style sheet to a set of precise device capabilities. For example, you might want to design pages differently for users browsing on a mobile device with a very small screen and low resolution, or a netbook having a moderate resolution, or a standard computer with a large screen and high resolution. Media queries help us query the environment specification and manipulate the page with CSS according to the input parameters we have.

```
@media screen {
   BODY {font-size:12pt;}
}
@media print {
   BODY {font-size:8pt;}
}
```

In the above code, we use the media rule and give two different parameters to specify our target device. When the page is shown on a screen the font size will be 12 but when we print the page the font size will be 8. Internet Explorer 9 adds support to media queries for the media rule.

```
@media screen and (max-width:400px) {
    div {font-size:6pt;}
}
```

In this case our code has an additional query for the media rule. Besides targeting the type of media “screen,” we are looking for a `400px` width resolution on the screen. If the target screen has a resolution of `400px` width or less, the font size will be 6, if not the browser will pick the default values specified in the rest of the CSS code. Here is a list of parameters you can add to your media queries;

»width

»height

»device-width

»device-height

»orientation

»aspect-ratio

»device-aspect-ratio

»color

»color-index

»monochrome

»resolution

CSS3 Namespaces

Internet Explorer 9 added support to CSS3 namespaces. CSS namespaces enable a developer to declare a default namespace for a CSS file. The @namespace rule declares an XML namespace its prefix associating it with a string that represents a namespace name. Developers can have default namespaces and additional namespaces with different prefixes.

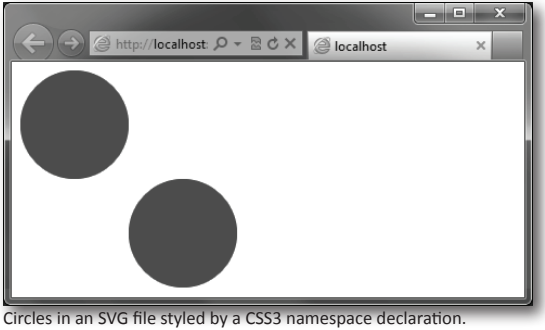
```
<style type="text/css">
    @namespace svg "http://www.w3.org/2000/svg";
    svg|circle {fill:blue;}
</style>
```

For example, above you can find an example of namespace declaration with the prefix svg. Furthermore the namespace is used to style all the circles in an SVG file or content. Be-

low you can find the full example and the appropriate view in browser.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <style type="text/css">
        @namespace svg "http://www.w3.org/2000/svg";
        svg|circle {fill:blue;}
    </style>
</head>
<body>
    <svg width="200" height="200">
        <circle cx="50" cy="50" r="50" />
        <circle cx="150" cy="150" r="50" />
    </svg>
</body>
</html>
```

None of the circles in the SVG definition have a `fill` property. The CSS declaration up in the style definition assigns the blue fill color to all the circles in an SVG area.



Circles in an SVG file styled by a CSS3 namespace declaration.

CSS3 Selectors

CSS3 Selectors are an important part of CSS definitions. In this section of our book we will cover some of the crucial concepts of CSS3 Selectors and the improvements made in IE9. We strongly suggest you get a CSS3 specific book for a much detailed look into the topic.

Class Selector

The Class Selector syntax will help you find elements with the class properties you define. In the below example, we have three DIV elements.

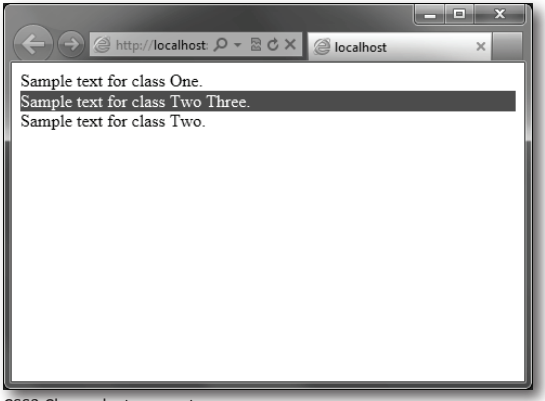
```
<div class="One">
    Sample text for class One.
</div>
<div class="Two Three">
    Sample text for class Two Three.
</div>
<div class="Two">
    Sample text for class Two.
</div>
```

As you see in the example code the first DIV has a class value of One, the second DIV has two different class values assigned as Two and Three, and finally the third DIV has a class value of Two. In our CSS Class Selector we want to select the DIV having both the Two and Three class names assigned.

```
div.Two.Three
{
  background-color:Blue;
  color:White;
}
```

In order to define such a class selector, we list all the class names we look for in a dotted ist and add it to our target ele-

ment name. The CSS Class Selector above will go find directly on our second DIV, not the first or the third one. The final result is shown below:



CSS3 Class selectors on stage.

:checked Pseudo-class

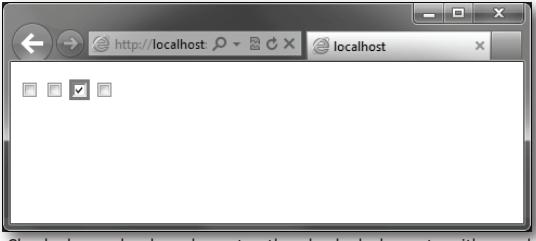
The checked Pseudo-class is part of the UI element which means that pseudo-classes, used to select UI elements (from controls such as radio buttons or check boxes) are in a certain state such as enabled, disabled, or selected/checked.

