

References



Table of Contents

S#	Session	Page #
1.	Session 1: Introduction to the Web	
	• Top 10 Reasons To Use HTML5 Right Now	4
	• Important Web Standards Resources for Developers	11
2.	Session 2: Introduction to HTML5	
	• SEO Tutorial	21
3.	Session 3: Formatting Text Using Tags	
	• Formatting Text in HTML	22
	• Easy to Read Fonts	24
4.	Session 4: Creating Hyperlinks and Anchors	
	• Images as Hyperlinks	27
	• Change color of links	30
5.	Session 5: Introduction to CSS3	
	• Cascading Style Sheets	31
	• CSS3 Border	37
6.	Session 6: Formatting Using Style Sheets	
	• Border Image	46
7.	Session 7: Displaying Graphics and CSS3 Animation	
	• Making a move with CSS3	51
8.	Session 8: Creating Navigational Aids and Division-based Layout	
	• Structural Semantic Elements in HTML5	59
	• CSS3 Multicolumn layout	62
9.	Session 9: Creating Tables	
	• Tables	69
10.	Session 10: HTML Forms	
	• Multiple Submit buttons	70
	• Customize HTML5	77
	• Fun With HTML5	79
11.	Session 11: HTML5 Audio and Video	
	• Embedding video and audio	87
	• Jaraoke – An HTML5 <audio> Tag Demo	88
12.	Session 12: Introduction to JavaScript	

S#	Session	Page #
	• Simple Things to make jQuery	90
13.	Session 13: Operators and Statements	
	• JavaScript Regular Expression Tester	93
	• Email Validation Using Regular Expressions	97
14.	Session 14: Loops and Arrays	
	• Comparing JavaScript loops performance	101
	• Going Loopy with JavaScript	103
	• JavaScript array	107
15.	Session 15: Functions and Objects	
	• JavaScript examples using the DOM	123
16.	Session 16: Building a Mobile Web Application	
	• Mobile Application Development	127
	• Mobile Development and Compiler HTML5	132
17.	Session 17: Canvas an JavaScript	
	• Creating a Drawing App with HTML5	134
18.	Session 18: HTML5 Web Storage	
	• HTML5 Local Storage	147
	• DOM Storage	155
	• Client-side Storage	165
19.	Session 19: HTML5 Geolocation and APIs	
	• Geolocation-enabled Opera build	174
	• Geolocation based on IP address	176
	• Drag and Drop	179

Session 1: Introduction to the Web

Top 10 Reasons To Use HTML5 Right Now

Source	http://tympanus.net/codrops/2011/11/24/top-10-reasons-to-use-html5-right-now/
Date of Retrieval	27/7/2012

So you're still not using HTML5, huh? I guess you probably have your reasons; it's not fully adopted yet, it doesn't work in IE, you don't like users, you're out of touch or you are just passionately in love with writing strict XHTML code. HTML5 is the revolution that the Web needed and the fact is, it is the future whether you like it or not — suck it up and deal. HTML5 isn't hard to use or understand and even though it's not fully adopted yet, there are still plenty of reasons to start using it **right now** — *like right after you get done reading this article.*

There are lots of articles touting the use of HTML5 and praising the benefits of it, yes this is another one of those. With all these articles, with Apple pushing it, with Adobe building new dev products around it, and with so many Web sites devoted to it, I still talk to fellow designers and developers who haven't or won't adopt it for a variety of reasons. I think the **main problem** is, it still seems like a mysterious creature to many. To many it feels more like the jet pack or the flying car — an awesome idea that is fun to think about but still not practical in its use. Wrong, the reality is that it is extremely practical right now! It's not the latest Mercedes concept car being towed around from car show to car show, it's a reality and it's not going anywhere.

In order to further **demystify HTML5** and help these knuckle dragging designers and developers to jump on the bandwagon I've put together a top ten list of reasons why we should all be using HTML5 right now. For those that currently use HTML5 this list may not be anything new or ground breaking, but hopefully it will inspire you to share the benefits of HTML5 with others in the community. We'll do this Letterman countdown style (minus the celebrity presenter) and start with number ten – *accessibility*.

10 – ACCESSIBILITY



HTML5 makes creating accessible sites easier for two main reasons: semantics and ARIA. The new (some currently available) HTML headings like `<header>`, `<footer>`, `<nav>`, `<section>`, `<aside>`, etc. allow screen readers to easily access content. Before, your screen readers had no way to determine what a given `<div>` was even if you assigned it an ID or Class. With new semantic tags screen readers can better examine the HTML document and create a better experience for those who use them.

ARIA is a W3C spec that is mainly used to assign specific “roles” to elements in an HTML document – essentially creating important landmarks on the page: header, footer, navigation or article, via role attributes. This has been well overlooked and widely under-used mostly due to the fact that it wasn’t valid, however, HTML5 will validate these attributes. Also, HTML5 will have built in roles that can’t be over-ridden making assigning roles a no brainer. For a more in depth discussion on HTML5 and ARIA please visit the WAI.

9 – VIDEO AND AUDIO SUPPORT

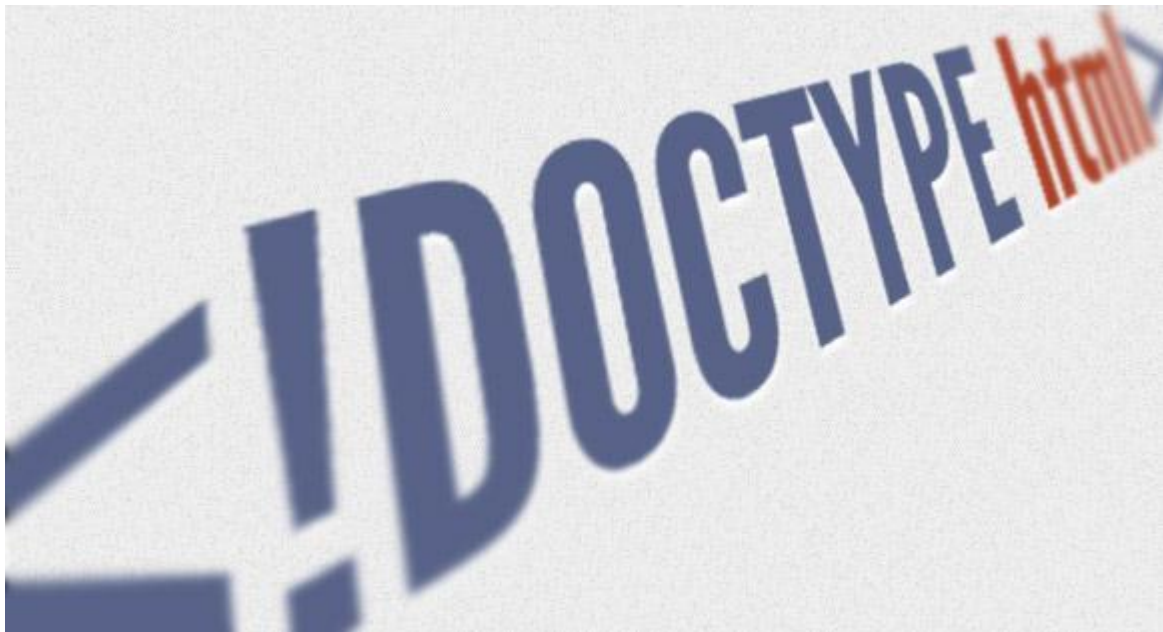
Forget about Flash Player and other third party media players, make your videos and audio truly accessible with the new HTML5 `<video>` and `<audio>` tags. Getting your media to play correctly has always been pretty much a nightmare, you had to use the `<embed>` and `<object>` tags and assign a huge list of parameters just to get the thing visible and working correctly. Your media tags just become these nasty, huge chunks of confusing code segments. HTML5’s video and audio tags basically treat them as images; `<video src="url"/>`. But what about all those parameters like height, width and autoplay? No worries my good man, just define those attributes

in the tag just like any other HTML element: `<video src="url" width="640px" height="380px" autoplay/>`.

It's actually that dead simple, however because old evil browsers out there don't like our HTML5 friend, you'll need to add a little bit more code to get them working correctly... but this code isn't nearly as gnarly and messy as the `<object>` and `<embed>` tags:

```
<video poster="myvideo.jpg" controls>
  <source src="myvideo.m4v" type="video/mp4" />
  <source src="myvideo.ogg" type="video/ogg" />
  <embed src="/to/my/video/player"></embed>
</video>
```

8 – DOCTYPE



```
<!DOCTYPE html>
```

Yup that's it, that is the doctype, nothing more, nothing less. Pretty simple right? No more cutting and pasting some long unreadable line of code and no more dirty head tags filled with doctype attributes. You can simply and easily type it out and be happy. The really great thing about it though, beyond the simplicity, is that it works in every browser clear back to the dreaded IE6.

7 – CLEANER CODE

If you are passionate about simple, elegant, easy to read code then HTML5 is the beast for you. HTML5 allows you to write clear and descriptive code, semantic code that allows you to easily separate meaning from style and content. Consider this typical and simple header code with navigation:

```
<div id="header">
  <h1>Header Text</h1>
```

```
<div id="nav">
  <ul>
    <li><a href="#">Link</a></li>
    <li><a href="#">Link</a></li>
    <li><a href="#">Link</a></li>
  </ul>
</div>
</div>
```

So this code is pretty clean and simple? But with HTML5 you can clean this up even more and at the same time give your markup more meaning:

```
<header>
  <h1>Header Text</h1>
  <nav>
    <ul>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
      <li><a href="#">Link</a></li>
    </ul>
  </nav>
</header>
```

With HTML5 you can finally cure your “divitis” and “classitis” by using semantic and HTML headers to describe your content. Previously you would generally just use div’s for every block of content then drop an id or class on it to describe its content but with the new <section>, <article>, <header>, <footer>, <aside> and <nav> tags, HTML5 allows you to code your markup cleaner as well as keep your CSS better organized and happier.

6 – SMARTER STORAGE



One of the coolest things about HTML5 is the new local storage feature. It's a little bit of a cross between regular old cookies and a client-side database. It's better than cookies because it allows for storage across multiple windows, it has better security and performance and data will persist even after the browser is closed. Because it's essentially a client side data base you don't have to worry about the user deleting cookies and it is been adopted by all the popular browsers.

Local storage is great for many things, but it's one of HTML5 tools that are making Web apps possible without third party plugins. Being able to store data in the user's browser allows you to easily create those app features like: storing user information, the ability to cache data, and the ability to load the user's previous application state.

5 – BETTER INTERACTIONS

Awe, we all want better interactions, we all want a more dynamic Website that responds to the user and allows the user to enjoy/interact your content instead of just look at it. Enter `<canvas>`, the drawing HTML5 tag that allows you to do most (if not more) interactive and animated possibilities than the previous rich Internet application platforms like Flash.

Beyond `<canvas>`, HTML5 also comes with a slew of great APIs that allow you to build a better user experience and a beefier, more dynamic Web application — here's a quick list of native APIs:

- Drag and Drop (DnD)
- Offline storage database
- Browser history management
- document editing
- Timed media playback

4 – GAME DEVELOPMENT

Yup, that is correct, you can develop games using HTML5's <canvas> tag. HTML5 provides a great, mobile friendly way to develop fun, interactive games. If you've built Flash games before, you'll love building HTML5 games.

3 – LEGACY/CROSS BROWSER SUPPORT



Your modern, popular browsers all support HTML5 (Chrome, Firefox, Safari IE9 and Opera) and the HTML5 doctype was created so that all browsers, even the really old and annoying ones, er, IE6 can use it. But just because old browsers recognize the doctype that doesn't mean they can use all the new HTML5 tags and goodies. Fortunately, HTML5 is being built to make things easier and more cross browser friendly so in those older IE browsers that don't like the new tags we can just simply add a Javascript shiv that will allow them to use the new elements:

```
<!--[if lt IE 9]>
  <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

2 – MOBILE, MOBILE, MOBILE

Call it a hunch, but I think mobile technology is becoming more popular these days. I know, that is a pretty crazy assumption and some of you are probably thinking — mobile is just a fad... right. Mobile devices are taking over the world. The adoption of mobile devices continues to grow very rapidly and this means that more and more users will be using their mobile browsers to view your Web site or application. HTML5 is the most mobile ready tool for developing mobile sites and apps. With Adobe announcing the death of mobile Flash, you will now count on HTML5 to do your mobile Web application development.

Mobile browsers have fully adopted HTML5 so creating mobile ready projects is as easy as designing and constructing for their smaller touch screen displays — hence the popularity of Responsive Design. There are some great meta tags that also allow you to optimize for mobile:

- Viewport: allows you to define viewport widths and zoom settings
- Full screen browsing: IOS specific values that allow Apple devices to display in full screen mode
- Home Screen Icons: like favicons on desktop, these icons are used to add favorites to the home screen of an IOS and Android mobile device

1 – IT'S THE FUTURE, GET WITH IT!

The number one reason why you should start using HTML5 today is this: it's the future, start using it now so you don't get left behind. HTML5 is not going anywhere and as more and more elements get adopted more and more companies will start to develop in HTML5. HTML5 is essentially just HTML, it's not scary, it's not anything you really need to figure out or relearn — if you're developing XHTML strict right now you are already developing in HTML5 so why not take full advantage of it's current capability?

You really don't have any excuses not to adopt HTML5 and begin your new love affair with it. Truly, the only real reason I prefer to use HTML5 is just to write cleaner code, all the other benefits and fun features I haven't even really jumped into yet, but that is the great thing about it, you can just start using it right now and not even change the way you design. So, start using it right now, whether you are just simplifying and making your markup more semantic OR you are gonna build some sick new mobile game that will take over the world — who knows, maybe you can start selling stuffed animal versions of your gaming characters too.

~~~ End of Article ~~~



## Important Web Standards Resources For Developers

|                           |                                                                                                                                                                           |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source</b>             | <a href="http://www.topdesignmag.com/important-web-standards-resources-for-developers/">http://www.topdesignmag.com/important-web-standards-resources-for-developers/</a> |
| <b>Date of Retrieval:</b> | 27/7/2012                                                                                                                                                                 |

We use standards on a daily basis, often without realizing it. When we buy a light bulb, for example, we know that if we buy a screw-fitting bulb, it will fit our light fitting when we get home. Standards ensure that the bulb we buy isn't "just a little too large" or "just a little too wide" to fit our light fitting.

Web standards pick up from the same principle. As browser manufacturers have moved toward embracing standards, the need to write a browser-specific markup has diminished. By using well-structured HTML to mark up content and CSS to control presentation, we should now be able to design one Web site, and it should display consistently across standards-compliant browsers, regardless of the operating systems.

Web standards save us time as designers and allow us to sleep at night, safe in the knowledge that our carefully crafted masterpiece is accessible regardless of who's viewing it on which browser and which platform.

If you'd like to learn about Web and accessibility standards then you are in the right place because the following sites will help you with everything you need from learning the foundation of Web standards and all the way to cross-browser compatibility, html 5 demos or css validators.


### Web standards and accessibility resources

World Wide Web Consortium (W3C)



The screenshot shows the W3C website. At the top, there's a blue header with the W3C logo. Below it, a navigation bar contains links for STANDARDS, PARTICIPATE, MEMBERSHIP, and ABOUT W3C. The main content area is divided into two columns. The left column, titled 'STANDARDS', lists various web standards with icons: Web Design and Applications, Web Architecture, Semantic Web, XML Technology, Web of Services, Web of Devices, and Browsers and Authoring Tools. The right column features a section titled 'W3C Workshop: Identity in the Browser' dated 21 March 2011. The text in this section discusses the importance of identity authentication and authorization technologies, mentioning a workshop organized by the Mozilla Foundation on May 24-25, 2011. It states that participants will investigate strategies to facilitate improved identity authentication and authorization technologies across the Web, and that there is no fee to participate. It also mentions that participants are required to submit additional participation requirements and provides a link to learn more about the workshop.

## Web Standards Project



**The Web Standards Project is a grassroots coalition fighting for standards which ensure simple, affordable access to web technologies for all.**

Search WaSP

[About](#) [Learn](#) [Action](#) [Buzz](#) [Press](#)

### Recent Buzz

**[The Sherpas are Here](#)**  
By [Aaron Gustafson](#) | March 13th, 2011

After many months of hard work, we're excited to announce the launch of Web Standards Sherpa.

Today, I am very proud to announce the launch of our newest endeavor: [Web Standard Sherpa](#). This project has been the better part of a year in the making and we're really excited to see it finally launch.

Web Standards Sherpa came about because we wanted to create a repository of best practices information while, at the same time, providing mentorship opportunities for practicing web professionals. With those goals in mind, we began to throw around ideas of what that could look like and we realized a pseudo-critique site could fit that bill perfectly. We say "pseudo" because the reviews we'll be posting on Web Standards Sherpa are not traditional critiques, but rather focused reviews of a particular aspect of a site.

### Current WaSP Projects

**[InterAct: Web Standards Curriculum](#)**  
InterAct is a living, open curriculum based upon web standards and best practices, designed to teach students the skills of the web professional.

**[InterAct Learning Tracks](#)**  
The InterAct curriculum framework has 6 learning tracks. Adapt and reuse our resources. Contribute your own content and ideas.

**[Small Business](#)**  
WaSP's Small Business effort and, thanks to progress.

By [Aaron Gustafson](#)

## Web Accessibility in Mind



[Products](#) [Services](#) [Articles](#) [Resources](#) [Community](#)

Expanding the web's potential for people with disabilities

**Getting Started**  
[Introduction to web accessibility](#)

**From the Blog**  
The Web Accessibility Game Plan  
March 19, 2011

## Web Accessibility Training

Join WebAIM for two days intensive, hands-on training  
**» Register today**  
Logan, Utah  
May 25-26, 2011

| Training               | Evaluation                  | Design                      |
|------------------------|-----------------------------|-----------------------------|
| Accessibility Training | Accessibility Certification | Design & Development        |
| Technical Assistance   | Monitoring & Reporting      | Accessibility International |

| Current Features     | Popular Resources           |
|----------------------|-----------------------------|
| Screen Reader Survey | Web Accessibility Checklist |

## Tutorials, references, statistics, and forums

### W3Schools



**w3schools.com**  
the world's largest web development site

**educate yourself!**  
beginners and experts

Search w3schools.com

**HTML Tutorials**

- Learn HTML
- Learn HTML5
- Learn XHTML
- Learn CSS
- Learn CSS3
- Learn TCP/IP

**Browser Scripting**

- Learn JavaScript
- Learn HTML DOM
- Learn DHTML
- Learn VBScript
- Learn AJAX
- Learn jQuery
- Learn E4X

## Learn to Create Websites

At w3schools.com you will learn how to make a website. We offer free tutorials in all web development technologies. Select a tutorial from the menu to the left.


**Make your own Website »**

» Tutorials » Try it Yourself » References

**WEB REFERENCES**

- HTML 4.01
- HTML5
- XHTML
- XML DOM
- JavaScript
- HTML DOM
- jQuery
- HTML Colors
- CSS
- CSS3
- PHP
- XSLT
- XPath
- XSL-FO
- Color Picker

### JavaScript Kit



**JAVASCRIPT KIT**

Home Free JavaScripts Tutorials ▼ References ▼ Applets Coding Forums Freewarejava

**FEATURES**

["Learning jQuery \(Packt Publishing\)" book review](#)

We review "Learning jQuery", a book aimed at introducing new comers to the jQuery framework.