```
input:checked
{
    background-color:Red;
}
```

The above CSS pseudo definition will pick all of the input elements on the page and apply the appropriate design if the

element is checked/selected. In a full example below, the selected checkboxes will be decorated with a red background color.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <style type="text/css">
        input:checked
        {
            background-color:Red;
        }
    </style>
</head>
<body>
    <p>
        <input id="Checkbox1" type="checkbox" />
        <input id="Checkbox2" type="checkbox" />
        <input id="Checkbox3" type="checkbox" />
        <input id="Checkbox4" type="checkbox" />
    </p>
</body>
</html>
```



:Checked pseudo class decorates the checked elements with a red background color.

"UI element states pseudo-classes" supported in Internet Explorer 9 are enabled, disabled, checked and indeterminate.

:last-child Structural Pseudo-class

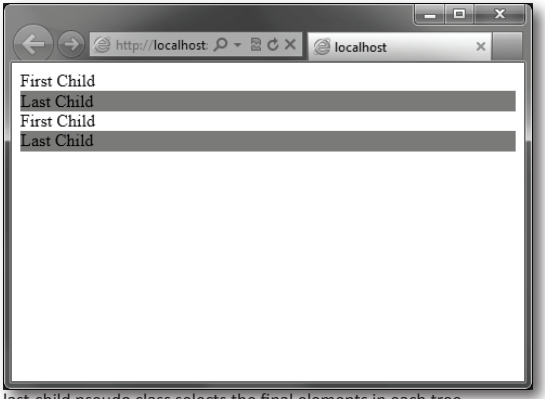
The last-child Pseudo-class is an example for Structural Pseudo-Classes. Structural pseudo-classes enable selections based on extra information in the document tree that cannot be selected using simple selectors. For Example, the last-child structural pseudo-class lets the developer pick the final element on the element tree in a context given to the CSS query.

```
<div>
    <div>First Child</div>
    <div>Last Child</div>
</div>
<div>
    <div>First Child</div>
    <div>Last Child</div>
</div>
```

In the HTML code above we have two separated DIV elements each having two other DIVs included in the content. The functionality that we want to achieve is to pick the final element from each DIV.

```
div div:last-child
{
  background-color:Red;
}
```

The CSS definition above goes through the DIVs and selects the nested DIVs final child. In our sample, the final child elements contain the text "Last Child" to make it clearly visible when the page runs.



last-child pseudo class selects the final elements in each tree.

List Of Selectors

Below you can find a list of selectors that you can use;

`X:root`	Selects an X element that is the root of the document
`X:nth-child(n)`	Selects an X element that is the n-th child of its parent
`X:nth-last-child(n)`	Selects an X element that is the n-th child of its parent, counting from the last one
`X:nth-of-type(n)`	Selects an X element that is the n-th sibling of its type
`X:nth-last-of-type(n)`	Selects an X element that is the n-th sibling of its type, counting from the last one

`X:last-child`	Selects an X element that is the last child of its parent
`X:first-of-type`	Selects an X element that is the first sibling of its type
`X:last-of-type`	Selects an X element that is the last sibling of its type
`X:only-child`	Selects an X element that is the only child of its parent
`X:only-of-type`	Selects an X element that is the only sibling of its type
`X:empty`	Selects an X element that has no children (including text nodes)
`X:enabled`	Selects an X form control element that is enabled
`X:disabled`	Selects an X form control element that is disabled
`X:checked`	Selects an X form control element that is selected
`X:indeterminate`	Selects an X form control element whose state cannot be determined

2

HTML5

In This Chapter

HTML5 is the new version of HTML, fulfilling developers' and designers' needs for building next generation web applications and web sites. Some of the HTML5 features were already implemented in Internet Explorer 8 and the number of implementations from the HTML5 Working Draft increases with the new Internet Explorer 9. In this section of our book, we will go through the list of new possibilities with HTML5 both on IE8 and IE9.

Dom Storage

Think about this as a disk space you have on the client side. Comparing to cookies, the major benefits of the Dom Storage are; it does not transmit values to the server with every request like cookies do, nor does the data in a local storage area ever expire. Moreover, in Internet Explorer, cookies can store 4 kilobytes (KB) of data where Dom Storage provides 10 megabytes (MB) for each storage area.

window.sessionStorage

The sessionStorage is a storage area where developers can save key/value pairs which are available during the current session. When a user's session is finished, all of the data will be purged.

The example below has two buttons running two different JavaScript functions and a textbox. The first button calls a JavaScript function which will save the TextBox content to the sessionStorage and the second button calls another JavaScript function loading the saved data from the `sessionStorage` to the TextBox.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        function Load() {
           document.getElementById("Text1").value =
sessionStorage.JustAText;
        }
        function Save() {
            sessionStorage.JustAText = document.
getElementById("Text1").value;
        }
```

```
    </script>
</head>
<body>
    <input id="Text1" type="text" />
    <input id="Button1" type="button" value="Save"
onclick="Save();"  />
    <input id="Button2" type="button" value="Load"
onclick="Load();"  />
</body>
</html>
```

As in the example above, in order to use the sessionStorage you can start writing a property name and assign a value to it. The Storage object supports expanded properties. Each of the property names and the values should be strings. If you have a different type of object to be saved in the DOM Storage, then you will need to serialize them.

WINDOW.LOCALSTORAGE

The local storage can save data on multiple windows and can be saved beyond the current session. The localStorage provides persistent storage areas for domains. It allows Web applications to store user data such as entire documents or `localStorage` for client side caching for performance reasons.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        function Load() {
            document.getElementById("Text1").value
= localStorage.JustAText;
        }
        function Save() {
            localStorage.JustAText = document.
getElementById("Text1").value;
```

```
        }
    </script>
</head>
<body>
    <input id="Text1" type="text" />
    <input id="Button1" type="button" value="Save"
onclick="Save();"  />
    <input id="Button2" type="button" value="Load"
onclick="Load();"  />
</body>
</html>
```

The usage of `localStorage` is the same as `sessionStorage` except the keyword “local” rather than “session” is used.