**FEATURES**

[CSS Vertical List Menu](#)

This is a lean CSS vertical menu with support for 1 level of nested lists, displayed as a "drop down".

**SCRIPTS/ TUTORIALS**

- [Free JavaScripts](#)
- [JavaScript Tutorials](#)
- [DHTML Tutorials](#)

## Main Content Areas:

- [Free JavaScripts](#)
- [JavaScript Tutorials](#)
- [DHTML Tutorials](#)
- [Web Design Tutorials](#)
- [JavaScript Reference](#)
- [DOM Reference](#)
- [IE Filters/ Transitions Reference](#)
- [Free Java Applets](#)

Welcome to JavaScript Kit, a comprehensive resource for JavaScript tutorials, scripts, and more.

### JavaScript Kit- What's New?

[Setting CSS3 properties using JavaScript](#)

With the numerous CSS vendor prefixes one has to contend with when it comes to defining CSS3 properties such as `-moz-box-shadow` or `-webkit-` more confusing. This tutorial looks at how to streamline the setting of CSS3 property values in JavaScript, by checking for and targeting on

[Rescue Text and TEXTAREA field values script](#)

One of the worst things that can happen when a user is entering data into an `INPUT type="text"` or `TEXTAREA` element is an accidental browser re uses HTML5's `sessionStorage` to store the text entered into these fields as the user types, and recalls them in an event of a page refresh or even

[Going beyond cookies- Using DOM sessionStorage and localStorage to persist larger amounts of info](#)



## The jQuery Project



The screenshot shows the jQuery project website. At the top, there is a navigation bar with links: jQuery, Plugins, UI, Meetups, Forum, Blog, About, and Donate. Below this is a secondary navigation bar with links: Download, Documentation, Tutorials, Bug Tracker, and Discussion. The main content area features the jQuery logo with the tagline "write less, do more." and a description: "jQuery is a new kind of JavaScript Library. jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript." Below the description are three checkmarks indicating features: "Lightweight Footprint", "CSS3 Compliant", and "Cross-browser". On the right side, there is a section titled "GRAB THE LATEST VERSION!" with a sub-section "CHOOSE YOUR COMPRESSION LEVEL:" containing two radio buttons: "PRODUCTION (29KB, Minified and Gzipped)" and "DEVELOPMENT (214KB, Uncompressed Code)". Below these is a large "Download (jQuery);" button with a download icon. At the bottom right of this section, it says "Current Release: v1.5.2".

## HTML5 Demos



The screenshot shows the "HTML 5 Demos and Examples" website. The main heading is "HTML 5 Demos and Examples". Below it is a paragraph: "HTML 5 experimentation and demos I've hacked together. Click on the browser support icon or the technology tag to filter the demos (the filter is an OR filter)." Below this is a large image showing a workshop with a presenter and an audience. To the right of the image is a text box titled "Want to learn all about JavaScript?" with the text: "Full day limited places workshops covering HTML5 APIs, Node and jQuery for designers at £200 running throughout May. Find out more and grab your ticket now." Below the image and text box is a section titled "Filter demos:" followed by a grid of technology tags: canvas, contenteditable, dataset, dnd, events, file-api, geolocation, history, manifest, offline, postMessage, sql-database, storage, video, websocket, and workers.

## HTML5 Reset



ABOUT V2 THE WORDPRESS THEME THE CODE DOWNLOAD CONTRIBUTE  
ACKNOWLEDGEMENTS

## Why?

Like a lot of developers, we start every HTML project with the same set of HTML and CSS files. We've been using these files for a long time and have progressively added bits and pieces to them as our own personal best practices have evolved.

Now that modern browsers are starting to support some of the really useful parts of HTML5 and CSS3, it's time for our best practices to catch up, and we thought we'd put our files out there for everyone to use. By no means do we see this as the One True Way to start every project, but we think it's a good starting place that anyone can make their own.

## Web Safe Font Tester



Font Tester

Home Help About Contact

## Web Safe Fonts Preview

Preview Text:

Font Size: medium 10 pt

The quick brown fox jumps over the lazy dog.  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz  
 0123456789

font-family: Arial, Helvetica, sans-serif;

**The quick brown fox jumps over the lazy dog.  
 ABCDEFGHIJKLMNOPQRSTUVWXYZ  
 abcdefghijklmnopqrstuvwxyz  
 0123456789**

## Web Style Guide

# Web Style Guide 3rd edition

by Patrick J. Lynch  
and Sarah Horton

[HOME](#)
[ABOUT US](#)
[Web Style Guide Online](#)

## Web Style Guide Online

### Contents

Foreword

Preface

Acknowledgements

1 Process

- The site development team
- Sidebar: Web teams
- Initial planning
- A list of reminders

book cover

### CHAPTERS

- 1 Process
- 2 Universal Usability
- 3 Information Architecture
- 4 Interface Design
- 5 Site Structure
- 6 Page Structure
- 7 Page Design
- 8 Typography
- 9 Editorial Style
- 10 Forms and App

## 960 Grid System



[Download](#) - Templates: Acorn, Fireworks, Flash, InDesign, GIMP, Inkscape, Illustrator, OmniGraffle, Photoshop, Visio, Exp Design. Also: PDF sketch sheets + CSS files. Repository at [GitHub](#).

# 960

## GRID SYSTEM

ADS BY FUSION

3 days, 28 talks, 4 workshops, speakers inc: Josh Clark, Jeff Veen, Ethan Marcotte.



[CUSTOM CSS GENERATOR](#)
[HTML LAYOUT GENERATOR](#)
[GRID OVERLAY BOOKMARK](#)

### Essence

The 960 Grid System is an effort to streamline web development workflow by providing commonly used dimensions, based on a width of 960 pixels. There are two variants: 12 and 16 columns, which can be used separately or in tandem. [Read more](#).

### Dimensions

The 12-column grid is divided into portions that are 60 pixels wide. The 16-column grid consists of 40 pixel increments. Each column has 10 pixels of margin on the left and right, which create 20 pixel wide gutters between columns. [View demo](#).

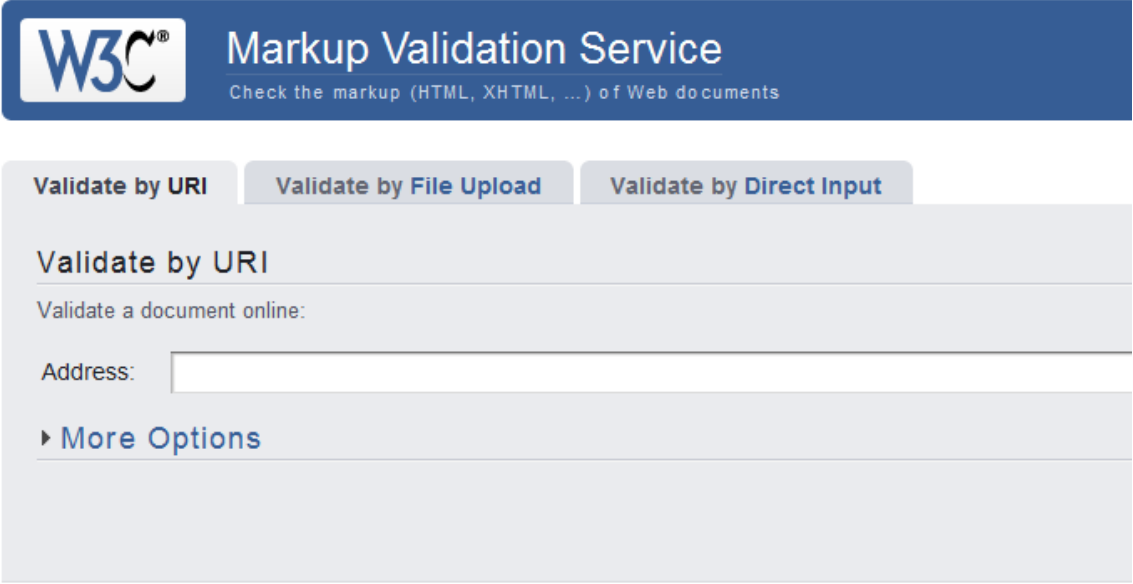
### Purpose

The premise of the system is ideally suited to rapid prototyping, but it would work equally well when integrated into a production environment. There are printable sketch sheets, design layouts, and a CSS file that have identical measurements.



## HTML/XHTML code validators

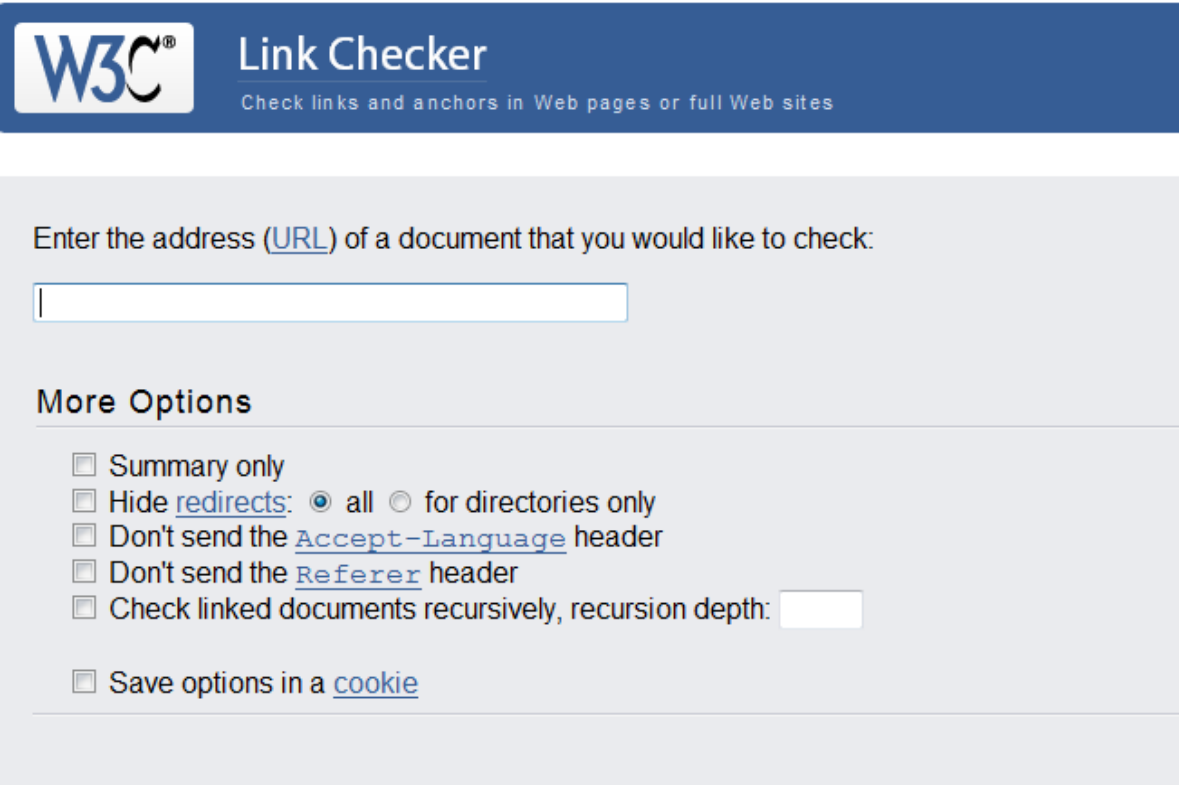
### W3C Markup Validator



The W3C Markup Validation Service interface features a dark blue header with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below the header are three tabs: "Validate by URI", "Validate by File Upload", and "Validate by Direct Input". The "Validate by URI" tab is active, showing a form with the label "Validate by URI" and the instruction "Validate a document online:". There is an "Address:" label followed by a text input field. Below the input field is a link "More Options".

This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc.

### W3C Link Checker



The W3C Link Checker interface has a dark blue header with the W3C logo and the text "Link Checker" and "Check links and anchors in Web pages or full Web sites". The main content area is light gray and contains the instruction "Enter the address ([URL](#)) of a document that you would like to check:" followed by a text input field. Below this is a section titled "More Options" with a list of checkboxes: "Summary only", "Hide [redirects](#):" (with radio buttons for "all" and "for directories only"), "Don't send the [Accept-Language](#) header", "Don't send the [Referer](#) header", "Check linked documents recursively, recursion depth:" (with a text input field), and "Save options in a [cookie](#)".

## W3C Log Validator


[\[QA Home\]](#)
[QA IG](#)
[Documents](#)
[Tools](#)
[Feedback](#)

### The Basics - what you should run on all your web pages

- [Unicorn](#) - W3C's Unified Validator
- The [Markup Validator](#) - Also known as the *HTML validator*, it helps check Web documents in formats like HTML and XHTML or MathML.
- The [Link Checker](#) - Checks anchors (hyperlinks) in a HTML/XHTML document. Useful to find broken links, etc.
- The [CSS Validator](#) - validates CSS stylesheets or documents using CSS stylesheets.

The above three can be used all-in-one by running the [Log Validator](#). Unlike the others, this tool helps improve the quality of a site, step by step, by finding the most popular documents that need to be fixed in priority. Learn more about this method in the [Validation Standards Switch](#) document.

Developing mobile-friendly content? The [mobileOK checker](#) is a one-stop service to check your Web site and improve its mobile friendliness.

### Specific Tools - for Specific Needs

## WDG HTML Validator



### WDG HTML Validator

Other languages: [français](#)

Enter the URL of an HTML document to validate. To quickly validate multiple URLs, try the [batch mode](#). Alternatively, you can [validate files on your computer](#) or you can [enter your HTML](#).

URL:


☒ Include warnings ☐ Show input ☐ Validate entire site ☐ Hide valid results

### About the WDG HTML Validator

- [Common validation problems](#)
- [Tips on using the validator](#)
- [How this validator differs from others](#)
- [Character encodings supported](#)
- [Debian and Red Hat packages](#)
- [Source code release](#)
- [Changes to this service](#)
- [Offline HTMLHelp.com Validator](#)
- [Why Validate?](#)

## CSS code validators

### W3C CSS Validator



# CSS Validation Service

Check Cascading Style Sheets (CSS) and (X)HTML documents with style sheets

By URI
By file upload
By direct input

## Validate by URI

Enter the URI of a document (HTML with CSS or CSS only) you would like validated:

Address:

► [More Options](#)

## Browser compatibility verification

### Browsershots

Browser Compatibility Test
Web Design Gallery
Icon Search Engine

Enter URL Here:



| Linux                                             | Windows                                           | Mac                                               | BSD                                                |
|---------------------------------------------------|---------------------------------------------------|---------------------------------------------------|----------------------------------------------------|
| <input checked="" type="checkbox"/> Chrome 10.0   | <input checked="" type="checkbox"/> Avant 11.7    | <input type="checkbox"/> Opera 9.50               | <input checked="" type="checkbox"/> Dillo 2.0      |
| <input checked="" type="checkbox"/> Chrome 11.0   | <input checked="" type="checkbox"/> Chrome 10.0   | <input type="checkbox"/> Opera 9.51               | <input checked="" type="checkbox"/> Epiphany 2.22  |
| <input checked="" type="checkbox"/> Chrome 12.0   | <input checked="" type="checkbox"/> Chrome 1.0    | <input type="checkbox"/> Opera 9.52               | <input checked="" type="checkbox"/> Firefox 3.0    |
| <input checked="" type="checkbox"/> Chrome 5.0    | <input checked="" type="checkbox"/> Chrome 2.0    | <input type="checkbox"/> Opera 9.60               | <input checked="" type="checkbox"/> Kazehakase 0.5 |
| <input checked="" type="checkbox"/> Chrome 9.0    | <input checked="" type="checkbox"/> Chrome 3.0    | <input type="checkbox"/> Opera 9.61               | <input checked="" type="checkbox"/> Konqueror 3.5  |
| <input checked="" type="checkbox"/> Dillo 0.8     | <input checked="" type="checkbox"/> Chrome 8.0    | <input type="checkbox"/> Opera 9.62               | <input checked="" type="checkbox"/> Opera 9.64     |
| <input checked="" type="checkbox"/> Dillo 2.2     | <input checked="" type="checkbox"/> Chrome 9.0    | <input type="checkbox"/> Opera 9.63               | <input checked="" type="checkbox"/> SeaMonkey 1.1  |
| <input checked="" type="checkbox"/> ELinks 0.12   | <input checked="" type="checkbox"/> Firefox 2.0   | <input type="checkbox"/> Opera 9.64               | - No waiting                                       |
| <input type="checkbox"/> Epiphany 2.22            | <input type="checkbox"/> Firefox 3.0              | <input checked="" type="checkbox"/> Opera 9.80    | - Check dev & test sites behind your firewall      |
| <input checked="" type="checkbox"/> Epiphany 2.30 | <input type="checkbox"/> Firefox 3.5              | <input checked="" type="checkbox"/> Safari 4.0    | - Supports basic auth & form based logins          |
| <input checked="" type="checkbox"/> Firefox 1.0   | <input checked="" type="checkbox"/> Firefox 3.6   | <input checked="" type="checkbox"/> Safari 5.0    | - Private results                                  |
| <input checked="" type="checkbox"/> Firefox 2.0   | <input checked="" type="checkbox"/> Firefox 4.0   | <input checked="" type="checkbox"/> SeaMonkey 1.1 | - Free 7 day trial                                 |
| <input type="checkbox"/> Firefox 3.0              | <input checked="" type="checkbox"/> Flock 2.6     | <input checked="" type="checkbox"/> SeaMonkey 2.0 | <b>CrossBrowserTesting</b>                         |
| <input type="checkbox"/> Firefox 3.5              | <input checked="" type="checkbox"/> Opera 10.0    | <input type="checkbox"/> Shiretoko 3.5            |                                                    |
|                                                   | <input checked="" type="checkbox"/> Opera 10.60   |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Flock 7.0     |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Minefield 3.6 |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Minefield 3.7 |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Mozilla 1.7   |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> MSIE 8.0      |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> MSIE 9.0      |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Navigator 9.0 |                                                   |                                                    |
|                                                   | <input type="checkbox"/> Opera 10.0               |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Opera 10.60   |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> K-Meleon 1.5  |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Minefield 3.7 |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> MSIE 6.0      |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> MSIE 7.0      |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> MSIE 8.0      |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Navigator 9.0 |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Netscape 8.1  |                                                   |                                                    |
|                                                   | <input type="checkbox"/> Opera 10.50              |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Opera 10.63   |                                                   |                                                    |
|                                                   | <input type="checkbox"/> Opera 11.1               |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Opera 11.10   |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Opera 7.54    |                                                   |                                                    |
|                                                   | <input checked="" type="checkbox"/> Opera 8.54    |                                                   |                                                    |
|                                                   | <input type="checkbox"/> Opera 9.27               |                                                   |                                                    |

## Browser Cam

**BROWSERCAM - Cross Browser Compatibility Testing Tools**

See your web design on any browser on any operating system. Check javascripts, DHTML, forms and other dynamic functionality on any platform. Not just yours. Use our bank of testing machines remotely to test your website.

## Services

| Browser Capture                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Device Capture                                                                                                                                                                                                                                                                                                                                 |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  <ul style="list-style-type: none"><li>• Test your site on any <a href="#">browser and operating system</a></li><li>• Create and share screen shots</li><li>• Speed up your workflow</li><li>• <a href="#">Gomez WebInsight</a> - Easiest way to capture any page, even form results. Click <a href="#">here</a> for an introduction video of this tool.</li><li>• <b>NEW: Now Featuring IE 9!</b></li></ul> <p><a href="#">Features</a>   <a href="#">Try</a>   <a href="#">Flash Demo</a> <a href="#">Read More&gt;&gt;</a></p> |  <ul style="list-style-type: none"><li>• Test your site on <b>iPhone OS, Android, Blackberry, and Windows Mobile</b></li><li>• Create and share screen shots</li><li>• <b>NEW: Now Featuring iOS 4.3!</b></li></ul> <p><a href="#">Read More&gt;&gt;</a></p> |

~~~ End of Article ~~~



Session 2: Introduction to HTML5

SEO Tutorial – Meta Tags Optimization

| | |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Source | http://www.seo-gold.com/seo-tutorial/meta-tags-optimization |
| Date of Retrieval | 27/7/2012 |

There used to be a time when the contents of a pages Meta Tags was very important, it was around about the same time the Berlin Wall was still standing!! Today meta tags hold little value to Search Engine Optimization (SEO).

What Are Meta Tags?

Way, way back when that wall was still upright search engine algorithms were so dumb they couldn't work out what a page was about just from the content. So some bright spark had the ingenious idea to create a set of tags (meta tags) that inferred information about a pages content to the search engines.

Great idea, except there was nothing to stop a webmaster stuffing or spamming their Meta Tags with irrelevant, but very high traffic keywords and keyword phrases. Which of course they did with enthusiasm, you would find adult sites using words like Disney and Pokemon in their Keywords Meta Tag for the traffic!!

Today the vast majority of meta tags are worthless and those that are still considered by search engines aren't worth that much. For example Google confers no benefit from any meta tags, so if you expect a high Google ranking from perfectly optimized keywords in your meta tags, don't hold your breath.

Which Meta Tags Should You Use?

For Google adding the Description Meta Tag won't result in a boost in the Search Engine Results Pages (SERPs), but the description might be used for the description for your SERP listings in Google. So though you won't get a ranking boost, if you write an interesting Description Meta Tag and Google uses it (not guaranteed), you might get a higher click through rate compared to a random snippet of text from your pages. All other meta tags (including the Keywords Meta Tag) are either completely ignored or won't result in a SERPs boost.

Yahoo says they use the Keyword Meta Tag when it ranks a page. So it makes sense to add one for Yahoo and any other minor search engines that still use. Also there are directories and other websites that automatically take this information to create a listing to your site. Don't fret over it though, add the main phrase for that page to the Keywords Meta Tag and user friendly description of the page to the Description Meta Tag and forget about it.

~~~ End of Article ~~



## Session 3: Formatting Text Using Tags

### Formatting Text in HTML

|                           |                                                                                                                         |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <b>Source</b>             | <a href="http://www.chebucto.ns.ca/help/Html/Formatting.shtml">http://www.chebucto.ns.ca/help/Html/Formatting.shtml</a> |
| <b>Date of Retrieval:</b> | 27/7/2012                                                                                                               |

#### What HTML ignores

If you've tried writing some HTML already then you've probably noticed that it doesn't pay much attention to how you format the text. All of the formatting in HTML is done with tags. Because of this, formatting the text does nothing. Here are some examples of what HTML doesn't do:

- Multiple spaces are ignored, only 1 space is shown.
- Tabs are displayed as a single space.
- Multiple tabs are ignored, only 1 space is shown.
- Returns are ignored completely.
- line breaks are ignored completely.

What this means is that all but single spaces get ignored, and it doesn't matter how a line ends.

#### How to format text in HTML

Now that you know what HTML ignores, you're probably wondering how to format your text in HTML. You have to use tags to do this. This is an advantage though because then it doesn't matter how big the text is when a person looks at the page, the browser takes care of that. Because the text could be displayed wider or narrower, bigger or smaller by different machines this lets you not have to worry about that. The same paragraph could be a very different length on 2 different browsers because one of them might display the text in a bigger window, but both would look right.

The first formatting tag is the `<p>` tag. Normally this tag is does not need a `</p>` to close it, as soon as the browser sees another `<p>` tag (or any other tag that starts a new line) it is assumed that the last one has ended. This tag is the paragraph tag, and is probably the tag you will use the most if you put a lot of text on your pages. It tells the browser that the text is to be formatted like a paragraph. What this means is that the browser will put in a blank line and then start to display the text, wrapping to new lines as needed. When the next `<p>` tag comes up the browser puts another blank line and then does the text that follows that tag.

If you want to end one line but not put in a blank line like this:

First line

Second line

Then you need to use the `<br>` tag, the break tag. All that this does is end the line you are on.

The `<br>` tag is also useful if you want to make multiple blank lines. You can't do this with multiple `<p>` tags, they just collapse into one blank line. Instead, you use `<br>`, first put a `<br>` tag at the end of the line you're on, then put a `<br>` tag for each blank line you want.

### Aligning text

You now know how to format paragraphs but what if you don't want the text to look just like a normal paragraph? What if you want something just on the right side, or centered? To do this you use the align attribute. An attribute is additional information that you can put into a tag. The `<p>` tag can accept many attributes (most of which are experimental and so are not well supported), including the align attribute. To use any attribute you just put it into the tag:

```
<p align="center">text</p>
```

Note that here you have to put in a `</p>` tag to close the paragraph, this is so that the browser is certain where the special formatting ends. You can use this to align the text left, right, or center.

~~~ End of Article ~~



Select Easy-To-Read Fonts

| | |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| Source | http://www.netmechanic.com/news/vol6/usability_no17.htm |
| Date of Retrieval: | 27/7/2012 |

Select Easy-To-Read Fonts

by Larisa Thomason,
Senior Web Analyst,
NetMechanic, Inc.

How often have you visited a Web page and strained to read the text? Do visitors to your Web site have the same complaint? They may - unless you've selected fonts that are easiest for online reading.

The Web Is Not Print

A printed page always looks the same. You can read it in the library, in the bathtub, or on the deck chair of an ocean liner. No matter what your location, the type, size, and colors will always look the same.

But the print on a Web page is subject to many more variables:

- **Screen resolution:** A font face and size that looks great at 800x600 may be indecipherable at 1024x768. That's because you have a lot more pixels pushed into a smaller space. Although it seems backwards, the "bigger" the resolution numbers, the smaller the page elements appear.
- **Screen size:** Many schools and people with limited space (and money) are still using 15 inch or smaller monitors. Other visitors have 19 inch or larger monitors that look more like movie screens than computer monitors.
- **Different platforms and browsers:** PCs and Macintosh systems support different fonts and display them differently. There are even differences between how Netscape and Explorer display common fonts like Arial and Times New Roman.

In these cases, the only quick and reliable way to know how your page looks is to test it with Browser Photo. You'll see actual screen shots of your Web page in 16 different browser, operating system, and screen resolution combinations. You'll know in an instant if you've selected a problem font.

Selecting A "Good" Font

But if you've used a problem font, how can you find one that works for the majority of your visitors? Usability testing is vitally important, but it may not give you all the information you need about font selection. The problem is that usability tests usually involve just a small number of people and your personal PC.

That's why the studies done for the Software Usability Research Laboratory (SURL) at Wichita State University are such valuable resources. In January 2002, SURL published the results of a research study titled: "A Comparison of Popular Online Fonts: Which Size and Type is Best?."

It's helpful to read the entire study, but we're going to summarize some of the most important points here:

1. **Legibility:** The most legible fonts were Arial, Courier, and Verdana. Comic Sans was found to be the most illegible of the eight fonts evaluated.
2. **Attractiveness:** Study participants found Georgia and Times New Roman the most attractive fonts.
3. **Font size:** At the 10 point size, participants preferred Verdana. Times New Roman was the least preferred. At the 12-point size, Arial was the most preferred and Times New Roman the least preferred. Overall, Verdana was the most preferred font and Times New Roman the least preferred.

Now, some of these results appear to conflict. How could Times New Roman be the most attractive font and at the same time be the least preferred - no matter what the size?

The study's authors speculate that most people are just more comfortable with Times New Roman because they see it a lot:

"It is possible that Georgia and Times were considered attractive because of their widespread use in both print and on computer screens (Times also serves as the primary default for Microsoft Office software suites) and, thus, participants were more familiar with this type of font."

Quicker Is Better Online

During the study, the researchers found "significant differences in reading time. Generally, Times and Arial were read faster than Courier, Schoolbook, and Georgia." They also note that overall

"...Verdana appears to be the best overall font choice. Besides being the most preferred, it was read fairly quickly and was perceived as being legible."

Remember that people read differently online than in print. They quickly scan and look for important points before they settle down and begin reading the content

closely. So your font choice and size are important! Anything that makes the page harder to read makes visitors more likely to click away to another site.

But font choices aren't the only determining factor in page legibility. You have other options too:

- **Header tags** help define page structure for easy scanning. Keywords in header tags will also help boost your site in the search engine rankings.
- **Descriptive link text** gives important clues about the link's destination and content. Visitors are more likely to click when they know where they're going!
- **Color choices** give visitors visual clues about important content by drawing the eye towards highlighted sections. Just be sure you don't give important information using only color as a guide. That will lock out visitors using assistive technologies like screen readers.
- **Contrast.** Always make sure there's enough contrast between your text and page background color - particularly if you're using a background image on the page! You may find that you have to use a larger text size to compensate for the visual distraction from the background image.

All these are important components of page legibility and usability. Consider them all carefully: font, font size, page structure, links and navigation, color, and contrast. When they work together well, visitors see a well-designed, easy-to-read, and informative page. They're likely to stay for a while and make many return visits!

~~~ End of Article ~~



## Session 4: Creating Hyperlinks and Anchors

### Images as Hyperlinks

|                    |                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| Source             | <a href="http://www.simplehtml.co.uk/pages/images3.html#images">http://www.simplehtml.co.uk/pages/images3.html#images</a> |
| Date of Retrieval: | 27/7/2012                                                                                                                 |

#### *Using an Image as a Hyperlink*

As mentioned before you can use an image, like a button for example, as a hyperlink. If you want to put images on your pages as illustrations, perhaps your holiday photos, as these are likely to be comparatively large you could consider putting a thumbnail image (a smaller copy of the full size image) just big enough to show what it is, and make it into a hyperlink to a page containing the full size image. Then only if people want to see the full image will they have to wait for it to load.



The thumbnail image on the left is a hyperlink, click on it to see the full size image.

The normal code for the image hyperlink above would look **like this**:

```
<a href="harley_sketch.gif"></a>
```

#### *Using one Image for Multiple Hyperlinks*

As the name suggests an image map is a navigational object based on an image. Instead of an entire image being used as a hyperlink hotspot a single image can have many different hyperlinks associated with it. Additionally these hotspots can be different shapes according to the image used. One obvious example would literally be a map with hyperlinks in specific geographic locations. Alternatively you may have shaped button type images on a transparent background and you only want the buttons to be clicked, not the background as well.



The image on the left is a simple image map. As your mouse pointer moves over the map it will change to the "click" finger when you are over a hotspot defined in the HTML code. You should also see a description of the target.

England and Wales are rectangular hotspots, Northern Ireland is a circular hotspot and Scotland is a polygon hotspot. You can have more than one hotspot linked to the same target, here for example England has a large rectangle for the main part of the country and a smaller one for the south-west corner, underneath the hotspot for Wales. Depending on the browser version you are using you may see the outline of the hotspot appear when you click on one with your mouse.

These are the HTML tags you will need to create an image map.

```
<map name="map_of_uk">    map lets the browser know that the following code is
an image map. The name tag is to associate this set of
```

map co-ordinates with the correct map image.

```
<area shape="rect"
coords="0, 20, 100, 50"
href="index.html"
alt="Description">
```

This `area` tag is for `shape` that is a **rectangle**. The `coords` are the co-ordinates in pixels of the top left x and y of the hot spot, followed by the bottom right x and y co-ordinates. The target is specified by the `href` variable in the usual way, complete with the `alt` text for a description.

```
<area shape="circle"
coords="50, 100, 25"
href="index.html"
alt="Description">
```

This `area` tag is for `shape` that is a **circle**. The `coords` are the centre of the circle x and y co-ordinates in pixels, followed by the radius of the circle. The target is specified by the `href` variable in the usual way, complete with the `alt` text for a description.

```
<area shape="polygon"
coords="10, 60, 100, 80,
70, 60, 20, 10"
href="index.html"
alt="Description">
```

This `area` tag is for `shape` that is a **polygon**. The `coords` are the co-ordinates in pixels of the x and y corners of the hot spot. The target is specified by the `href` variable in the usual way, complete with the `alt` text for a description.

```
</map>
```

This lets the browser know that this is the end of the MAP co-ordinates list.

```

```

This is an image tag just like you would normally use for an image with the addition of the `usemap` attribute that must show the name used in the `map name` tag above. This line could go before the `map` tag without altering how the code works.

If any of the areas overlap the browser will search the list and use the first hyperlink listed for those co-ordinates. For people with older browsers that do not recognise client-side image maps, or those who have images turned off (or in case your image becomes deleted or corrupted) always provide corresponding text links to the hotspots on your map. It is common to put a block of text links at the bottom of the web page that replicate the links on the page.

The code for the image map above looks **like this**:

```
<map name="map_of_uk">
<area shape="rect" coords="90, 224, 131, 259" href="england.html"
alt="England">
<area shape="rect" coords="129, 112, 217, 241" href="england.html"
alt="England">
<area shape="rect" coords="94, 164, 128, 218" href="wales.html"
alt="Wales">
<area shape="circle" coords="68, 132, 23" href="ireland.html"
alt="Northern Ireland">
<area shape="polygon" coords="146, 100, 99, 131, 56, 66, 80, 23,
129, 21, 149, 50" href="scotland.html" alt="Scotland">
</map>
```

```

```

## ***Defining Your Co-ordinates***

If you use an image that you have created you can check where the co-ordinates should be by looking at your image in the package you used to create it, using its' built in measurement tools. For example in Microsoft Photo Editor and Microsoft Image Composer the current position of the mouse is shown on the status bar. Make a note of the co-ordinates at the appropriate positions on your image and enter them into your code. When you test your web page you can make any adjustments to the co-ordinates as necessary.

You can use the same technique to check the co-ordinates of other images by opening them in your image editing software too.

~~~ End of Article ~~



How do I change the color of my link's?

| | |
|---------------------------|--------------------------------------------------------------------------|
| Source | http://www.angelfire.com/nm/thehtmlsource/jazzup/linkcolormouseover.html |
| Date of Retrieval: | 27/7/2012 |

To make your text links change color when you place your mouse over them, you have to add a little bit of **CSS (Cascading Style Sheets)** to your web page.

How To Do It

Place the following code into the **<HEAD>** section of your web page, right after the **<title>** tag:

```
<style type="text/css">
<!--
A:link { text-decoration: underline; color:"#YourColor" }
A:visited { text-decoration: underline; color:"#YourColor" }
A:hover { text-decoration: underline; color:"#YourColor" }
-->
</style>
```

Configuring The Script

To make this work correctly on your web page, you need to replace (**#YourColor**) in the above code for all 3 attributes (**A:link**, **A:visited**, **A:hover**) with the actual color code of the color you want. If you don't know what the color code is of the color you want, **Click Here** to see a complete list of all the colors and there color codes.

A:link will be the normal color of your text links.

A:visited will be the color of the text links that you have already visited.

A:hover will be the color of the text link when you have your mouse over it.

TIP: If you don't like the underline on your text links, you can replace the word (**underline**) in the above code to the word (**none**).

~~~ End of Article ~~



## Session 5: Introduction to CSS3

### CSS -- Cascading Style Sheets -- Concepts

<http://cnx.org/content/m36730/latest/>

#### ***Introduction***

Cascading Style Sheets (abbreviated CSS) are used to describe how (X)HTML or XML text is to be presented. This module is currently a commented link list (or a draft of it) of important entry points for reading about CSS.

This module gives a simple example how a CSS style sheet works.

#### ***CSS rules***

A CSS file consists of statements which are either at-rules or rule sets (often just called rules). A rule set (rule) consists of a selector and a declaration block which contains one or more declarations (more see Specification). A declaration contains a property and a value (e.g. "color" and "green"), see example below. These rules describe how the formatting should take place.

Below are some sample rules. First come one or several selectors and then the attribute value pair or pairs.

```
/* Sample rules */  
  
h1, h2 {color: green}  
  
#box {width: 70%}  
  
.topic {color: red}
```

---

Sample rules

#### **Selector types**

- h1 and h2 are tag selectors
- #box is an selector for an identification
- .topic is a selector for a class

#### ***Internal and external style sheets***

A style sheet may reside in a separate file (**external style sheet**) or within an (X)HTML file (**internal style sheet**).

#### **External style sheet**

```
<head>  
<link rel="stylesheet" type="text/css" media="screen, projection,  
handheld, print" href="css/general.css" />  
<link rel="stylesheet" type="text/css" media="print"  
href="css/ourPrintStyleSheet.css" />
```

```
</head>
```

In the case of the code above from an HTML file we have a general style sheet for different kinds of output media called `general.css` and a specific one for printing (`ourPrintStyleSheet.css`) which adds and overrides some rules specific for printing.

### Internal stylesheet (HTML)

```
<style type="text/css">
  h1 {
    color: green;
  }
</style>
```

### Internal stylesheet (XHTML)

```
<style type="text/css"><![CDATA[
  h1 {
    color: green;
  }
]]></style>
```

### @import rule

The `@import` rule allows you to import rules from another style sheet into your current one. For an example see 'web fonts' below.

## Reference

CSS 2.1 specification (implemented by most browsers); an [index](#)

All W3C CSS standards and drafts

The link above includes the documents commonly called CSS 3 which consists of different modules. They are implemented to various degrees in the current browsers.

## Tutorial

w3.org: Starting with HTML and CSS (basic introduction)

Complete course with interactive exercises [w3schools](#)

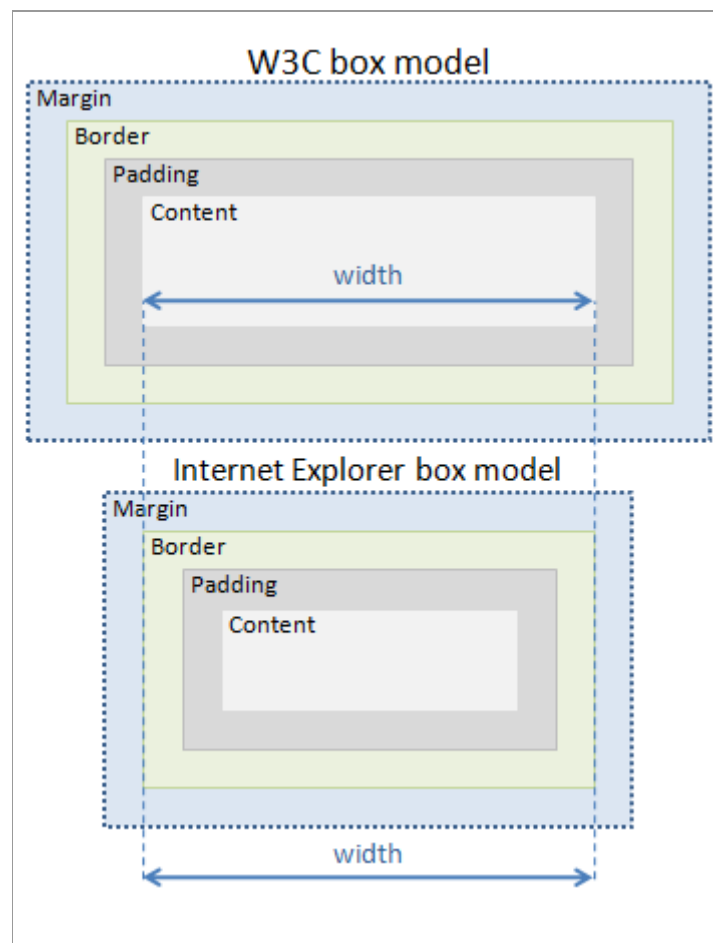
Searching for the key words `css cheat sheet` provides helpful links if you have learned CSS in the past. This brings up for example this list which leads to [30 examples](#).