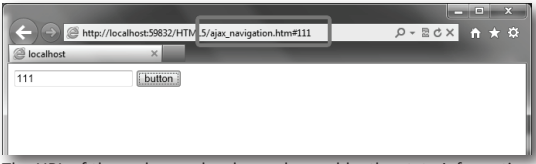
AJAX Navigation

Building an AJAX web site is an easy job thanks to the JavaScript frameworks. Nowadays it is a must to implement AJAX requests into a web site in order to provide the best user experience to the visitors of your site. However you face the big “navigation history” problem when you start building an AJAX based web site. The page content gets changed by your JavaScript code but the browser is not aware of the situation and cannot update the history accordingly. As a result when the user clicks the “back” button on the browser he/she gets redirected to the previous page visited. What if the browser knew that there was a new page or state when we manipulated the page with AJAX requests. With HTML5 this problem is solved through an additional API.

The first step is to let the browser know when there is a new state on the page. This will be the moment when you start an AJAX call to the server.

```
window.location.hash = State;
```

The call is made and you have the response save a unique state info to the window.location.hash property. When you assign a value to the hash property, the URL of the page will be changed and the history of the browser will be updated.



The URL of the web page has been changed by the state information of the hash property.

The next step is to detect when the user clicks the back or forward navigation buttons on the browser. When these buttons are pressed the browser will fire an event called onhashchange. If we attach an event listener to the JavaScript event we will be able to detect the change and get the new hash value from `window.location.hash`. After we get the value we can restore the page to the previous state with an additional AJAX call or with the data we had in a variable or cache.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        var State = 0;

        function GoToPage() {
            State += 1;
            document.getElementById("Text1").value
= State;
            window.location.hash = State;
        }
```

```
        function OnNavigate() {
            var hash = window.location.hash;
            State = hash.substr(2);
            document.getElementById("Text1").value
= State;
        }
    </script>
</head>
<body onhashchange="OnNavigate()">
    <input id="Text1" type="text" value="0" />
    <input id="Button1" type="button"
value="button" onclick="GoToPage()" />
</body>
</html>
```

Above you see a full example of the AJAX Navigation features of HTML5. In our example we have just one button and a textbox. Each time we click the button a local variable is getting changed and textboxes value changes. This means there is a new state. The values could be responses to our AJAX calls in another example.

Additionally, we have a JavaScript function attached to the body's onhashchange event. The event will be fired then the user navigates the "back" or "forward" buttons of the browser. When the `OnNavigate` method is called we go and grab the old hash value so that we can restore the page to the previous state.

Cross Document Messaging

Did you ever need to implement a web site into another and wanted them to be able to get in touch with each other? Hosting multiple web sites or contents in the same web page is difficult. You can use `IFRAME`, or `FRAMES`, or additional modal

dialogs and windows. The problem is not having or managing so many sources, it is to communicate between different environments and applications. Cross Document Messaging provides a secure and easy way to communicate these applications without going into nasty hacks.

Imagine we have a main page and another one hosted in an IFRAME. Here is the easiest implementation to get these different environments in touch with each other:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        function CallToIframe() {
            var o = document.
getElementById('AFrame');
            o.contentWindow.postMessage('Welcome');
        }
    </script>
</head>
<body>
    <div>
        <input id="Button1" type="button"
value="Click Me" onclick="CallToIframe();" />
        <iframe height="200" width="200"
src="document_messaging_2.htm" id="AFrame" />
    </div>
</body>
</html>
```

The first code example above shows our main page having the frame. Additionally, we have buttons which trigger communication between the frames. Finally, our CallToIframe method finds the Iframe on the page and calls a method called postMessage with a string parameter. The string parameter will go directly to the sub IFrame page. To be able to get the

message from inside the IFRAME, there is another implementation step we should opt for.

```
<script type="text/javascript">
    window.attachEvent('onmessage', function (e) {
        if (e.domain == 'example.com') {
            if (e.data == 'Hello World') {
                e.source.postMessage('Hello');
            } else {
                alert(e.data);
            }
        }
    });
</script>
```

This is the script code we use in order to get the message from the main page hosting the iframe. We attach an event listener to the page's onmessage event. The argument itself has all the data we need. We can check the source domain by the domain property and simply get the data from the data property. Finally if you need to send a response to the main page you can use the source property and use the `postMessage` function again. In this case, the main page should have another listener attached to the main page's `onmessage` event.

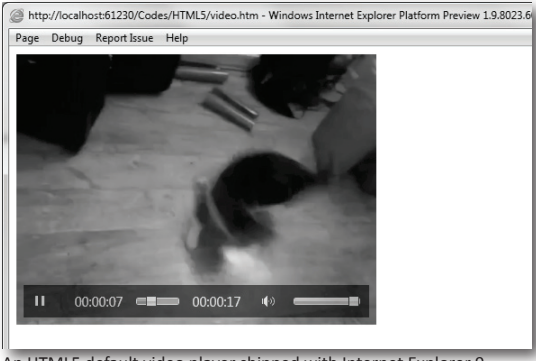
Video and Audio

The most popular and interesting feature included in HTML version 5 is the video and audio playback capabilities. The implementations are straight forward:

```
<video width="400"
height="300"
src="avideo.mp4"
poster="images/placeholder.jpg"
autoplay controls loop>
```

```
This content appears
if the video tag or the codec is not supported.
</video>
```

Above is a sample code playing a video file named "avideo.mp4." The `poster` property specifies an image which will be shown during the preloading of the video file. The autoplay property specifies if the video will be played instantly after it gets loaded or not. Additionally, the `controls` property enables a set of playback controls visible to the user. Finally, the `loop` property will let the player loop endlessly during playback.