## Layout with CSS

For doing layout with CSS the 'position' attribute and the box model are used. In the past this was difficult because the box model implemented by the Internet Explorer (IE) differed from the web standard and from the box model implemented by other browsers. In the meantime the Internet Explorer supports the standard W3C box model so using CSS in modern browsers is made easier as there needs not to be a 'switch' anymore for IE and the other browsers. Or at least the effort is reduced.



### CSS box model



**Figure 1:** The W3C standard and the Internet Explorer (earlier versions) box model

#### ***Validator***

A validator checks if a CSS file contains no erroneous CSS selectors and rules. [W3C validator](#)

#### ***CSS 3***

CSS3 is modularized and consists of several separate recommendations. See [current work](#) of W3C.

News about the implementation of the new features [css3.info](#). Google demonstration [website](#).

Interactive construction of CSS3 code (e.g. rounded corners, shadow, etc) [css3please.com](#)

- Multicolumn Layout e.g. `column-width: 20em;`
- Rounded Corners
- Web Fonts (see example below)

- Text Wrapping
- Text Stroke
- Transitions
- 2d Transforms
- Animations
- Gradients
- Selectors

### ***Implementation***

- <http://www.quirksmode.org/compatibility.html>
- <http://www.caniuse.com/> answers in which web browsers a particular CSS selector or attribute may be used.
- Gecko (used in Firefox) CSS support chart

### ***Debugging***

There is an add-on called Firebug to the Firefox web browser which helps to debug CSS style sheets. It also helps to learn about CSS <http://getfirebug.com/css>.

Internet Explorer 8 and 9 give access to developer tools ([Manual](#)) through the F12 key.

### ***CSS versus XSL***

Why does W3C recommend two different style languages? Which one should you use? Basically, the rule can be summarized very succinctly: Use CSS when you can, use XSL when you must. <http://www.w3.org/Style/CSS-vs-XSL>

### ***CSS in mobile devices***

CSS mobile (current status) aims at describing a subset of CSS useful for mobile devices.

### ***Media queries***

CSS **media queries** (W3C spec) allow that style rules are only applied to specific output media, i.e. screen, print, handheld or other. They can also depend on conditions like display width. The following example demonstrates the idea

```
@media (min-width: 320px) {  
    /* rules for devices with larger screens go here */  
}  
  
@media (max-width: 320px) {  
    /* rules for mobile devices go here */  
}
```

}

[More](#) about media queries.

## Frameworks

A CSS framework is one or more CSS files which include style declarations on which you can build by either extending or replacing them. The aim is that you do not need to create workarounds for the browser inconsistencies.

YAML (web site in German), in English

YUI grids

Less Framework 3 is a cross-device CSS grid system based on using inline media queries.

Blueprint.css

### Mobile devices

[iwebkit](#) (Ipad, Iphone)

[cssgrid](#) -- all the way down from a desktop browser to a mobile device.

## Web fonts

Current browsers may load fonts from the web (@font-face property) and thus support the font part of [CSS 3](#).

### Google font service

Google provides a service from which you can choose free fonts. It generates some code which you can add to your website. They also provide a font API and host a repository with free fonts. So you might add the **Gentium** font by adding an import rule at the beginning of your style sheet

```
@import url(http://fonts.googleapis.com/css?family=Gentium+Basic&subset=latin,latin-ext);
```

But you may as well copy a free font to a directory of your own website and serve it from there. An example of loading a font.

```
@font-face {
  font-family: Gentium;
  src: url(http://yoursite/fonts/Gentium.ttf);
}
```

```
p { font-family: Gentium, serif; }
```

Gentium is a free font under the Open Font License (OFL).

***XHTML plus CSS to PDF***

- Printing a book with CSS (PrinceXML)
- pd4ml (free for non-commercial use)

~~~ End of Article ~~~



Session 5: Introduction to CSS3

CSS3 border module example

Source <http://hyeonseok.com/docs/css/css3-border.php>

border-color

Required border-style, inherited from font color by default

border-color

```
p {  
    border-color: #0c0;  
}
```

border-top-color

```
p {  
    border-top-color: #0c0;  
}
```

border-right-color

```
p {  
    border-right-color: #0c0;  
}
```

border-bottom-color

```
p {  
    border-bottom-color: #0c0;  
}
```

border-left-color

```
p {  
    border-left-color: #0c0;  
}
```

border-style

none

```
p {  
    border-style: none;  
}
```

hidden

```
p {  
    border-style: hidden;  
}
```

dotted

```
p {  
    border-style: dotted;  
}
```

dashed

```
p {  
    border-style: dashed;  
}
```

solid

```
p {  
    border-style: solid;  
}
```

double

```
p {  
    border-style: double;  
}
```

groove

```
p {  
    border-style: groove;  
}
```

ridge

```
p {  
    border-style: ridge;  
}
```

inset

```
p {  
    border-style: inset;  
}
```

outset

```
p {  
    border-style: outset;
```

```
}
```

border-width

thin

```
p {  
    border-width: thin;  
}
```

medium

```
p {  
    border-width: medium;  
}
```

thick

```
p {  
    border-width: thick;  
}
```

border-top-width

```
p {  
    border-top-width: thick;  
}
```

border-right-width

```
p {  
    border-right-width: thick;
```



```
}
```

border-bottom-width

```
p {  
    border-bottom-width: thick;  
}
```

border-left-width

```
p {  
    border-left-width: thick;  
}
```

border-radius

border-radius

```
p {  
    border-radius: 1em;  
}
```

border-top-right-radius

```
p {  
    border-top-right-radius: 1em;  
}
```

border-bottom-right-radius

```
p {  
    border-bottom-right-radius: 1em;
```

```
}
```

border-bottom-left-radius

```
p {  
    border-bottom-left-radius: 1em;  
}
```

border-top-left-radius

```
p {  
    border-top-left-radius: 1em;  
}
```

1em 3em (top, right)

```
p {  
    border-top-right-radius: 1em 3em;  
}
```

3em 1em (top, right)

```
p {  
    border-top-right-radius: 3em 1em;  
}
```

3em 1em (top-left & bottom-right, top-right & bottom-left)

```
p {  
    border-radius: 1em 3em;  
}
```

1em 3em 2em (top-left, top-right & bottom-right, bottom-left)

```
p {
    border-radius: 1em 3em 2em;
}
```

1em 3em 2em 1em (top-left, top-right, bottom-right, bottom-left)

```
p {
    border-radius: 1em 3em 2em 1em;
}
```

1em[1] / 3em[2]

```
p {
    border-radius: 1em / 3em;
    /*
    border-top-left-radius:    1em[1] 3em[2];
    border-top-right-radius:   1em[1] 3em[2];
    border-bottom-right-radius: 1em[1] 3em[2];
    border-bottom-left-radius: 1em[1] 3em[2];
    */
}
```

1em[1] 3em[2] / 2em[3]

```
p {
    border-radius: 1em 3em / 2em;
    /*
    border-top-left-radius:    1em[1] 2em[3];
    border-top-right-radius:   3em[2] 2em[3];
    border-bottom-right-radius: 1em[1] 2em[3];
    */
}
```

```

border-bottom-left-radius: 3em[2] 2em[3];

*/

}

```

1em[1] 3em[2] 2em[3] / 2em[4] 1em[5]

```

p {

border-radius: 1em 3em 2em / 2em 1em;

/*

border-top-left-radius:      1em[1] 2em[4];
border-top-right-radius:     3em[2] 1em[5];
border-bottom-right-radius:  2em[3] 2em[4];
border-bottom-left-radius:   3em[2] 1em[5];

*/

}

```

*1em[1] 2em[2] 1em[3] 3em[4] / 3em[5] 2em[6] 3em[7]
2em[8]*

```

p {

border-radius: 1em 2em 1em 3em / 3em 2em 3em 2em;

/*

border-top-left-radius:      1em[1] 3em[5];
border-top-right-radius:     2em[2] 2em[6];
border-bottom-right-radius:  1em[3] 3em[7];
border-bottom-left-radius:   3em[4] 2em[8];

*/

}

```

border-image

```
p {  
    border-image: url(9box.png) 27 round stretch;  
}
```

border-image (vendor prefix)

```
p {  
    -o-border-image: url(9box.png) 27 round stretch;  
    -webkit-border-image: url(9box.png) 27 round stretch;  
    -moz-border-image: url(9box.png) 27 round stretch;  
    border-image: url(9box.png) 27 round stretch;  
}  
  
div {  
    border-width: 79px 75px 79px 233px;  
    -o-border-image: url(border-image.png) 79 75 79 233 round;  
    -webkit-border-image: url(border-image.png) 79 75 79 233 round;  
    -moz-border-image: url(border-image.png) 79 75 79 233 round;  
    border-image: url(border-image.png) 79 75 79 233 round;  
}
```

~~~ End of Article ~~~



## Session 6: Formatting Using Style Sheets

### border-image

**Source** <http://www.css3files.com/border/>

Images can be assigned to borders based on a single image file. The `border` property must be set.

#### Compatibility

`-moz-border-image` **Firefox**  
`-webkit-border-image` **Safari 5+, Chrome 10+**  
`-o-border-image` **Opera 11+**  
`border-image` **Official syntax**

Not supported by Internet Explorer 9, only partly supported by Firefox and Opera.

#### General description

`border-image: 1url(image.png) 23 / 35px 4round`

**1**The path to the background image file.

**2**Specifies how the image is applied to the border.

Can be used in two ways: **Matrix applied** and **Whole image taken**.

**3(Optional, separated by a slash)** Border width. Can be different from value 2 and the `border-width` property set. General **shorthand rule** applies.

**4(Optional, separated by a slash)** Specifies how the image is fitted to the four sides of the border. Possible values are: `stretch` (default), `round` and `repeat` (see examples). If one value is set it is taken for all four sides of the border, with two value the first applies to top/bottom and the second to left/right. This value has no effect on the corners.

Matrix applied

The main usage of `border-image` is to provide an image file that represents the four corners and four sides of the border as well as background of the element. In this case the image gets cut into nine slices, where each one represents a different part of the border.

`border-image: url (image.png) 13`

**1**Sets the size of the areas that are taken to draw border. Can consist of up to four values each representing one side of the border. General **shorthand rule** applies.



#### Whole image taken

It is also possible to use the whole image for the border. Not supported by Firefox and Opera.

`border-image: url(image.png) 1100%`

[Toggle highlighting](#)

1 Must be set to 100% or the actual dimensions of the image.



### Examples

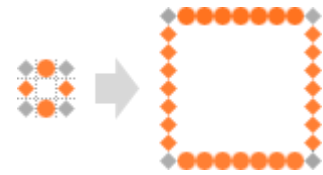
Tiled border image with matrix applied

border-image: url(image.png) 1 9 2 round

[Toggle highlighting](#)

1 The area taken for all the four sides is 9 pixels high/wide

2 The background image is tiled and fitted (round).



[Switch to live view](#)

### Border image with matrix applied and two sides stretched

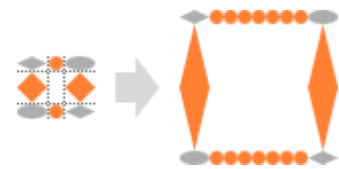
border-image: url(image.png) 1 7 2 15 3 round stretch

[Toggle highlighting](#)

1 The top and bottom parts of the background image are 7 pixels high.

2 The left and right parts are 15 pixels wide.

3 The top/bottom areas of the border image are tiled and fitted (round), the left/right areas are stretched to the whole height (stretch).



[Switch to live view](#)

### Whole image taken and repeated

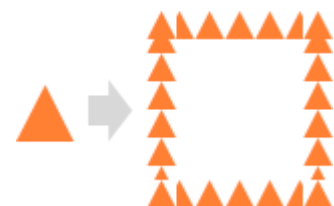
border-image: url(image.png) 1 26 / 2 13px 3 repeat

[Toggle highlighting](#)

1 All four sides of the border set equally and to the actual size of the border image. As the border image is 26 x 26 pixels this corresponds to 100%.

2 Border width set half the image width. This can also be achieved by setting the general border-width to the same value and omitting this one.

3 The background image is simply repeated (and centered) without the attempt to fit (repeat). Has almost the same effect like round.



[Switch to live view](#)

**Further reading**

For more info see the [W3C site](#), the [Safari site](#) and the [site at MDN](#) (including compatibility info for older browser versions).

**border-radius**

Is used to round the corners of an element that has the border or background property set.

**Compatibility**

`-moz-border-radius` **Firefox 3.6**

`-moz-border-radius-topleft`

`-moz-border-radius-topright`

`-moz-border-radius-bottomright`

`-moz-border-radius-bottomleft`

`border-radius` **Firefox 4+ Safari 5+, Chrome 10+ Internet Explorer 9+ Opera 11+ Official syntax**

`border-top-left-radius`

`border-top-right-radius`

`border-bottom-right-radius`

`border-bottom-left-radius`

Supported by all modern browsers. To maintain compatibility with older browser versions it however is advised to additionally use `-webkit-border-radius`, which orients on the the official syntax, but with one [difference](#). For detailed compatibility info see [caniuse.com](#).

- Firefox 3.6  
No betas, nightlies or suchlike, only final versions. Safari includes mobile browsers based on Webkit.

**General description**

`border-radius: 18px / 213px`

**1**The radius of the quarter circles that form the shapes of the corners.

**2 (Optional, separated by a slash)** The vertical radius, which results in a quarter ellipse when different from the horizontal radius.

**Examples**

Four values given

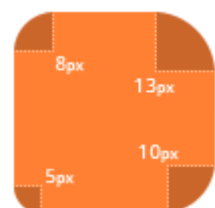
`border-radius: 18px 213px 310px 45px`

**1**Top left corner.

**2**Top right corner.

**3**Bottom right corner.

**4**Bottom left corner.



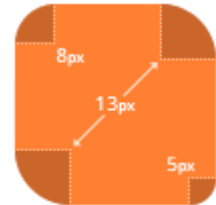


[Switch to live view](#)

Three values given

```
border-radius: 18px 213px 5px
```

- 1 Top left corner.
- 2 Top right **AND** bottom left corner.
- 3 Bottom right corner.

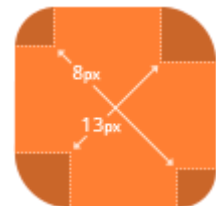

[Switch to live view](#)

Two values given

Note: When using `-webkit-border-radius` it is assumed that the first value stands for the horizontal radii and the second for the vertical radii like [here](#), which is completely different from the official interpretation below.

```
border-radius: 18px 213px
```

- 1 Top left **AND** bottom right corner.
- 2 Top right **AND** bottom left corner.


[Switch to live view](#)

One value given

```
border-radius: 18px
```

- 1 All four corners rounded equally.

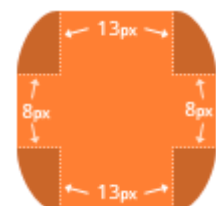

[Switch to live view](#)

Elliptical corners, two values given

With the addition of a slash it is possible to set the horizontal and vertical radii separately.

```
border-radius: 18px / 213px
```

- 1 All horizontal radii set equally.
- 2 All vertical radii set equally.



[Switch to live view](#)

Elliptical corners, four values given

Three and four values on each side of the slash follow the notation given above (without slash).

border-radius: 18px 213px / 310px 45px

1 Horizontal radii for top left **AND** bottom right corners.

2 Horizontal radii for top right **AND** bottom left corners.

3 Vertical radii for top left **AND** bottom right corners.

4 Vertical radii for top right **AND** bottom left corners.

~~~ End of Article ~~~



Session 7: Displaying Graphics and CSS3 Animation

Making a move with CSS3 animations

Source <http://dev.opera.com/articles/view/css3-animations/>

Introduction

Traditionally, the Web was a very static place. Achieving animations was not really possible in any sane way until JavaScript, animated GIFs and Flash came along, at which point we rejoiced and applauded the ensuing slew of skip intros and horrible obtrusive animations.

This was all well and good, but there was still no way for non-developers to create animations using open standards. You can spout all the religious arguments you want about animation belonging in the behavior layer rather than the presentation layer, but I think animation definitely falls in the realm of design. And now, with CSS3 transitions and animations, we can animate elements of our web documents. Standards-based web design with added fun! And added skip intros, if you're that way inclined...

Opera has supported transitions now for a long time, and we've already written about them in CSS3 transitions and 2D transforms. This article focuses on the other way to animate things using style sheets — **CSS3 animations**. Below we'll give you a concrete introduction including where the specification and browser support is at, how animations differ from transitions, basic syntax, and a list of examples.

How mature is the technology?

The CSS animations spec, proposed and edited by Apple, is currently in Working Draft status and is labeled as "Outdated" on the CSS working group current work page, so I think we can expect to see some minor changes before it is completed, but the basic principle should remain the same.

In any case, it has already been implemented experimentally in Firefox (since 5), Chrome (since 4), Safari (since 4), IE (in 10), and Opera since version 12. Note "experimentally" — this means that you need to use appropriate vendor prefixes for each browser.

Basic syntax

Animations differ from transitions in that transitions are only triggered when a state change occurs (such as hovering over an element), whereas animations can be triggered independent of a state change, usually a set amount of time after a page loads, or on an event, via JS. Also, a transition animation always occurs on any properties that change their values when a state change occurs, whereas a CSS animation can animate between property values even if those properties are not set on the default state of the element being animated.

Defining an animation

To set up an animation, you first have to define some animation key frames, in a special new at-rule block, which looks like this:

```
@key frames spin {  
  
    from { transform: rotate(0deg); }  
  
    to { transform: rotate(360deg); }  
  
}
```

Here we are saying that this animation is called spin, and what the animation does is smoothly rotates whatever it is applied to through 360 degrees. Nothing else is defined here at all, which may sound lacking, but in actual fact it is very flexible, as you can define an animation once, and then apply it to many different elements with different durations and other behavior.

You can also use 0% and 100% in place of from and to. If you like, you can include intermediate key frames, for more complex animations:

```
@key frames spin {  
  
    0% { transform: rotate(0deg); }  
  
    25% { transform: rotate(30deg); }  
  
    50% { transform: rotate(120deg); }  
  
    100% { transform: rotate(360deg); }  
  
}
```

You can write key frames that have the same values on one line, like this:

```
25%, 50% { opacity: 0.9; }
```

Note: You can set your animation so that the first key frame doesn't appear at 0%, e.g. 40%. If you do this, however, nothing will change before that point in the animation. If you don't set the last key frame as 100%, e.g. 70%, the animation will reach that point and stay the same up until 100%.

The current main annoyance of using animations is that you have to define a separate key frames block for each browser, as the at-rule itself uses a prefix, so @-o-key frames animation1 { }, @-moz-key frames animation1 { }, etc. And this is in addition to the different prefixed versions of all the animation properties you'll see later. But at least within each prefixed at-rule, you only need to include the single prefixed properties for that browser, in the case of properties that require prefixes. And there are a couple of JavaScript libraries worth looking at, which detect the rendering engine at

runtime and add the appropriate prefixes where needed, saving you having to write all of them out. See [Prefixfree](#) by Lea Verou, and [Prefixr](#) by Jeffrey Way.

Applying an animation

To animate an element using this animation, you apply it to the element using the `animation-name` property:

```
#image {  
  
    animation-name: spin;  
  
}
```

To make this do anything, you also need to tell the animation how long it should take to run from start to finish: this is done using `animation-duration`:

```
#image {  
  
    animation-name: spin;  
  
    animation-duration: 3s;  
  
}
```

These are all the properties you need to get an animation to run once on an element. Let's now look at what other properties we have to control animations.

How many times do we want it to happen?

To make our animation run a set number of times, you use `animation-iteration-count`:

```
#image {  
  
    animation-name: spin;  
  
    animation-duration: 3s;  
  
    animation-iteration-count: 10;  
  
}
```

The value of `animation-iteration-count` can be any positive whole number, or you can set it to infinite to make it go on forever. The default value is 1.

To see what we have so far in action, check out my basic spinner example.

Varying animation rate

The first example doesn't look too bad, but you'll notice that in each iteration of the animation, the sun starts off by spinning fast, then slows down to a stop. You can alter the rate of animation change by setting different values of the `animation-timing-function` property. The different possible values are:

- `linear` makes the animation happen at the same rate from beginning to end.
- `ease`, the default value, causes the animation to start quickly and then gradually slow down, as you've already seen.
- `ease-out` makes the animation start quickly, stay quick for longer than `ease`, and then slow down more abruptly at the end.
- `ease-in` makes the animation start off slowly, and then speed up toward the end.
- `ease-in-out` means that the animation starts off by accelerating, is quite fast for most of the duration, and decelerates toward the end.
- `steps()` is slightly different. Instead of giving a smooth animation throughout, this causes the animation to jump between a set number of steps placed equally along the duration. For example, `steps(10)` would make the animation jump along in ten equal steps. There's also an optional second parameter that takes a value of `start` or `end`. `steps(10, start)` would specify that the change in property value should happen at the start of each step, while `steps(10, end)` means the change would come at the end.
- `cubic-bezier()` applies your own custom cubic Bézier curve to dictate the change in animation rate. This function takes four arguments: the X and Y coordinates of the beginning control handle, and the X and Y coordinates of the end control handle: for example `cubic-bezier(.28, 1.48, .9, .02)`.

Don't understand cubic Bézier curves? Lea Verou has created a fantastic visual tool allowing you to easily visualize what we are on about, see the effects of different timing function values, and even generate your own cubic bezier values. Check out cubic-bezier.com.

To make the spinning sun look more consistent, I set `animation-timing-function` to `linear` — load up linear spinner to see the result.

To create a "bounce" effect, you can use a cubic Bézier curve value with drag handle values greater than the lower or upper bounds of the graph, e.g. `cubic-bezier(.2, -0.36, .71, 1.45)`

Animation delays

You can set a delay before the animation starts, by setting an `animation-delay` property:

```
#image {  
  
    animation-delay: 4s;
```

```
}
```

This property can take positive and negative values. A positive value will delay when the animation starts, whereas a negative value will make the animation start part way through the specified animation duration. You'll most commonly use this when you've got multiple animations that you want to fire at different times to provide a complete sequence.

Start to end, or back and forth?

By default, animations that run multiple times will go from start to end, then flick straight back to the start and go to the end again, and so on. You can instead make the animation go smoothly back and forth, so from start to end, then from end to start, start to end, and so on, by specifying `animation-direction: alternate;` on the element.

To see the effect of this, check out my alternate spinner example. Note that the timing function effect is reversed as well, on the alternate animations.

animation-fill-mode

The last property we'll look at is `animation-fill-mode`. This allows you to specify how the animated element is displayed after an animation ends or during an animation-delay. The possible values of `animation-fill-mode` are:

- `none` is the default value — by default, when an animation ends the element it is applied to will go back to using its intrinsic styling. In addition, no styling from animation key frames will be applied to an element during an animation delay.
- `forwards` makes an element with an animation applied to it retain the styles defined by the properties in the final key frame after the animation ends.
- `backwards` causes styles defined in the first key frame to be applied to the element the animation is applied to during an animation-delay, rather than the default element styles.
- `both` applies the combined effects of `forwards` and `backwards` to an element undergoing an animation.

To experiment with some of these effects, I've created another example including an animation that moves the spinning sun, and an animation delay. I've then made four versions, each of which has a different value of `animation-fill-mode`:

- `animation-fill-mode: none;` — notice that the sun starts in the middle of the screen as defined in the `#image` ruleset, then jumps to the position defined in the 0% key frame when the animation starts. At the end of the animation, it jumps back to the middle of the screen again.
- `animation-fill-mode: backwards;` — here the sun starts at the position defined in the 0% key frame, and stays there until the animation starts. At the end of the animation, it jumps back to the middle of the screen again.
- `animation-fill-mode: forwards;` — in this case, the sun starts in the middle of the screen as defined in the `#image` ruleset, then jumps to the position defined in the 0% key frame when the animation starts. At the end of the animation, it stays in the position defined in the 100% key frame.
- `animation-fill-mode: both;` — the sun now starts at the position defined in the 0% key frame, and stays there until the animation starts. At the end of the animation, it stays in the position defined in the 100% key frame.

Animation shorthand

You need to write a lot of code for CSS animations due to having to write multiple key frame blocks *and* multiple properties including all the different prefixes. Fortunately, you can use shorthand to seriously reduce the amount of code needed.

The following properties:

```
animation-name: spin;

animation-duration: 3s;

animation-timing-function: linear;

animation-delay: 3s;

animation-iteration-count: infinite;

animation-direction: alternate;

animation-fill-mode: both;
```

can be replaced by this one line:

```
animation: spin 3s linear 3s infinite alternate both;
```

The spec is not very specific in defining the exact order of the property values in the shorthand, but it's best to stick with the order shown above to avoid potential browser bugs. Various sources indicate that this order fulfils the idiosyncrasies different browsers currently have.

You *need* to include animation-name and animation-duration for the animation to do anything at all. If you don't explicitly specify the other values, their default values will be applied:

```
animation-timing-function: ease;

animation-delay: 0s;

animation-iteration-count: 1;

animation-direction: normal;

animation-fill-mode: none;
```


You can apply multiple animations in a single rule by including them in the same property, separated by commas. This works for both longhand and shorthand values. Here's an example:

```
animation: spin 3s, movement 5s;
```

```
animation-name: spin, movement;
```

```
animation-duration: 3s, 5s;
```

If you're using longhand properties with different numbers of values, you need to specify all the animation names to be applied. Thereafter, if any of the other properties have less values than the number of animations specified, they will be alternated to fill up the gaps. For example:

```
animation-name: spin, movement, glow;
```

```
animation-duration: 3s, 5s;
```

```
animation-delay: 2s;
```

The spin animation will have a duration of 3 seconds, and the zap animation will have a duration of 5 seconds. The glow animation will have a delay of 3 seconds again, because the duration values have run out, so they start from the beginning again. All of the animations will have a delay of 2 seconds.

A More involved example

To round off the article, I've created a slightly more interesting example with some more animations involved — check out my [Sunrise example](#) in all its glory. If you dissect the code, you'll see a number of animations working together to produce the final effect.



Summary

That's it for our basic tour of CSS animations — let us know what you think, and have fun.

~~~ End of Article ~~~



Session 8: Creating Navigational Aids and Division-based Layout

Structural Semantic Elements in HTML5

Source <http://cnx.org/content/m41595/latest/>

Summary: In HTML5 more tags were introduced to show the meaning (semantics) of the text it contains. Some of these semantic tags are for showing the structure of a web page like header, navigation part, a section for the content and footer.

Aim

A web page typically has a header part at the top containing a title, maybe a graphic and some other elements. Then there is a navigation area, one or more content sections and a footer.

In HTML5 semantic tags (like <header>, <footer>, <section>, <nav>, <aside>) were introduced to help people mark the structure of a web page.

These tags allow programs (e.g. search engines) to extract information better from the web pages as the elements are identified by a standard tag name.

Before HTML5

Before HTML5 people used to mark areas with **div tags** as this was the only element available to do this. Even before they used a table to do the layout. For the div tags attributes were used to distinguish the different parts. For a navigation box all the following code examples are possible.

```
<div id="navigation">.....</div>
```

or

```
<div id="navbox">.....</div>
```

or

```
<div class="navigation">.....</div>
```

or

```
<div class="nav">.....</div>
```

HTML5 structural semantic tags

- <header>
- <footer>
- <section>
- <nav>

- `<aside>`
- `<article>`

With this HTML5 standardization the ambiguity in the example of a navigation area mentioned above is eliminated. The code now is only

```
<nav>.....</nav>
```

.

<article> tag

The `<article>...</article>` tags are meant to enclose text content which can stand for itself - a self-contained unit.

Examples

Example of a simple page layout

```
<!DOCTYPE html>
<html>
<head>
<title>The title of the document</title>
</head>

<body>
<header><h1>Our web site</h1></header>
<nav><a href="details.html">Details</a></nav>
<section >the content</section>
<footer>Contact: ..... Email: .....</footer>
</body>

</html>
```

...will be expanded and commented later...

Another example

A more elaborate [example](#).

An example with articles.

```
<!DOCTYPE html>
<html>
<head>
<title>The title of the document</title>
</head>

<body>
<header>
  <h1>Our web site</h1>
  <nav><a href="details.html">Details</a></nav>
</header>

<section >
  <article>
```

```
...
</article>

<article>
...
</article>

<article>
...
</article>
</section>

<footer>Contact: ..... Email: .....</footer>

</body>
</html>
```

~~~ End of Article ~~~



CSS3 Multi-column layout

Source <http://dev.opera.com/articles/view/css3-multi-column-layout/>

Introduction

Multiple column design that allows text to flow naturally from column to column depending on width and other parameters has proven invaluable in print design, crossing languages, cultures and a range of media. It will therefore come as no surprise to anyone reading this that such a multiple column capability is one of the most hotly requested CSS feature additions!

The Multi-column Layout Module in the CSS3 specification consists of the layout algorithm, properties to create the column structure and properties to control the flow and break of multicolumn elements. In this article, I'll give you an introduction to using the features of this module in your CSS layouts.

Building columns by creating multicol elements

Since there is no specific markup element for multiple columns, CSS is used to modify a given element and turn it into a *multicol element*, which occurs automatically when certain column styles are set on an element.

Note: There are exceptions. A `table` element or any replaced block-level element cannot be made into a multicol element.

Review Listing 1, in which I've set up a header, a division, and several paragraphs of text.

```
<h2>Excerpt from the Saga Of Hakon Herdebreid </h2>

<div>

<p>Hakon, King Sigurd's son, was chosen chief of the troop which had
followed King Eystein, and his adherents gave him the title of king. He was ten
years old. At that time he had with him Sigurd, a son of Halvard Hauld of Reyr,
and Andreas and Onund, the sons of Simon, his foster-brothers, and many chiefs,
friends of King Sigurd and King Eystein; and they went first up to Gautland.</p>
```

```
<p>King Inge took possession of all the estates they had left behind, and  
declared them banished. Thereafter King Inge went to Viken, and was sometime  
also in the north of the country. Gregorius Dagson was in Konungahella, where  
the danger was greatest, and had beside him a strong and handsome body of men,  
with which he defended the country.</p>
```

```
<p>The summer after (A.D. 1158) Hakon came with his men, and proceeded to  
Konungahella with a numerous and handsome troop. Gregorius was then in the town,  
and summoned the bondes and townspeople to a great Thing, at which he desired  
their aid; but he thought the people did not hear him with much favor, so he  
did not much trust them.</p>
```

```
</div>
```

Listing 1: Some elements and content used to create a simple multicol element.

Without any author styles applied, the div and its child elements will simply remain in the normal flow. But now I'm going to turn the div to have multi-column layout by simply specifying either of these properties:

```
div {column-count: 4; }
```

or

```
div { column-width: 100px; }
```

- **column-width** sets the width of each column within the parent element. Values include pixels and ems, but not percentages.
- **column-count** sets the number of columns that should be displayed within the element.

You could specify them both, but the pseudo-algorithm used to calculate the number of columns would only use either one of them in most cases.

Figure 1 shows the result, as rendered by Opera 11.10.



Figure 1: Creating a multicol element.

Now all the content in the `div` flows from column to column. Each column acts as a containing block for their content. This means any floated element within that column would be aligned to that column box.

Note: columns do not establish containing blocks for elements with `position: fixed` or `position: absolute`.

Adding gaps and rules

Of course, you'll want to add gaps and rules to your columns as necessary. To add gaps and rules, use the following multicolumn-specific properties:

- **column-gap**: This property sets the length of the gap between columns.
- **column-rule-color**: Use any standard value for the rule color.
- **column-rule-style**: This property allows you to style rules using border styles from CSS 2.1 (eg dashed, dotted, etc.).

- **column-rule-width:** Sets the width of the rule between columns. The rule will appear in the middle of the gap.
- **column-rule:** The shorthand for setting `column-rule-color`, `column-rule-style`, and `column-rule-width`.

Here are the styles I've added:

```
div { column-gap: 20px; column-rule: 2px solid #33c; }
```

Now I've got a series of distinct columns, which can be seen in Figure 2. You'll note I removed the guiding outlines for the main elements.



Figure 2: Adding gaps and rule styles to the element to demonstrate the structure.

Controlling breaks in element flow

Just as we can break pages in print for increased organization and understanding, we can break elements with multi-column layout for paged and visual media. This provides a level of control over content flow, helping us to avoid columns breaking apart in odd places, such as at the point where an image or other object is placed within the columns.

If content spans multiple columns, you would want to control the occurrence of column breaks. We have three properties to do so:

- **break-before:** Control column breaking (prevent, allow, force) before an element inside of an element with multi-column layout.
- **break-after:** Control column breaking (prevent, allow, force) after an element inside of an element with multi-column layout.
- **break-inside:** Control the break behavior inside the element.

Each of these column break properties can take several values which lets you control how the text is spread out over multiple columns.

Spanning multiple columns

What if you'd like to have an element span across the full length of the element, for example if you wanted a heading or image to appear across the columns but retain the multicolumn flow? We can do just that easily by using the `column-span` property.

There are two values for `column-span`:

- `none`: The element will not span across columns.
- `all`: This will cause the selected element to span all columns.

In Listing 2, I've simply moved the `h2` into the element with multi-column layout and added a rule to the CSS, `h2 { column-span: all; }` in order to have it span across all the columns.

```
<div>

<h2>Excerpt from the Saga Of Hakon Herdebreid </h2>


<p>Hakon, King Sigurd's son, was chosen chief of the troop which had

followed King Eystein, and his adherents gave him the title of king. He was ten

years old. At that time he had with him Sigurd, a son of Halvard Hauld of Reyr,

and Andreas and Onund, the sons of Simon, his foster-brothers, and many chiefs,

friends of King Sigurd and King Eystein; and they went first up to

Gautland.</p>


<p>King Inge took possession of all the estates they had left behind, and

declared them banished. Thereafter King Inge went to Viken, and was sometimes

also in the north of the country. Gregorius Dagson was in Konungahella, where

the danger was greatest, and had beside him a strong and handsome body of men,

with which he defended the country.</p>


<p>The summer after (A.D. 1158) Hakon came with his men, and proceeded to

Konungahella with a numerous and handsome troop. Gregorius was then in the town,
```

```
and summoned the bondes and townspeople to a great Thing, at which he desired
their aid; but he thought the people did not hear him with much favor, so he did
not much trust them.</p> </div>
```

Listing 2: Creating a spanning element in multicol layout

Once you've applied a column-span to an element, that element then becomes what is referred to as a *spanning element*. You can see the effect in Figure 3.

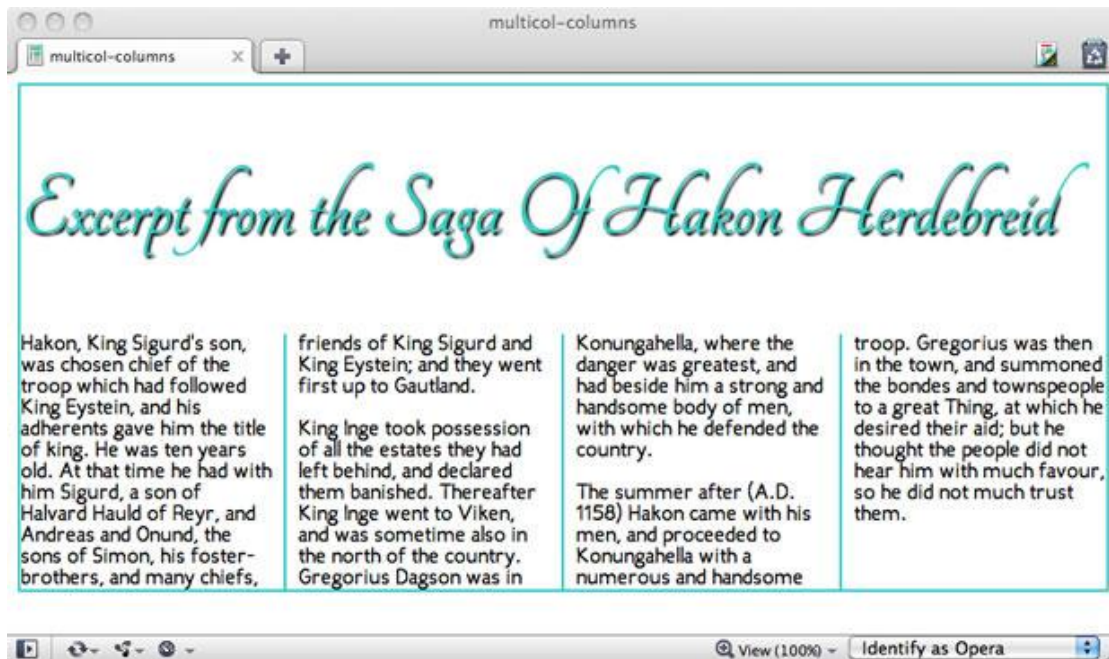


Figure 3: Creating a spanning element.

Balancing Columns

When you have multiple columns, browsers make an effort to make sure each column balances and that the variation in column length is minimal. You can change this by using the `column-fill` property. Setting it to `auto` (instead of the initial value of `balance`) could make the columns be filled sequentially which might result in some partially filled columns (or empty ones). You can see how different they look in this demo.

Now comes the fun part

Armed with this article, your favorite editor and Opera 11.10 (or Opera Mobile 11.10), you can begin experimenting with multi-column layout. You could conceivably create a layout combining multiple columns on the body, floated elements within the columns or even positioned elements. You can play around with spanning elements, using background images and transparency, and controlling content flow.

I encourage you to "just have fun with it," but I also want to add that multi-column layout, while widespread in print, is a new feature for Web developers and designers. As such, its uses and best practices for accessibility, usability, cross media and cross-browser interoperability are still immature. What's more, these are critical issues that cannot be addressed effectively in theory, so it is up to us to take this emerging layout feature and add it to our toolbox with care, thought and most important, creativity!

Browser Support

Opera 11.10+, Opera Mobile 11.10 and IE 10 have the most complete support for Multicolumn layout module, while partial support can be found (via prefixes) for Firefox 3.6+, Safari 3.2+, Chrome 10+

~~~ End of Article ~~



## Session 9: Creating Tables

### Tables

|                          |                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source</b>            | <a href="https://p2pu.org/en/groups/html-introduction/content/tables/">https://p2pu.org/en/groups/html-introduction/content/tables/</a> |
| <b>Date of Retrieval</b> | 31/7/2012                                                                                                                               |

There are times when we wish to display data that is sorted into columns and rows. This is called tabular data and HTML has a **<table>** tag and sub elements for organizing such data. For example, a vegetable comparison table might consist of vegetable names and their corresponding color or nutritional value. Lets construct a simple table.