An HTML5 default video player shipped with Internet Explorer 9.

If the client browser does not support HTML5 video playback, the inner text content of the video element will be rendered and the text inside will be visible to the user that tells it to install a new version of the browser, or get to the appropriate codec.

Customizing Video Playback Experience

It is possible to use the embedded controls inside the browser or create your own controls from scratch. The example below illustrates how to implement your own control set and you will discover some of the new capabilities in the video element as well.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        function Play() {
            var Video = document.
getElementById("myvideo");
            Video.play();
            RefreshInfo();
        }
        function RefreshInfo() {
            var Video = document.
getElementById("myvideo");
            document.getElementById("InfoContent").
innerHTML = Video.currentTime + " / " + Video.dura-
tion + " seconds";
            setTimeout("RefreshInfo()", 1000);
        }
        function SpeedUp() {
            var Video = document.
getElementById("myvideo");
            Video.playbackRate = 2;
        }
    </script>
</head>
<body>
    <video id="myvideo" width="400"
    height="300"
    src="avideo.mp4"
```

```
    poster="images/placeholder.jpg">
    This content appears
    if the video tag or the codec is not supported.
    </video>
    <div id="InfoContent"></div>
    <input id="Button1" type="button" value="Play"
onclick="Play();" />
    <input id="Button2" type="button" value="Speed
Up" onclick="SpeedUp();" />
</body>
</html>
```

The example has a `RefreshInfo` function getting the natural length of the video file and the current position. The position and the length report into a div element to make it visible to the user like a regular playback experience.

Additionally we have another button titled "Speed Up". This is especially very valuable for podcast content. We not only are able to rewind or forward the video, but we are also able to change the speed of the playback as well. The video element in HTML5 provides a rich set of methods and properties to use. You can find a list of these at ***http://msdn.microsoft.com/en-us/library/ff975073.aspx***.

Picking The Right Codecs

Internet Explorer 9 supports multiple video element sources. For example, you can have a video file encoded with two different codes and list them by priority order inside the video element.

```
<video width="400" height="300"
    poster="frame.png" autoplay
    controls loop>
    <source src="video.ogv" type='video/ogg;
codecs="theora, vorbis"'>
```

```
    <source src="video.mp4" type='video/mp4;
codecs="avc1.42E01E, mp4a.40.2"'>
</video>
```

Internet Explorer 9 will go through the list from top to the bottom and pick the first running video file for playback.

The Audio Element

Running an audio file inside the browser is as easy as the video player implementation. It requires the same code as the video file playback except the element name is not "video" anymore, it is "audio."

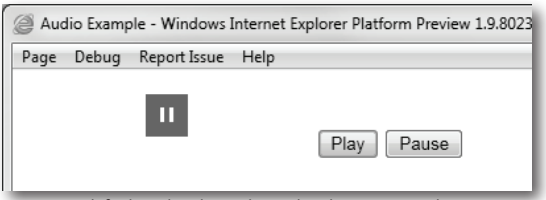
```
<html>
  <head>
    <title>Audio Example</title>
     <script type="text/javascript">

          function playAudio() {
              var a = document.getElementById('a1');
              a.play();
          }

          function pauseAudio() {
              var a = document.getElementById('a1');
              a.pause();
          }
     </script>
</head>

<body>
  <audio id="a1" style="width:25%" controls="true"
src="images/audio.mp3" >Not supported</audio>
<button onclick="playAudio();">Play</button> <but-
ton onclick="pauseAudio();">Pause</button>
</body>
</html>
```

Above you can find a full example of an audio playback in Internet Explorer 9 with HTML5.



An HTML5 default audio player shipped with Internet Explorer 9.

TIP Internet Explorer 9 supports MP4 container, H.264 video, all profiles audio in AAC or MP3.

Canvas

The canvas element is the origin of all the amazing HTML5 sites you saw up until now. Canvas is an entry point to dynamic drawings and animations in HTML5. In this part we will start creating some very simple drawings and see how we can programmatically push content to the GPU. Afterwards we will create an animation with APIs.

Creating A Canvas From Scratch!

In order to create a canvas we just need one line of code embedding the canvas element inside the DOM.

```
<canvas width="300" height="225"></canvas>
```

In the code above you see a canvas with a height of `225` pixels and width of `300` pixels. This will be our area of drawing. The area that the canvas control will be rendered and where we can start using additional functions and APIs to push visual

content to the screen.

Of course in order to be able to access our canvas element the element should have a name.

```
<canvas id="a" width="300" height="225"></canvas>
```

Before starting pushing some visual elements on the canvas we need to get a drawing context. The drawing context will provide us all the functions we need to draw shapes or text.

```
function draw() {
     var a_canvas = document.getElementById("a");
     var a_context = a_canvas.getContext("2d");
     a_context.fillRect(60, 35, 150, 100);
}
```

The above code is just a JavaScript function we will be calling to draw a rectangle on our canvas. First we get our canvas element with `document.getElementById` and second we request a context by calling the method getContext. The `getContext` method requires a parameter which in our case will be "2d". This is actually the only choice we have at the moment. We can't create a 3D canvas but with the hope that it will happen in the future it's nice to have the parametric mechanism already implemented. Finally within our context we call a method named `fillRect`. This method eventually draws rectangles according to the parameters you specify. The first two parameters of `fillRect` is the X, Y coordinates of the rectangle and the second two parameters are the height and width of the rectangle.

Note that the upper left corner of the canvas is the origin point and the coordinates we assign to the `fillRect` should be relative to the origin point. Besides `fillRect` you can use the `strokeRect` function as well in order to draw an empty rectangle with only borders.

Drawing in a Canvas

Let's start with a little bit complex example. In the code below we will be using two new methods we didn't see till now. The first one is the `moveTo` method. The `moveTo` method moves our context in the canvas. It's like moving your pencil on a paper but drawing nothing. In the for loop below we move our canvas slowly to the right with each cycle. Second we have the `lineTo` method which draws a line from the current position of the pencil to another point.

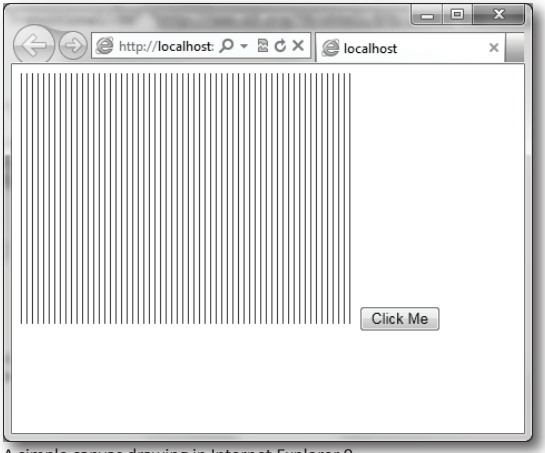
```
for (var x = 0.5; x < 300; x += 5) {
     context.moveTo(x, 0);
     context.lineTo(x, 375);
}
```

As you see we are actually drawing vertical lines with `5` pixel margins to each other. When you run this code you will see nothing on screen. The problem is we didn't specify what kind of pencil we have.

```
context.strokeStyle = "#004400";
context.stroke();
```

Here we go with a new `strokeStyle` for our current context and finally we call the stroke method which will draw everything we had in our buffer. Below is the final code.

```
function draw() {
      var a_canvas = document.getElementById("a");
      var context = a_canvas.getContext("2d");
      for (var x = 0.5; x < 300; x += 5) {
           context.moveTo(x, 0);
           context.lineTo(x, 375);
       }
       context.strokeStyle = "#004400";
       context.stroke();
       }
```



A simple canvas drawing in Internet Explorer 9

On top of drawing you can embed some text as well. To paint a text on the canvas we need to use the method called `fillText`. But before that we need to specify what font and size we will be using. In order to assign a style to the text we use the font property of the context. Finally the `fillText` method gets a text and the relative x, y coordinates where the text will be rendered and pushes the visual content on the canvas.

```
function draw() {
      var a_canvas = document.getElementById("a");
      var context = a_canvas.getContext("2d");
      context.font = "bold 14px sans-serif";
      context.fillText("Sample Text", 100, 50);
}
```

We now know how to draw lines and text on the screen but what about some more visually appealing gradients? What kind of fill styles we can use?

```
var gradient = context.createLinearGradient(0, 0, 300, 0);
```

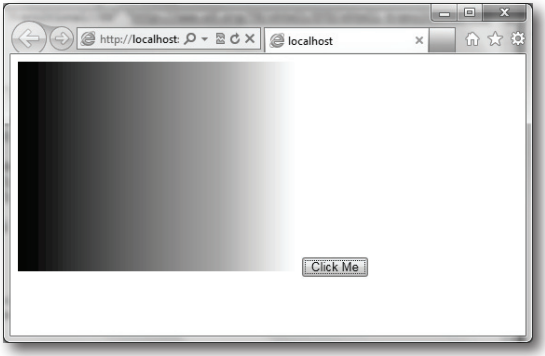
The code above creates a gradient brush which will go from `0,0` to `300,0` `(x,y)` So the parameters of `createLinearGradient` is the x and y coordinates of two different points where the gradient will be drawn. Of course it is not enough to define the gradient like this, we need gradient stops, colors and the positions of gradient stops and finally use this brush to draw something.

```
gradient.addColorStop(0, "black");
gradient.addColorStop(1, "white");
```

Here we add two different `GradientStops` with their positions throughout the gradient and their color. The first parameter can be between `0` and `1` which will define the position of the point between the beginning and the end of the gradient and the second parameter is simple the color which will be used at the gradient stop.

```
context.fillStyle = gradient;
context.fillRect(0, 0, 300, 225);
```

Finally it's time to draw something, maybe a rectangle to the screen so we can use our gradient brush. First we assign our brush to the `fillStyle` property of our context and then use the `fillRect` method to draw a rectangle on the screen.



Our first gradient brush used to draw a rectangle in a canvas.

Animations

In order to create animations with a canvas we need to use the old cartoon animations tactics. With all the example we had in this part of the book now we know we can draw whatever we want on a canvas. What if we run this "drawing" process over and over every 25 millisecond? Wouldn't that look like we are running a cartoon animation? Yes, and this is what we will do now.

Think about the last gradient we draw. We will enhance it a little bit and add one more gradient stop. After having three gradient stops we will start animate or change the position of the second gradient stop to have a simple gradient animation.

```
var Position = 0;
var Change = 0.01;
```

First we define two variables outside our drawing function.

These variables will keep the current position of the second gradation stop and the change we need to reflect to the position. At the beginning the position will start from `0` and we will start adding `0.01` on top of that until we get to `1` which is the end of the gradient locations. When we reach the position `1` we will change direction and change the variable called change to `-0.01` so we start subtracting `0.01` from `1` and go back to the beginning. This whole process will be repeated forever with a siteTimeout function.

```
var Position = 0;
        var Change = 0.01;

        function draw() {
            Position += Change;

            if (Position >= 1) {
                Position = 1;
                Change = -0.01;
            }
            else if (Position <= 0) {
                Position = 0;
                Change = 0.01;
            }

            var a_canvas = document.getElementById("a");
            var context = a_canvas.getContext("2d");

            var gradient = context.createLinearGradient(0, 0, 300, 0);
            gradient.addColorStop(0, "black");
            gradient.addColorStop(Position, "white");
            gradient.addColorStop(1, "black");
            context.fillStyle = gradient;
            context.fillRect(0, 0, 300, 225);
            setTimeout("draw()", 25);
        }
```

The code above is the final code running the animation. For every 25 milliseconds the draw function will be called thanks to the setTimout which exists at the end of the function. On each run the function will be drawing a slightly different gradient creating an illusion of an animation.

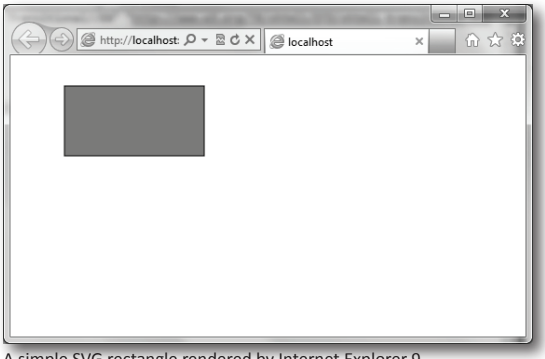
SVG

Support for Scalable Vector Graphics (SVG) was one of the most requested features and is now implement in Internet Explorer 9. SVG is a way to define vector object with XML and get them rendered client side.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <svg version="1.1" xmlns="http://www.
w3.org/2000/svg">
        <rect fill="red"
                stroke="black"
                width="150"
                height="75"
                x="50"
                y="25" />
        </svg>
</body>
</html>
```

In order to have an SVG area in a document you can simple create an SVG tag, specify the XML namespace and finally the version number 1.1 which is the version supported by IE9. Moving on you can define SVG objects in the SVG area with the regular XML syntax. The example above will render a rectangle of 150*75 pixel size at the coordinate of 50,25 filled

with a red solid color brush.



A simple SVG rectangle rendered by Internet Explorer 9.

As we will not go deep into the details of the SVG format itself one of the important functionalities we shouldn't be missing is the interactivity abilities of SVG objects.

```
<circle id="thecircle" cx="100" cy="100" r="50" fill="red"
onmousedown="blue()" onmouseup="red()"/> </svg>
```

Above you can see a simple SVG object, a circle having two event listeners attached to the `onmousedown` and `onmouseup` events. These event listeners will be fired when the user clicks the circle.

```
var blue = function () {
  redcircle = document.getElementById('thecircle');
  redcircle.setAttribute("fill", "blue");
}
var red = function () {
   var redcircle = document.getElementById('thecircle');
   redcircle.setAttribute("fill", "red");
}
```

In our JavaScript we are pulling the SVG object by its name and setting the fill attribute to different values. In this case different colors. Finally it is important to know how we can integrate all this together.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <svg xmlns="http://www.w3.org/2000/svg"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        width="600" height="600">  <script type="text/
ecmascript">
        <![CDATA[
            var blue = function () {
                var redcircle = document.
getElementById('thecircle');
                redcircle.setAttribute("fill", "blue");
            }
            var red = function () {
                var redcircle = document.
getElementById('thecircle');
                redcircle.setAttribute("fill", "red");
            }
        ]]>
        </script>
        <circle id="thecircle" cx="100" cy="100"
r="50" fill="red"
        onmousedown="blue()" onmouseup="red()"/> </
svg>
</body>
</html>
```

3

Windows 7 Integration

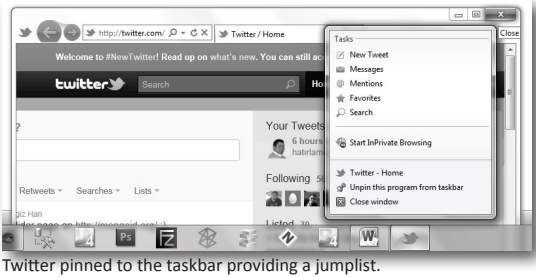
In This Chapter

It is important to give your visitors the feeling of a regular windows application in a web site. Usually end-users are used to slow web sites and the lack of user experience comparing to locally running windows applications. The abilities of a web site and windows application are pretty much different right?

Internet Explorer 9 is helping us provide the same desktop feeling on the web with a wide variety of Windows 7 integration features. Let's go one by one and dig into the details of the technical implementations.

Pinned Sites

Imagine your web site being a windows application and pinned to the taskbar as a favorite one. This is not a dream. With Internet Explorer 9 it is possible to drag and drop a tab hosting a web site inside IE and pin it to the taskbar of the operating system. After this pinning process is done the web site will have additional abilities like showing notifications, providing jumplists and thumbnail toolbars like every windows application can do.



Twitter pinned to the taskbar providing a jumplist.

Before going into the additional details let's first focus what kind of customizations we can do after our web site has been pinned and elevated to a desktop application mode.

```
<head>
    <title></title>
```

```
    <meta name="msapplication-starturl"
content="http://www.bing.com/" />
    <meta name="msapplication-window" content="widt
h=1024;height=600"/>
    <meta name="msapplication-tooltip"
content="Just a tooltip!" />
    <meta name="application-name" content="Sample
App" />
</head>
```

Above you see a list of items you can define inside the header of your web site as meta attributes. These attributes can be used by Internet Explorer 9 on Windows 7. The first msapplication-starturl defines the first web page which will be opened when the user pins the web site to the taskbar. The second `msapplication-window` setting can define the size of the window hosting your web site when it is launched from the taskbar. Finally the `tooltip` and the `application-name` are just texts shown to the user on top of the title of the window hosting the application.

Jumplists On The Web

In order to provide jumplists to your visitors the first choice is to define a static list of commands or links in the header of your web page.

```
<head>
     <title></title>
     <meta name="msapplication-task"
      content="name=Test Action;action-uri=http://
      www.bing.com;icon-uri=http://localhost:59832/
      win7/favicon.ico" />
</head>
```

The three different parameters you can set are the name of the action, the target web sites URL and the icon URL to be shown in the jumplist. The crucial part is the target action

URL. This URL can be targeting a remote web site or targeting your own web site with a specific parameter which can lead your visitors directly to a different part of your web site.

In case you need dynamically update the jumplist on-the-fly you can access the API with JavaScript as well.

```
window.onload = function () {
    try {
      if (window.external.msIsSiteMode()) {
       window.external.msSiteModeCreateJumpList("My
Site");
       window.external.msSiteModeAddJumpListItem(
                "Task2", "/users/44324847/task2",
"img/icon.ico");
        }
    }
    catch (ex) {
    // Failed!
    }
}
```

The code above goes through the `window.external` API which is only available on IE9 and check if the web site is running in a pinned mode (SiteMode) with a method called `msIsSiteMode`. After the check is done the code is free to call the `msSiteModeCreateJumpList` method and start creating a new jumplist. Finally the `msSiteModeAddJumpListItem` method can add multiple jumplist items to the jumplist.

Additionally you can use the `msSiteModeClearJumpList();` method to clear the jumplist items and msSiteModeShowJumplist method to update the list.

Taskbar Notifications

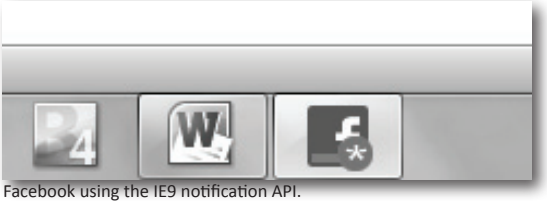
Facebook has its own notification mechanism showing you

notifications in the web site and sending you e-mails. When you have facebook open but minimized into the taskbar there is no way you can see the notifications happening in the web site. With the help of the taskbar notifications in Windows 7 we can notify the user even the web site or application is minimized to the taskbar but it is a living one.

```
function sendNotification() {

   try {
      if (window.external.msIsSiteMode()) {
          window.external.msSiteModeSetIconOverlay(
"http://localhost:59832/win7/favicon.ico",
         "Notification down here!");
        }
        }
        catch (e) {
        // Failed.
        }
}
```

The notification API is available through `window.external` namespace. It is always a good practice to check if the web site is running on `SiteMode` first and then go head and implement the additional functionality. In the code above the `msSiteModeSetIconOverlay` method is called to show a notification in the taskbar. The notification will stay there until you clear it with the `msSiteModeClearIconOverlay()` method.

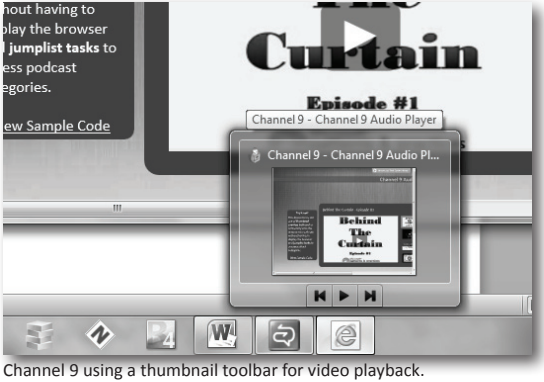


Facebook using the IE9 notification API.

Thumbnail Toolbars

Thumbnail toolbars are specifically known thanks to Windows Media Player. When you hover the taskbar icon of a running Media Player and get the thumbnail preview of the application you will see additional buttons stick to the preview allowing users play, pause, like and so on different songs they have in their library.

Thumbnail toolbars is a functionality you can implement to your web site as well.



Channel 9 using a thumbnail toolbar for video playback.

In order to create our own Thumbnail toolbars first we need to define the toolbar buttons.

```
var btnPrev = window.external.msSiteModeAddThumbBar
Button('Images/prev.ico', 'Previous');
var btnPlayPause = window.external.msSiteModeAddThu
mbBarButton('Images/play.ico', 'Play');
var btnNext = window.external.msSiteModeAddThumbBar
```

```
Button('Images/next.ico', 'Next');
```

The code above defines three different buttons with different icons and tooltip texts. The method we use to add buttons to the toolbar is called `msSiteModeAddThumbBarButton` . The second step is to push the toolbar to the operating system so it will be visible to the user. This is done through the msSite-ModeShowThumbBar function.

```
window.external.msSiteModeShowThumbBar();
```

Finally it's time to attach some event listeners so we can detect when buttons get hit.

```
if (document.addEventListener) {
    document.addEventListener('msthumbnailclick',
onButtonClicked, false);
}
else if (document.attachEvent) {
         document.attachEvent('onmsthumbnailclick',
onButtonClicked);
}
```

This is the place where we attach our event listener. We don't need to attach event listeners to each button. All the buttons will fire a local DOM evet called 'onmsthumbnailclick'. It is enough to start listening to this event and pick which button is clicked.

```
function onButtonClicked(btn) {
     switch (btn.buttonID) {
        case btnPrev:
              alert('Prev Hit');
              break;
        case btnPlayPause:
              alert('Play Hit');
              break;
        case btnNext:
              alert('Next Hit');
```

```
                break;
    }
}
```

Finally our event listener code gets a reference of the button clicked so we can check which button is hit. After we know the button clicked we can trigger any kind of client side mechanism with JavaScript.

4

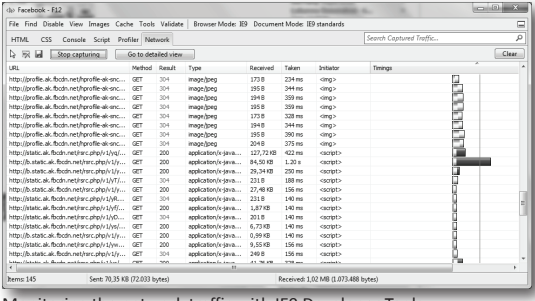
Developer Tools (F12)

In This Chapter

As developers a long time we were looking for something like the "developer tools" embedded inside the browser and that happened with Internet Explorer 8. With the new Internet Explorer 9 we have some advancement and new features added to the pack you can reach when you hit the **F12** key from your keyboard.

Network Tab

Internet Explorer 9 has a new network tab in the developer tools which will let you inspect the network for different testing or performance tuning purposes. You can capture HTTP and HTTPS network traffic, display and save the contents of captured requests and responses and display extra details about captured data, such as cookies, sizes, timings, and cache information.

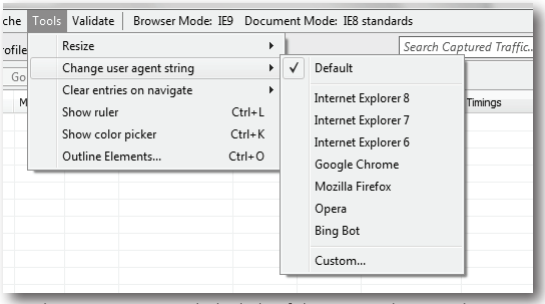


Monitoring the network traffic with IE9 Developer Tools.

User-Agent Switcher Tool

Sometimes it is crucial to be able to change the user-agent string easily for testing purposes. With the help of the new feature called “user agent switcher” in IE9 developer tools you can modify the UA string that identifies the browser type and version to the web server. The chosen UA string will be sent across the network as a header in every request. To change the UA string of the browser, on the developer tools click the “Tools” menu, go to “Change user agent string”, and then cho-

ose the browser whose UA string you want to send.



Switching user agents with the help of the IE9 Developer Tools.

The End!

The web is slowly getting what it really needs thanks to HTML5 and IE9 as being the hardware accelerated browser allowing us to build high performance and user experience rich web applications or web sites. Combined with all the features of HTML5 we introduced in this book I'm sure you already have a lot of different ideas of implementation popped up in your mind already. I will not keep you here anymore, now it's time to get your hands on the work!

Let me know of your HTML5 projects and the amazingly magical stuff you do to share the excitement all together. You can always send me an e-mail by ***daron@yondem.com***! Keep in touch!

NOTES

NOTES

NOTES

Welcome to Windows Internet Explorer 9 developer and designer book! Both as a developer or as a designer I'm sure you are eager to know what's new or changing with the new version of Internet Explorer. Is it going better? Or worse? Will that make my life easier, or maybe make the products I build better? The book will not only give you the answer of the generic question "What's in it for me?" moreover we will have the chance to dig into the technical details and real world implementations as well.

Windows® Internet Explorer 9

- CSS3
- 2D Transforms
- New Ways of Defining Colors
- Rgba: Red, Green, Blue and Alpha
- HSL & HSLA
- Using Custom Fonts
- CSS3 Namespaces
- CSS3 Selectors
- HTML5
- Dom Storage
- AJAX Navigation
- Cross Document Messaging
- Pinned Sites
- Windows 7 Integration
- Pinned Sites
- Network Tab
- User-Agent Switcher Tool
- Developer Tools (F12)
- Network Tab
- User-Agent Switcher Tool
- Video and Audio
- Canvas
- SVG
- Examples

Microsoft® Most Valuable Professional

Microsoft® Regional Director PROGRAM

Daron Yöndem is the founder of Deveload Software – a UX company based in Turkey. He is a Microsoft Regional Director with a Gold Global Impact Award in year 2009 and 2010. Daron is an international speaker leading the INETA MEA, he is a Silverlight MVP and author of two ASP.NET AJAX books and an HTML5 book. He is passionate about UX and can host sessions everywhere anytime. He has more than 180+ sessions hosted in year 2010 including a full night of free Silverlight community training called SilverNight! You can follow his thoughts at daron.yondem.com

daron.yondem.com

dikeyeksen
www.dikeyeksen.com

ISBN 605-61677-6-8
9 786056 167768