To begin, we open the `<table>` tag:

```
<table>
```

Immediately after `<table>` tag, we can include a `<caption>` tag to quickly inform the reader the intent of the table.

```
<caption>A table of vegetable colors.</caption>
```

Tables consist of rows and columns. The **table row <tr>** tag is used to separate table rows, while **table data <td>** fits within each row.

Also note, the first row of this table will be reserved for **table headers <th>**. Headers let a reader know what each column, or row, of a table represents.

```
<tr>
  <th>Name of Vegetable</th>
  <th>Color</th>
</tr>
```

Additional table rows `<tr>` and table data `<td>` will follow the same format as the table headers `<th>`, i.e. they will be typed in the same order.

```
<tr>
  <td>Carrot</td>
  <td>Orange</td>
</tr>
<tr>
  <td>Asparagus</td>
  <td>Green</td>
</tr>
```

Lastly, we will close the table tag.

```
</table>
```

~~~ End of Article ~~~



Session 10: HTML Forms

Multiple Submit Buttons on HTML Forms: Alternate Methods

Source http://911-need-code-help.blogspot.in/2008/10/multiple-submit-buttons-on-html-forms_27.html

HTML forms often need to present a choice between two or more operations on the same data. Here is an example:

```
[x] User 1
[ ] User 2
[x] User 3

[De-activate selected user(s)] [Permanently delete selected user(s)]
```

In this article, I will outline few examples to accomplish this. All of the following examples refer to a field called "command" which specifies the operation to perform. Assume that the "verbosity" on the captions is required.

Method 1: Use Multiple Submit Buttons

`<input type="submit">` tag uses the value attribute for both the caption and the posted value. You can therefore use two submit buttons to specify the two operations:

```
<form name="form1" method="post" action="">

  <input type="checkbox" name="UserID" value="1"> User 1<br>
  <input type="checkbox" name="UserID" value="2"> User 2<br>
  <input type="checkbox" name="UserID" value="3"> User 3<br>
  <br>
  <input type="submit" name="command" value="De-active selected user(s)">
  <input type="submit" name="command" value="Permanently delete selected user(s)">

</form>
```

```
☐ User 1
☐ User 2
☐ User 3
```

```
De-active selected user(s)  Permanently delete selected user(s)
```

When a form contains more than one submit button with the same "name", the value submitted by the browser is the value of the submit button that was clicked. The server will therefore receive one of the following values for the "command" field:

- De-activate selected user(s)
- Permanently delete selected user(s)

In order to process the form, you can use string matching and hard code the two values in the server side code. The resulting code would be similar to:

```
<%  
  
select case Request.Form( "command" )  
  
    case "De-activate selected user(s)"  
  
        ' process the command  
  
    case "Permanently delete selected user(s)"  
  
        ' process the command  
  
end select  
  
%>
```

While this would work, some of you will not find this method very "elegant". You will have to make sure that the captions on the buttons and the strings in the server side code always match. If you are frequently asked to revise the captions, e.g. change "De-activate" to "Deactivate", "Disable" etc., chances are that you will make a mistake eventually. For example, you may forget to match the case of words or mistakenly add additional spaces in the server side code. Some coders also like to see and use generic, shortened names for commands, such as user_disable, user_delete. While perfectly acceptable inside the code, these strings make non-sense captions.

Method 2: Use Multiple Radio Buttons and a Submit Button

You can use radio buttons to specify the two operations. The caption can be displayed separately (preferably using the `<label>` tag):

```
<form name="form2" method="post" action="">  
  
    <input type="checkbox" name="UserID" value="1"> User 1<br>  
  
    <input type="checkbox" name="UserID" value="2"> User 2<br>  
  
    <input type="checkbox" name="UserID" value="3"> User 3<br>  
  
    <br>  
  
    <input type="radio" name="command" value="user_disable"> De-activate selected user(s)  
  
    <input type="radio" name="command" value="user_delete"> Permanently delete selected  
    user(s)<br>  
  
    <br>
```

```
<input type="submit" value="Submit">
```

```
</form>
```

☐ User 1
☐ User 2
☐ User 3

☐ De-activate selected user(s)
☐ Permanently delete selected user(s)

Submit

The server will receive one of the following values for the "command" field, depending on the radio button that was checked:

- user_disable
- user_delete

The values are easier to code. This method, however, requires the user to click a radio button before clicking the "Submit" button. This method is therefore not very intuitive but useful in some cases.

In order to process the form, you can use a script similar to the above one, with minor changes:

```
<%
select case Request.Form( "command" )
case "user_disable"
' process the command
case "user_delete"
' process the command
end select
%>
```

Method 3: Use Multiple Buttons and JavaScript

You can use buttons along with JavaScript as in the following example:

```
<form name="form3" method="post" action="">
<input type="checkbox" name="UserID" value="1"> User 1<br>
<input type="checkbox" name="UserID" value="2"> User 2<br>
<input type="checkbox" name="UserID" value="3"> User 3<br>
<br>
<input type="hidden" name="command" value="">
```



```
<input type="button" onclick="document.form3.command.value = 'user_disable';
document.form3.submit( );" value="De-active selected user(s)">

<input type="button" onclick="document.form3.command.value = 'user_delete';
document.form3.submit( );" value="Permanently delete selected user(s)">

</form>
```

- ☐ User 1
- ☐ User 2
- ☐ User 3

De-active selected user(s)

Permanently delete selected user(s)

The buttons created with `<input type="button">` are usually used as triggers for client-side scripts. Here, we use two such buttons to modify the value of the "command" field (hidden intentionally) and submit the form. This is done by embedding JavaScript code in the "onclick" event of the buttons. The server will receive one of the following values for the "command" field:

- user_disable
- user_delete

This method, however, requires a JavaScript capable browser having JavaScript enabled. If this is not the case then the buttons simply will not work.

In order to process the form, you can the script similar to the above one:

```
<%
select case Request.Form( "command" )
case "user_disable"
' process the command
case "user_delete"
' process the command
end select
%>
```

Method 4: Use Multiple Submit Buttons and CSS

As described in method 1, `<input type="submit">` tag uses the value attribute for both the caption and the posted value. You can therefore use two submit buttons to specify the two operations. This variation of the example uses captions that make more sense to the server side code but not to the user:

```
<form name="form4" method="post" action="">

<input type="checkbox" name="UserID" value="1"> User 1<br>

<input type="checkbox" name="UserID" value="2"> User 2<br>

<input type="checkbox" name="UserID" value="3"> User 3<br>
```

```

<br>

<input type="submit" name="command" value="user_disable" class="submit-button button-
disable">

<input type="submit" name="command" value="user_delete" class="submit-button button-
delete">

</form>

```

- ☐ User 1
- ☐ User 2
- ☐ User 3

The server will therefore receive one of the following values for the "command" field, depending on which button was clicked to submit the form:

- user_disable
- user_delete

It is perfect except for the captions. We can add a little CSS to hide the captions and spice things up a little:

```

<style type="text/css">

.submit-button {

    margin: 0;

    border: 0;

    padding: 0;

    width: 20px;

    height: 20px;

    text-indent: -1000px; /* important part -- the css text-indent property
                           this rule attempts to push the caption outside
                           the button, effectively hiding it */

    background-color: #CCCCCC;

    background-position: center center;

    background-repeat: no-repeat;

}

.button-disable {

    background-image: url(icon-disable.png);

}

```

```
.button-delete {
    background-image: url(icon-delete.png);
}
</style>
```

And the result is:

☐ User 1
☐ User 2
☐ User 3






Variations of Method 4

The above mentioned example is suitable for use inside data grids. Following example shows a data grid that contains multiple rows of data, each with a delete button containing the ID of the record:

```
<form name="form5" method="post" action="">
    <input type="hidden" name="command" value="user_delete">
    <table>
        <tr>
            <td>User 1</td>
            <td>... data ...</td>
            <td><input type="submit" name="UserID" value="1" class="submit-button button-
delete"></td>
        </tr>
        <tr>
            <td>User 2</td>
            <td>... data ...</td>
            <td><input type="submit" name="UserID" value="2" class="submit-button button-
delete"></td>
        </tr>
        <tr>
            <td>User 3</td>
            <td>... data ...</td>
```

```
<td><input type="submit" name="UserID" value="3" class="submit-button button-  
delete"></td>  
  
</tr>  
  
</table>  
  
</form>
```

And the result is:

| | | |
|--------|--------------|-----------------------------------------------------------------------------------|
| User 1 | ... data ... |  |
| User 2 | ... data ... |  |
| User 3 | ... data ... |  |

~~~ End of Article ~~~



## Customize HTML5 form validation with JavaScript and CSS

**Source** <http://www.codeproject.com/Tips/401908/Customize-HTML5-form-validation-with-JavaScript-an>

### Introduction

With **HTML5**, to check **forms** validity, new **input type** values (email, URL, etc.) and also the `pattern` attribute were introduced, but the control is done when the form is submitted.

Is there a way to do this before that?

### JavaScript

There are many ways to personalize control validation, not only the message of error with `setCustomValidity`, but also calling the `checkValidity()` function to check inputs values, on some input event as `onchange`.

Here the JavaScript code of example:

```
window.onload=function() {
    var query = ["input[pattern]", "input[type=email]"];
    for(var q = 0; q < query.length; q++) {
        var inp = document.querySelectorAll(query[q]);
        for(var i = 0; i < inp.length; i++) {
            inp[i].onkeyup = validation;
        }
    }
};

function validation(event) {
    var p = this.parentNode.querySelector("p");
    if (this.checkValidity() == false) { //the control is here!
        p.innerHTML=this.getAttribute("title");
    } else {
        p.innerHTML="";
    }
}
```

And this is the HTML code:

```
<form>
  <fieldset>
    <legend>Form</legend>
    <div>
      <label for="input01">Name</label>
      <div>
        <input type="text" pattern="[a-zA-Z ]+"
          required title="Insert only letters of the alphabet" id="input01"
name="name">
        <p class="err-msg"></p>
      </div>
    </div>
  </div>
```

```

        <label for="input02">Pin</label>
        <div >
            <input type="text" pattern="[a-zA-z0-9]{16}"
                required title="Insert 16 chars" maxlength="16" id="input02"
name="pin">
            <p class="err-msg"></p>
        </div>
    </div>
    <div>
        <label for="input03">Cellphone</label>
        <div>
            <input type="text" pattern="[0-9]+"
                required title="Insert only numbers chars" id="input03" name="cell"
>
            <p class="err-msg"></p>
        </div>
    </div>
    <div>
        <label for="input04">Email</label>
        <div >
            <input type="email" required title="Insert a valid email"
id="input04" name="email">
            <p class="err-msg"></p>
        </div>
    </div>

    <div style="width:100%; text-align:center;">
        <input type="submit">
        <input type="reset" >
    </div>
</fieldset>
</form>

```

**Take a look:**

- The `title` attribute value is used by browsers to personalize the error message for an input.
- The `required` attribute specifies that an input field must be filled out before submitting the form.

**CSS**

You can personalize also the inputs style with pseudo-classes namely

[:invalid](#), [:valid](#), [:required](#).

Other pseudo-classes can be found here: [CSS Basic User Interface Module Level 3 \(CSS3 UI\)](#)

For example, with the following CSS rule, input texts will be red until the input is invalid.

```

input:invalid {
    color:red;
}

```

~~~ End of Article ~~~



Fun with HTML5 Forms

Source<http://www.boogdesign.com/b2evo/index.php/fun-with-html5-forms?blog=2>

There have been many blog posts describing all the new elements and input types HTML5 provides check out this 24ways post for a good summary, or you could even read chapter 3 of my book. In this post I'm going to focus instead on the validation API and some related HTML5 features by building a simple game based on entering an email address. The goal is to explore some HTML5 features rather than do everything in the most straightforward way, but there should be some practically useful code snippets.

The form itself is going to be very straightforward:

```
<form id="game">
  <fieldset>
    <legend>Enter a valid email address before the timer runs
down</legend>
    <label for="email">Email</label>
    <input id="email" type="email" autofocus required>
  </fieldset>
  <label for="countdown">You have
    <output id="countdown">10</output>
    seconds.
  </label>
  <label for="score">You have
    <output id="score">0</output>
    points.
  </label>
</form>
```

In the initial version we will take advantage of three HTML5 features:

- The `output` element
- The `email` input type
- The `checkValidity` method from the HTML5 Form Validation API

In the countdown function we use the `value` property of the `output` element. The `output` element is something like a `span`, it's a start and end tag with arbitrary content, but you can access the contents using the `value` attribute like a form field:

```
var counter = function (cd, sc, em) {
  cd.value -= 1;
  sc.value = calcScore(em.value);
}
```

This may not look particularly helpful, but consider what the code would look like without the `output` element (or in a browser which doesn't support `output`):

```
var counter = function (cd, sc, em) {  
    var count = cd.innerHTML;  
    count -= 1;  
    cd.innerHTML = count;  
    sc.innerHTML = calcScore(em.innerHTML);  
}
```

The function is now mostly about the details of manipulating the DOM, rather than the more straightforward algebraic style of the original. Granted, libraries like jQuery have for a long time provided abstractions which allow you to write this code more cleanly but, like design patterns, these are as much an indication of the lack of expressiveness in the underlying platform as they are examples of best practice.

The game needs a function to calculate a 'score' from the email address. This is, of course, completely arbitrary:

```
function calcScore(email) {  
    var s = 0;  
    s += email.length;  
    s += email.indexOf('@') > -1 ? email.indexOf('@') : 0;  
    s += email.lastIndexOf('.') > -1 ? email.lastIndexOf('.') : 0;  
    return s;  
}
```

Now we need to wire up the functions to appropriate events when the game starts. To keep things simple I'm going to store a reference to the interval in a global variable:

```
var cdInterval;  
var gameStart = function () {  
    window.clearInterval(cdInterval);  
    var em = document.getElementById('email');  
    var cd = document.getElementById('countdown');  
    var sc = document.getElementById('score');  
    cd.value = 10;  
    sc.value = 0;  
    em.value = '';  
    em.readOnly = false;  
    cdInterval = window.setInterval(counter, 1000, cd, sc, em);  
    window.setTimeout(gameOver, 10000, cd, sc, em);  
}
```

Again, this function takes advantage of the value attribute on the output elements. The last line calls a `gameOver` function after ten seconds, we'll use that to clear the interval and calculate the score. The `checkValidity` method of the email field will be used to determine if the email address is currently valid:

```
var gameOver = function (cd, sc, em) {  
    window.clearInterval(cdInterval);
```



```

    var score = calcScore(em.value);
    if (!em.checkValidity()) {
        score = 0;
        window.alert("You lose!");
    }
    cd.value = 0;
    sc.value = score;
    em.readOnly = true;
}

```

The `checkValidity()` method is part of the HTML5 validation API, it returns `true` if the browser understands the contents of the field as a valid email. Note that we are able to call this method without submitting the form, so it is easy to hook form submission into a custom validation function if we want. At no point do we have to implement any code to determine what a valid email address would be. This is a second major saving, have a look at this regular expression example if you thought that code was straightforward:

```

(?:[a-z0-9!#$%&'*/+=?^_`{|}~-]+(?:\.(?:[a-z0-9!#$%&'*/+=?^_`{|}~-]+)|))
+)*|"(?:[x01-x08\x0b\x0c\x0e-x1f\x21\x23-x5b\x5d-x7f]|\\[x01-
\x09\x0b\x0c\x0e-x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-
9])?)\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?)|\\[(:?(:25[0-5]|2[0-4][0-
9]|01)?[0-9][0-9]?)\.){3}(:25[0-5]|2[0-4][0-9]|01)?[0-9][0-
9]?|[a-z0-9-]*[a-z0-9]:(?:[x01-x08\x0b\x0c\x0e-x1f\x21-x5a\x53-
\x7f]|\\[x01-x09\x0b\x0c\x0e-x7f])+)\.))

```

Finally, start the game when the page loads:

```

window.addEventListener('load', gameStart, false);

```

The first version of the game will work in Opera, and the development versions of Firefox (4 - relevant bugs are 345624 and 346485) and Chrome (9 - relevant bugs are 27452 and 29363). If you want it to work on older versions of Chrome and Safari then you'll need to replace the output based syntax with the `innerHTML` approach as per the alternate version of `counter` above.

For the second iteration we're going to look at these two features:

- Using the `pageshow` event from the History API
- How to reset `output` elements

If the user visits another page after completing the game and then goes back, then the game will sit there inert until you reload the page. This is because the game logic hangs off the `onload` event, and navigating forward and backwards in the browser history doesn't fire `onload`. The HTML5 History API gives us another option: the `onpageshow` event. This fires whenever a page is displayed rather than when it is loaded. There is currently support

for `onpageshow` in Firefox and Chrome, but not in Opera, so we'll use the `iseventsupported` script to detect support and fall back to `onload` if it's not available:

```
if (isEventSupported('pageshow')) {  
    window.addEventListener('pageshow', gameStart, false);  
} else {  
    window.addEventListener('load', gameStart, false);  
}
```

We'll also do some extra work in the `gameStart` function to clear and reinstate the timeout and, since the `autofocus` attribute gets ignored in `onpageshow`, focus the input field:

```
var gameStart = function () {  
    window.clearInterval(cdInterval);  
    window.clearTimeout(cdTimeout);  
    var em = document.getElementById('email');  
    var cd = document.getElementById('countdown');  
    var sc = document.getElementById('score');  
    cd.value = 10;  
    sc.value = 0;  
    em.value = '';  
    em.readOnly = false;  
    cdInterval = window.setInterval(counter,1000, cd, sc, em);  
    cdTimeout = window.setTimeout(gameOver,10000, cd, sc, em);  
    em.focus();  
}
```

Missing from the first version of the game is any way to restart or replay. Let's start off by adding a reset button to the form:

```
<input id="restart" type="reset" value="New Game">
```

There are some obvious issues with this, not least that the form reset isn't going to remove the countdown and game over timeouts. However, rather than deal with that by hooking the event in a sensible way, we're going to take this opportunity to investigate an interesting feature of the `output` element, the `defaultValue`.

If you try clicking the reset button as it stands, this is what happens to the two output elements:

You have seconds.

You have points.

The output element is something of a hybrid. Although it can be created, declaratively, in markup, it isn't really any use without JavaScript. While not a direct consequence of this state of affairs, it therefore doesn't matter that there's no way to set the default value of an

output element in the HTML, you have to do it with JavaScript using the `defaultValue` property in the DOM. Let's add a function to set up these fields correctly which will be run on `onload`:

```
var gameSetup = function() {  
    document.getElementById('countdown').defaultValue = "10";  
    document.getElementById('score').defaultValue = "0";  
    document.getElementById('restart').addEventListener('click',  
gameStart, false);  
}
```

You'll note that there's also an `onclick` handler added to the button to restart the game. The reason we don't want to capture the form reset event itself is that we're going to use the form reset method within `gameStart`. These three lines of code:

```
cd.value = 10;  
sc.value = 0;  
em.value = '';
```

Will become this one line of code:

```
document.getElementById('game').reset();
```

But now resetting the form will restore the output fields to their `defaultValue`, as can be seen in the second checkpoint:

You have 10 seconds.

You have 0 points.

For the third iteration we're going to concentrate on user feedback. This is going to involve both HTML5 and CSS3:

- Using the `onforminput` event to update the score immediately
- Giving instant feedback on the current validity of the email with CSS3

So far feedback to the user happens every second, in the counter function. Instead of updating the score every second let's update it every time the input changes. There are a number of options in HTML4 already for handling this - we could attach an event to the email field and monitor it for changes, however HTML5 (for now) presents us with another option: `onforminput`. Instead of being associated with the input field, the item being updated, `onforminput` can be associated with the output element, the item we want to update:

```
document.getElementById('score').addEventListener('forminput',  
updateScore, false);
```

Because the output element is the target, the update score function looks like this:

```
var updateScore = function(ev) {  
    ev.target.value =  
    calcScore(document.getElementById('email').value);  
}
```

Unfortunately it seems that `onforminput` may be dropped from HTML5, so we'd better check for support and fall back to capturing any `oninput` event on the whole form through event bubbling:

```
if (isEventSupported('forminput'))  
{  
    document.getElementById('score').addEventListener('forminput',  
    updateScore, false);  
}  
else  
{  
    document.getElementById('game').addEventListener('input',  
    updateGlobalScore, false);  
}
```

This does complicate the function to update the score as the target of the event is no longer the output element, it's whatever input element has been updated. At least in our simple case there's only one input element so we can use the `ev.target` shortcut, just on the other side of the assignment:

```
var updateGlobalScore = function(ev) {  
    document.getElementById('score').value =  
    calcScore(ev.target.value);  
}
```

If you've been following along in Firefox you will have seen that both provide some default styling for invalid fields:



However Chrome (top) and Opera (bottom) don't show any indication:



Let's apply some styles explicitly to make everything consistent across browsers:

```
input[type=email]:invalid {  
    box-shadow: none;  
    background-color: rgba(255,0,0,0.5);  
}  
input[type=email]:valid {  
    box-shadow: none;  
    background-color: rgba(0,255,0,0.5);  
}
```

The `box-shadow` overrides the Firefox default, while the `background-color` will be applied in all three browsers, as you can see in the third checkpoint.

The game is now functionally complete, but before we wrap this up I'd like to use one more iteration to examine the `checkValidity` method more closely. Here's what the spec says about `checkValidity`:

Returns true if the element's value has no validity problems; false otherwise. Fires an invalid event at the element in the latter case.

We only perform additional work so, instead of using the boolean return value of the function to guide or logic, we could make the end of the game event driven. First we'll simplify the `gameOver` function, as all we need to do there now is ensure the event gets triggered if the email is invalid (and, thanks to `onforminput`, we know the score will always be up to date):

```
var gameOver = function (cd, sc, em) {  
    window.clearInterval(cdInterval);  
    cd.value = 0;  
    em.checkValidity();  
}
```

Now we need to declare a function to handle the `oninvalid` event, I've chosen to name it in tribute to cheesy European rock:

```
var theFinalCountdown = function(e) {  
    document.getElementById('score').value = 0;  
    window.alert("You lose!");  
    e.target.readOnly = true;  
}
```

Then we attach it to the `oninvalid` of the email field:

```
document.getElementById('email').addEventListener('invalid',  
theFinalCountdown, false);
```

Thus concludes my slightly erratic tour through HTML5 form features with the final checkpoint. If you've enjoyed this please check out the more extensive introduction to HTML5 Forms in Chapter 3 of my book.

~~~ End of Article ~~~



Session 11: HTML5 Audio and Video

Embedding video and audio

| | |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Source | http://ebw.co/kbase/epub-production-tips/embedding-video-and-audio |
| Date of Retrieval | 31/7/2012 |

When we created the original [Moxyland](#) ebook with an embedded soundtrack (it's no longer available, long story), the best way to embed rich media in an epub was with Flash. We created Flash video of an audio controller that played an audio track. Then we simply included this in our epub code:

```
<object type="application/x-shockwave-flash" data="filename.swf"
width="400" height="20">
<param name="movie" value="filename.swf" />
<param name="play" value="false" />
<param name="loop" value="false" />
</object>
```

If you're not technically oriented, don't worry. The point is that it wasn't complex.

The problem nowadays is that (a) iOS devices (iPad, iPhone, iPod) don't support Flash, and (b) Kindle is the dominant ebook environment, and doesn't really support this kind of embedding. So, while Flash will work on many ebook systems, it doesn't on the most important ones.

One could create an iOS-optimised epub with sound/video embedded using HTML5, but only HTML5-compliant systems will be able to play the media (iOS devices and good web browsers, but not Adobe Digital Editions and many other ebook software and readers). So it would effectively be an 'iOS-only' edition in your marketing.

At the moment, we're recommending to our clients that they place the video or audio on a media-sharing site (like YouTube, Vimeo, or MixCloud), embed the media on a dedicated site, and link to that site using a regular hyperlink in the ebook.

~~~ End of Article ~~~



## Jaraoke – An HTML5 <audio> Tag Demo

|                   |                                                                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source            | <a href="http://randallagordon.com/blog/2009/07/19/jaraoke-html5-audio-tag-demo/">http://randallagordon.com/blog/2009/07/19/jaraoke-html5-audio-tag-demo/</a> |
| Date of Retrieval | 31/7/2012                                                                                                                                                     |

July 19, 2009

So Firefox 3.5 recently hit the streets and [brought some new toys](#) for we designers and developers, including support for HTML5's <audio> and <video> tags. The Ogg Vorbis and Theora codecs are supported for audio and video respectively. I tinkered with them a little bit with an earlier beta, but didn't get too fancy. Now that 3.5 is available and more folks have it, I've put together a little demo I've been thinking about for a couple months.

Presenting, you're going to hate me for this, **karaoke in Javascript!** Or, as I like to call it, Jaraoke! Cheeky, eh?

Head over to <http://randallagordon.com/jaraoke/> and check it out. Just hit play, and then you're able to toggle the lyrics in the audio on and off. The song is "Discipline" by Nine Inch Nails, chosen because of the great forward-thinking license chosen for the song, a Creative Commons Attribution-Noncommercial Share Alike license. The lyrics start 24 seconds in, so be patient.

### Other Browsers

It is also worth noting that this works in Safari 4 using MP3 sources instead of Ogg Vorbis. Unfortunately the codec playing field is not a level one and Apple refuses to support Ogg on grounds that a patent could crop up out of nowhere...

### The Markup

The guts are really quite simple. I'm loading up two separate audio files to be played simultaneously, one with the lyrics present and one which is instrumental. Here you'll also see the two separate source declarations, one for Ogg Vorbis and another for MP3. Beyond that there are two links, one for play control and one to toggle between the vocal and instrumental tracks, and an unordered list element to insert the lyrics into. Here's the markup:

1. `<audio id='apN' autobuffer="true">`
2.     `<source src='audio/discipline_mixdown-nvocal.ogg' type='audio/ogg; codecs="vorbis"'>`
3.     `<source src='audio/discipline_mixdown-nvocal.mp3' type='audio/mp3; codecs="mp3"'>`
4.     A custom message can go here for browsers which don't support the `<code>audio</code> HTML5 element.`
5. `</audio>`
6. `<audio id='apW' autobuffer="true">`
7.     `<source src='audio/discipline_mixdown-wvocal.ogg' type='audio/ogg; codecs="vorbis"'>`



```

8.      <source src='audio/discipline_mixdown-
        wvocal.mp3' type='audio/mp3; codecs="mp3"'>
9.    </audio>
10.
11.    <a id="togglePlay" href="#">Play/Pause</a>
12.    <a id="toggleVocal" href="#">Toggle Vocals</a>
13.
14.    <ul id="lyricsList"></ul>

```

### The JavaScript

When the DOM is loaded, the Javascript does some minimal setup. The volume for the vocal track is set to 0 and the instrumental to 1. These settings are floats with a value of 0, volume completely off, to 1, volume completely on. This, of course, let's the two files play simultaneously without interfering. By calling play and pause during setup, it forces the files to play perfectly synchronized (at least on my system...) so there is no "skipping" when the vocals are toggled. The lyrics and cue times are simply hard coded into arrays and really should be imported from an XML file.

(UPDATE: Newer versions of Chrome no longer play the files in sync; I've yet to find another solution. If you have a solution, please leave a comment! Thanks!)

The grunt work is done by setting up an event listener on the audio element's "timeupdate" event. The callback just checks the current playback time against a set of cue times. If the playback time is past the current cue's time then it displays the current lyric and increments to the next cue. Here's the relevant code:

```

1.  var currCue = 0;
2.  var cues = [ 24.78, 26.81, 29.69, 34.88, ... ];
3.  var lyrics = [ "Am I...",
4.                "Am I still tough enough?",
5.                "Feels like I'm wearing down, down, down, down, down.",
6.                "..",
7.                "Is my viciousness", ... ];
8.
9.  this.apN.addEventListener("timeupdate", timeUpdate, true);
10. function timeUpdate() {
11.     if(jaraoke.apN.currentTime > (cues[currCue] - CUEOFFSET)) {
12.         var el = $(document.createElement('li'));
13.         el.update(lyrics[currCue++]);
14.         $('lyricsList').insertBefore(el, $('lyricsList').firstChild);
15.     }
16. }

```

~~~ End of Article ~~~



Session 12: Introduction to JavaScript

3 Simple Things to Make Your jQuery Code Awesome

| | |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source | http://www.joezimjs.com/javascript/3-simple-things-to-make-your-jquery-code-awesome/ |
| Date of Retrieval | 31/7/2012 |

jQuery is one of the most popular (if not the most) JavaScript libraries in existence and a great number of people use it to do some amazing things. Personally, jQuery is what got me excited for learning JavaScript. The problem is that a lot of programmers don't understand that with all that power massive amounts of CPU cycles are used. As much as jQuery engineers try to optimize jQuery, they are always limited in how fast they can make it go. There are a few things that you, as a user of jQuery, can do to make your jQuery code faster.

1 – jQuery Object Caching

Caching your jQuery objects may possibly be the best thing you can do to cut your code down to run leaner and meaner. If you use the same selector more than once in your code, or if you query the DOM inside a loop or in a function that is run more than once, then you can probably benefit a lot from caching your queries inside a variable. Consider the following 3 examples:

```
// Loop
for (var i=0; i<100; i++) {
    $('ul.special').append('<li>'+i+'</li>');
}

// Multiple Functions that each have the
// chance to be called multiple times
$('#showbutton').on('click', function() {
    $('#box').show();
});
$('#hidebutton').on('click', function() {
    $('#box').hide();
});

// Just re-querying
$('p').width(150);
$('p').css('color', 'red');
$('p').addClass('awesome');
```

In all of these cases, you could save some DOM searching trips by assigning the jQuery object to a variable (generally prefixed with a dollar sign to distinguish it as a jQuery object), like so:

```
var $ul = $('ul.special');
for (var i=0; i<100; i++) {
    $ul.append('<li>'+i+'</li>');
}

var $box = $('#box');
$('#showbutton').on('click', function() {
    $box.show();
});
$('#hidebutton').on('click', function() {
    $box.hide();
});
```

```
$('p').width(150).css('color', 'red').addClass('awesome');
```

One of the most expensive operations you can do is query the DOM, especially in older browsers that can't be optimized for with built-in functions. Each time you query the DOM (with a few exceptions), you have to search through the entire DOM to find each and every matching element, which can take time, especially on large documents. The third example actually uses chaining, which is similar to caching because you still optimize down to one DOM search, but it doesn't require you to save the jQuery object to a variable.

2 – Selector Optimization

The CSS selectors used in your DOM queries can sometimes make more of a difference in performance than the lack of caching the results of that search. The first thing you have to realize is that the selectors are read from right to left, so you always want your most specific selectors (most notably id's) to be as far to the right as possible. Many times, though, you are trying to find the children of an element with an id, therefore you can't have the id selector furthest to the right in the full selector. There is a way around this though, via context or using find or children:

```
// Instead of this:
$('#id p');

// Try one of these:
$('p', '#id');
$('#id').find('p');
$('#id').children('p')
```

The following selectors are ranked from fastest to slowest. Always try to have a faster selector further to the right or within the context parameter/find/children to make your selections as fast as possible.

1) `$('#id');`

An id selector is easily the fastest selector. There are two reasons for this: 1) There is only ever one element with an id, so once it is found, the searching stops and 2) browsers have a built in function for searching for elements by their id (`document.getElementById()`), and built-in functions run much quicker than hand-made functions.

2) `$('tag');`

Searching by a tag name is somewhat fast only because all browsers support the built-in function called `document.getElementsByTagName()`.

3) `$('.class');`

Searching via class would probably be comparable to searching by the tag name, but you have to be careful just because IE8 and below don't support the `natedocument.getElementsByClassName()`.

4) `$('[attribute]');` or `$('[attribute=value]');` or `$(':pseudo');`

No browser currently has a native function available to JavaScript for searching with these selectors, so jQuery is required to crawl through the DOM by itself and check each element to see if it matches this selector. There are some modern browsers that have `document.querySelectorAll()`, which is a native function that can take any selector, but even with the increased performance from this function, the look-ups for these selectors is still quite slow.

3 – Delegating Events

The third and final optimization involves events. If you are attaching an event handler to each and every cell of a table, you could be using a lot more memory than you really need to, plus it takes a little time to apply a handler to each of those cells. This might be done something like this:

```
$('#table td').on('click', function() {  
    // Do Something  
});
```

Event delegation allows us to attach a single event handler to a parent element – saving us memory and set up time – that only fires when the event is triggered on specific child elements. So, using the example above, we could attach a single handler to the table that would fire any time someone clicked on a td, but not when someone clicks on a th or other element within the table. The code to do this looks like this:

```
$('#table').on('click', 'td', function() {  
    // Do Something  
});
```

Notice, that the selector for the elements that you actually want the event to trigger on is now the second argument to the on function. The cool thing is that this still refers to the td that was clicked and not to the table, just like it would if you had attached the handler directly to the cells. This also has the added benefit that if more table cells are dynamically added in, you don't need to add the event handler to them because they are already covered by the parent element.

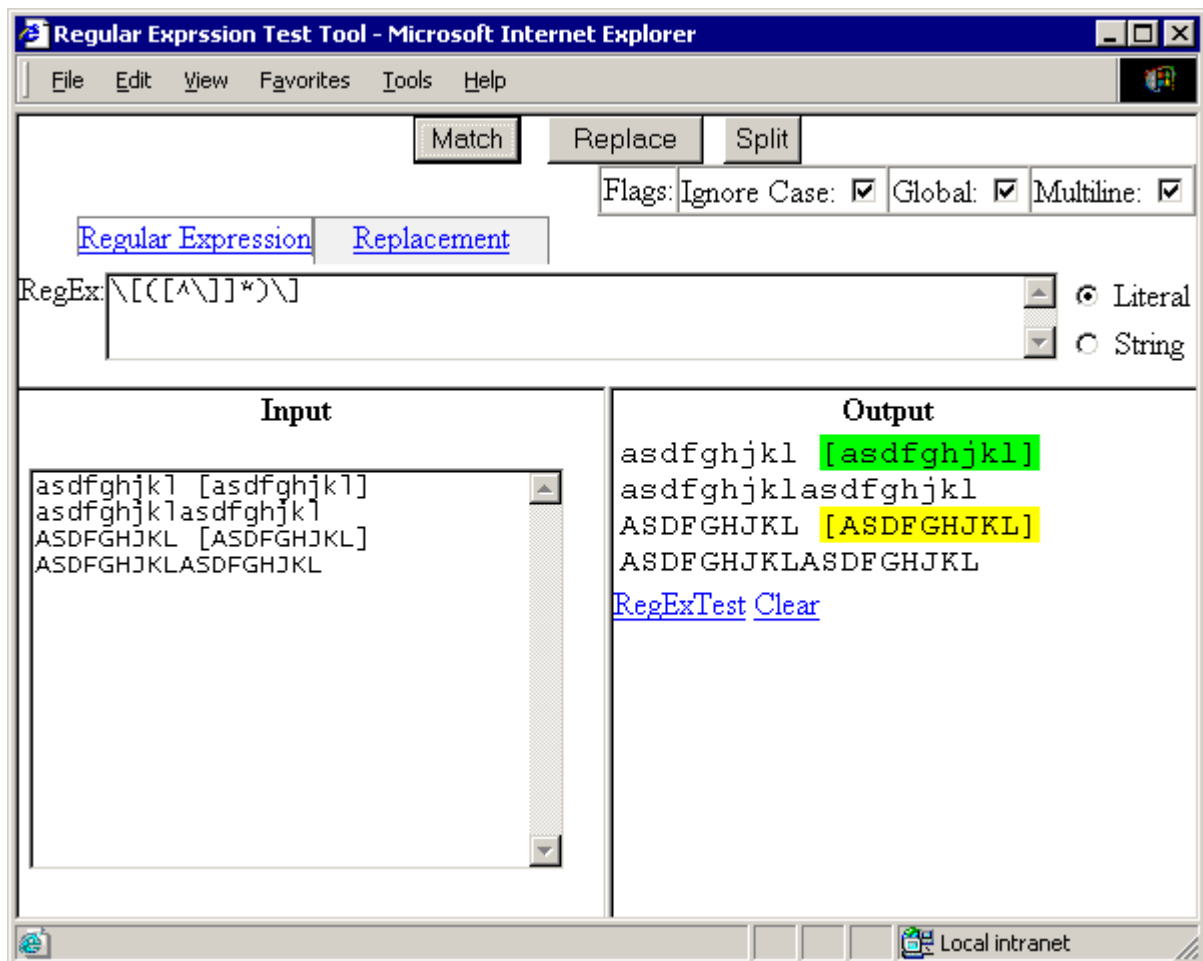
~~~ End of Article ~~~



## Session 13: Operators and Statements

### JavaScript Regular Expression Tester

**Source** <http://www.codeproject.com/Articles/8601/JavaScript-Regular-Expression-Tester>



- Download source files - 8.06 Kb

### Introduction

This handy utility program tests JavaScript Regular Expressions in a browser using JavaScript. Its interface is similar to other regular expression test tools, but unlike those tools, it tests JavaScript's implementation of Regular Expression in JavaScript.

### Background

I often find myself writing JavaScript code for both the client side and server side that uses Regular Expressions for parsing text or validating input. However, as Regular Expressions can often get very complex, I needed a reliable tool to be able to test the expressions that I would write to make sure that I got the expected results. While there

were many Regular Expression test tools, none were written specifically to test Regular Expressions as implemented in JavaScript. So I copied what I liked from the interfaces of some of these tools and wrote my own in HTML and JavaScript. This makes it perfect for testing JavaScript Regular Expressions as it's written with JavaScript and executed in JavaScript.

I share this code for several reasons. First is to let other people find it and use it. I think that it is useful, and I think that you may find it useful as well. I have used this with very large blocks of text that is greater than one megabyte, and found it to be reasonably fast for a test environment.

I also want to hear feedback about it, what is useful, what is good, and which parts are worse but only if you tell how you think it could be improved. As I am not currently writing many complex regular expressions any more, I don't have the reason to use this tool as hard. I would like to hear from people who can, and are, using it.

## ***Using the code***

Type the regular expression at the top and the text on which the regular expression is to work in the bottom left. Press the "match" button.

Other options include replacing the matched text (use \$1, \$2 etc. for back references).

There are a few options which JavaScript allows one to use, and I have exposed as much of these as I could. The regular expression can be written in text format, i.e., `re = new RegExp("ab+c")` or as a literal, i.e., `re = /ab+c/`, so I allow for both. There are also flags for "Ignore Case", "Global", and "Multiline" supported by regular expressions in JavaScript, and they are exposed as the checkboxes at the top right of the interface. Click on the "Replacement String" to edit what will be replaced when using Replace. Clicking on "clear" clears the output, and clicking on "RegExTest" shows the current content of the RegExTest object in JavaScript Object Notation (JSON), or what some people may call an "object literal".

## ***The technical details***

All of the data is stored in a JavaScript object called "RegExTest". There are three functions that work on this object, one each for the three top buttons. These are functions and not methods because I wanted the data to be exportable (via JSON) so that it could be saved externally (as a file, in a database, etc.) for me to work on later, but this import/export is a topic for its own article.

This interface uses frames, and so each frame needs to be able to communicate what its "state" is to the RegExTest object so that the program can work. In this end, I have delegated all authority to the top frame. The other frames are merely places to show the data to the user. That is, as little functionality as possible is built into each frame. The bulk of the functionality is actually at the top frame. This was done in part to make it easier to allow for import/export via JSON as well as to simplify the amount of code that would need to be in any frame.

To get the data from the frames into the object, every event in each form element copies the data to the top level RegExTest object when the item changes using the `CopyUp()`. There is also a `CopyDown()` function in each page that copies the data from the RegExTest object and places the data into the forms in each frame. This is an example:

```
function copyDown() {
    document.local.TheText.value = window.top.RegExTest.theText;
};
function copyUp() {
    window.top.RegExTest.theText = document.local.TheText.value;
};
```

You can think of the `CopyUp()` and `CopyDown()` functions essentially as methods of every frame. This makes the following code possible, and the `top.CopyUp()` is used before executing a match, replacement or split, and before saving the `RegExTest` object.

```
// Walk through all of the frames starting at the very top level, which this is
function CopyUp() {
    if (top.frames.length) {
        CopyUpFrames(top.frames);
    };
};

// Walks through all of the frames starting at the very top level
function CopyUpFrames(theFrame) {
    for (i=0;i<theFrame.length;i++)
        // Call copyUp() in each page of the frameset
        if (theFrame[i].copyUp) {
            theFrame[i].copyUp();
        };
    // if the page contains frames, walk through those too
    if (theFrame[i].frames.length) {
        CopyUpFrames(theFrame[i].frames);
    };
};
```

The `CopyDown()` function is called after a match, replace or split is done, and would also be called after loading the `RegExTest` object from the disk or from a database record.

## File Contents

| File                             | Contents                                                                                                                                                     |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>regex2.htm</i>                | Parent Frame, <code>RegExTest</code> object                                                                                                                  |
| <i>regex2.js</i>                 | Core functionality in this script file, namely the functions for performing a match, replace and split                                                       |
| <i>support/support.js</i>        | Top level <code>CopyUp()</code> and <code>CopyDown()</code> functions as well as the <code>CopyUpFrames()</code> and <code>CopyDownFrames()</code> functions |
| <i>support/flags.htm</i>         | Buttons to perform a match, replace and split, as well as the checkboxes for the "Ignore Case", "Global", and "Multiline" flags.                             |
| <i>support/RegExInput.htm</i>    | The text box to display and edit the regular expression (string or literal) to be executed                                                                   |
| <i>support/ReplaceString.htm</i> | The text box to display and edit the replacement text to be used when performing a replace                                                                   |

| File                      | Contents                                                                                                         |
|---------------------------|------------------------------------------------------------------------------------------------------------------|
| <i>support/input.htm</i>  | The text box to display and edit the text on which the regular expression will perform its work                  |
| <i>support/output.htm</i> | The area to display what happened after executing the regular expression on the text                             |
| <i>support/JSON.js</i>    | Contains the functions that turn an object into a string that can be exported and later re-imported quite easily |

## ***Naughty or Nice***

I know that many people don't like `Eval()`, but here is a time when I think that it needs to be used. The `Eval()` function was used as a shortcut to avoid having to unparse JavaScript string encoded regular expressions. Literal regular expressions don't need to be unparsed as they are already strings inside JavaScript. I also found it was necessary to use the `eval()` function to support the replace feature. Please show me if you can get this to work without `Eval()` without also adding a lot of complexity.

## ***Still to be done***

Replacement with strings, even with back references, works great. But I have not yet implemented the feature to do a replacement with a function, so it is currently a disabled feature in the interface, and is otherwise ignored.

~~~ End of Article ~~~



EMAIL VALIDATION USING REGULAR EXPRESSIONS

| | |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source | http://www.williammalone.com/articles/email-validation-regular-expressions/ |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Finding patterns in strings can be a common occurrence in interface design. Ask someone to enter their email or physical address and who knows what you will get. Validating this data can be a time consuming and tedious task. There are only so many indexOf methods and if else statements one can handle. An alternative is using regular expressions.



What are Regular Expressions?

Regular expressions describe a pattern of characters in a string. Shortened to regExp or regEx this pattern can be used (among other things) to match content of strings.

For example if I wanted to check if someone knew my name or nickname I could use the regular expression `\will\` to match the characters in a string "william" and "willie". However that also includes nicknames I don't go by such as "willow" or "willis". We can expand our expression to match a little better by adding an "i" to the expression. But that will only handle "willow" not "willis". So we adjust the expression using a meta character similar to a conditional to: `\willi(e|am)\`.

As another example let's validate a phone number. Let's restrict our requirements it a local 7 digit US number to keep it simple. We define acceptable permutations as:

- 123-3456
- 123.3456
- 1233456

We use the expression with the "^" meta character to specify the beginning of the string and nothing before. We follow that with `/d` which will require only numerals (0-9). Add the `{3}` requires 3 characters. We are currently at: `^\d{3}\`.

We accept any character next so we add a dot, ".", which allows for any character and `{0,1}` which means we can have zero to one character, but no more. Next we add `\d{4}` to match those last four digits. To make sure there is nothing afterward we use the meta character "\$". Finally we have the expression: `^\d{3}.\{0,1\}\d{4}$\`.

A Practical Example

Let's say we have a website where users are asked to enter a valid email address to continue to the next page. We are going to keep the requirements very simple for this example, such as:

1. Must contain the @ symbol
2. Must contain a character before the @ symbol
3. Must contain a dot '.'
4. Must contain two characters between the @ symbol and dot
5. Must contain two characters after the dot

Let's try to validate with JavaScript using a few standard methods:

```
function validEmail(str) {

    // Handle Requirement 1

    var atLoc = str.indexOf("@");

    if(atLoc == -1){

        return false;

    }

    // Handle Requirement 2

    if(atLoc < 1){

        return false;

    }

    // Handle Requirement 3

    var dotLoc = str.indexOf(".");

    if(dotLoc == -1){

        return false;

    }

    // Handle Requirement 4
```

```
    if(dotLoc - atLoc < 2){  
        return false;  
    }  
  
    // Handle Requirement 5  
  
    if(str.length - atLoc < 2){  
        return false;  
    }  
  
    return true;  
}
```

Now I'll try to make this more efficient:

```
function validEmail(str) {  
  
    // Handle Requirement 1 and 2  
  
    var atLoc = str.indexOf("@");  
  
    if(atLoc < 1){  
        return false;  
    }  
  
    // Handle Requirement 3  
  
    var dotLoc = str.indexOf(".");  
  
    if(dotLoc == -1){  
        return false;  
    }  
  
    // Handle Requirement 4 and 5
```

```
    if(dotLoc - atLoc + str.length - atLoc < 4){  
        return false;  
    }  
    return true;  
}
```

Enter Email to Validate:

Now we try using a regular expression:

```
function validEmail(str) {  
  
    // Handle All Requirements 1-5  
    var regEx = /^.{1,}@.{2,}\..{2,}/;  
    return regEx.match(str);  
}
```

Enter Email to Validate:


~~~ End of Article ~~~



## Session 14: Loops and Arrays

### ***Comparing JavaScript loops performance (for, do while, for in)***

|               |                                                                                                                                                                                           |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source</b> | <a href="http://www.sixhat.net/comparing-javascript-loops-performance-for-do-while-for-in.html">http://www.sixhat.net/comparing-javascript-loops-performance-for-do-while-for-in.html</a> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**update:** *The Chrome results were so strange that I retook the tests again with more samples and averaged the results. Things look more reasonable now.*

I always try to optimize code for the fastest speed possible. While playing with some JavaScript (the culprit is this marvelous game by [@tonyvirtual](#) called Chain Reaction) I started thinking about the different speed gains that one might get from using different kinds of loops. A basic for loop in JavaScript can be implemented in many ways but typical are these 3:

```
// Loop 1
for (var i=0; i < Things.length; i++) {
    // Do something with Things[i]
};
```

```
// Loop 2
var i = Things.length-1;
do {
    // Do something with Things[i]
} while (i--);
```

```
// Loop 3
for (var i in Things){
    // Do something with Things[i]
}
```

There are many online reports that favor one of the loops instead of the others, but I wasn't convinced so I decided to test them myself. I devised a simple test that you can run in your browser or you can modified it by getting it from [github.com](https://github.com) and tested the script in 3 different browsers all on the Mac OS X 10.5.8. The browsers are **Firefox**

**10.0.2, Safari 5.0.6 and Chrome 19.0.1049.3 dev** so I don't know if these will apply to other browsers or OSes (use the github file).

The JavaScript performance results are listed below in the format **min (average 10x)**:

### ***Firefox***

Loop 1 (for ;;) — 117 (118.5)

Loop 2 (do – while) — 119 (125.9)

Loop 3 (for in) — 600 (671.6)

### ***Safari***

Loop 1 (for ;;) — 180 (185.8)

Loop 2 (do – while) — 178 (183.3)

Loop 3 (for in) — 1591 (1615.9)

### ***Chrome***

Loop 1 (for ;;) — 180 (191.2)

Loop 2 (do – while) — 159 (167.6)

Loop 3 (for in) — 585 (610.2)

### ***JavaScript traditional for loop is the fastest***

I am really surprised with these results. I knew that Mozilla had made some trick to make for loops fast, but I didn't imagine that Firefox would beat the other two. Also, the **for in** is to be avoided as it is slower. Convenience in this case comes at a price. I retook the test for Google Chrome and it now makes sense, the do-while is faster than the traditional one but still slower than Firefox's.

I don't understand this JavaScript performance differences but they certainly need more testing. The only conclusion until now is that the traditional JavaScript for loop is fast and should be used without second thoughts. You can test the loops in your own browser and see if the results match mine and then **post the results in the comments** below!

~~~ End of Article ~~~



Going loopy with JavaScript

Source<http://test.lummie.co.uk/code/javascript/going-loopy-with-javascript>

I am in the middle of writing a JavaScript tree that can handle 100s of thousands of nodes in the tree and needed to see if I can optimize the code. Trying to optimize the loop iterations for array objects, I put together a test case to time the efficiency of various different types of loops in JavaScript.

The test

Create an array of 1 million numbers 0 to 1 million, for each of the loop tests, increment a counter then display it at the end of the iterations.

The test covers the following loops:

- For Loop – standard for loop accessing the length of the array in the iterator
- For Loop Local Count – standard for loop, accessing the length of the array stored in a local variable before the iterator
- For Loop Local Count And Iterator – standard for loop accessing the count and the iterator from local variables
- For In Loop – standard For In loop
- For In Loop Local Item – standard For In Loop, but declaring the iterator variable locally first.
- While Loop – While loop iterator accessing the length at each iteration
- While Loop Count First – while loop but storing the length of the array in a local variable before the iterator.

```
var    itemCount = 100000000;
var    items = [];

function initArray()
{
    for(var x=0; x < itemCount; x++)
    {
        items.push(x);
    }
    console.log(items.length);
}

function forLoop()
{
    var total=0;
    for(var i=0;i < items.length; i++)
    {
        total ++;
    }
    console.log(total );
}
```

```
}

function forLoopLocalCount()
{
    var total=0;
    var count = items.length;
    for(var i=0;i < count; i++)
    {
        total ++;
    }
    console.log(total );
}

function forLoopLocalCountAndIterator()
{
    var total=0;
    var count = items.length;
    var i = 0;
    for(i=0;i < count; i++)
    {
        total ++;
    }
    console.log(total );
}

function forInLoop()
{
    var total=0;
    for(item in items)
    {
        total ++;
    }
    console.log(total);
}

function forInLoopLocalItem()
{
    var total=0;
    var item = null;
    for(item in items)
    {
        total ++;
    }
    console.log(total);
}

function whileLoop()
{
    var total=0;
```



```

    var i = 0;
    while(i<items.length)
    {
        total ++;
        i++;
    }
    console.log(total );
}

function whileLoopCountFirst ()
{
    var total=0;
    var i = 0;
    var count = items.length;
    while(i<count)
    {
        total ++;
        i++;
    }
    console.log(total );
}

function runTests()
{
    this.initArray();
    this.forLoop();
    this.forLoopLocalCount();
    this.forLoopLocalCountAndIterator();
    this.forInLoop();
    this.forInLoopLocalItem();
    this.whileLoop();
    this.whileLoopCountFirst();
    console.log("Complete.");
}

runTests();

```

Results & Conclusion

[Run the test yourself \(requires you to have firebug / profiler\)](#)

| | |
|------------------------------|------------|
| whileLoopCountFirst | 43.213ms |
| forLoopLocalCountAndIterator | 43.214ms |
| forLoopLocalCount | 43.363ms |
| whileLoop | 44.336ms |
| forLoop | 45.252ms |
| forInLoopLocalItem | 1698.295ms |

| | |
|-----------|------------|
| forInLoop | 2017.073ms |
|-----------|------------|

Timings on Firefox 12.0, using the firebug profiling.

So what can we conclude from the results?

- Don't use a For In loop. There is a small increase in performance if you allocate the item variable outside of the loop.
- Always ensure that the count (used in the loop exit test) is stored in a temporary variable outside of the iterator i.e. rather than calling array.length in the iterator exit test.
- When local variables are used to store the iterator and count, a for loop performs the same as a while loop.

~~~ End of Article ~~~



# JavaScript array “extras” in detail

**Source** <http://dev.opera.com/articles/view/javascript-array-extras-in-detail/>

## Introduction

In this article we'll look at the functionality made available by the new methods of *array objects* standardized in ECMA-262 5th edition (aka ES5). Most of the methods discussed below are *higher-order* (we'll clarify this term shortly below), and related to functional programming. In addition, most of them have been added to different JavaScript implementations since version 1.6 (SpiderMonkey), although these were only standardized in ES5.

Unless stated otherwise, all the methods discussed below were introduced in JavaScript 1.6.

Note: You can probably learn a lot from this article whether you are an expert in JavaScript, or a comparative novice.

## Browser support

At the time of writing “Array extras” (which are actually standardized methods, rather than extras) are supported by the new versions of all major browsers. Unless stated otherwise, all the discussed methods can be safely used in:

- Opera 11+
- Firefox 3.6+
- Safari 5+
- Chrome 8+
- Internet Explorer 9+

If we have to support older browsers we can always implement our own versions of required methods by extending the `Array.prototype` object, for example:

```
// for old browsers

if (typeof Array.prototype.forEach != "function") {
  Array.prototype.forEach = function () {
    /* own implementation */
  };
}
```

With the introductory information out the way, we'll start our exploration of array extras by looking at the theory and practical rationale behind the methods.

## Theory and rationale

Every new generation of a programming language arrives with newer and higher abstractions. These new abstractions make our development (and the perception of programs in general) easier and allow us to control *complex structures* in *simpler ways*. Consider e.g. the following two functions:

```
// sum of numbers in range
```

```
function getSum(from, to) {
    var result = 0;
    for (var k = from; k < to; k++) {
        result += k;
    }
    return result;
}

var sum = getSum(1, 10); // 45

// sum of squares of numbers in range

function getSumOfSquares(from, to) {
    var result = 0;
    for (var k = from; k < to; k++) {
        result += k * k;
    }
    return result;
}

var sumOfSquares = getSumOfSquares(1, 10); // 285
```

In first function we loop through the numbers in the required range and collect the sum of the numbers. In the second function we do the same, but collect the squares of the numbers. What if we wanted to provide a function that calculates the sum of *cubes* of numbers for example, or *any* possible transformation?

Obviously, we have almost identical code structures in the two previous examples: In a well-designed system we'll want to *reuse* the common parts. This is called *code reuse* in computer science. Generally, it may appear in several aspects (for example, in OOP we can *reuse* code from *ancestor classes* in *descendant classes*).

The common part of the above two functions (the exact *action* applied on current number *k*) can be *encapsulated* into a *function*. In such a way, we can *separate* the common (often boring) part of the *processing* (the *for...length* in this case) from the *transformation* made on the each element. Having such an approach, the transformation can be *passed as an argument* to our common function, like so:

```
function genericSum(handler, from, to) {

    var result = 0;

    for (var k = from; k < to; k++) {

        result += handler(k);

    }

}
```

```
    return result;

}
```

Here every subsequent summed value is presented not simply as the current number, but as a *result of the transformation* (provided by the function handler) made on the number. That is, we get the ability to *parameterize* the handling of every element in the sequence.

It's a very powerful abstraction, which allows us to have for example *just a sum* (where the handler function simply returns the number):

```
var sum = genericSum(function (k) { return k; }, 1, 10); // 45
```

or the *sum of squares*:

```
var sumOfSquares = genericSum(function (k) { return k * k; }, 1, 10); // 285
```

or even *sum of cubes*:

```
var sumOfCubes = genericSum(function (k) { return k * k * k; }, 1, 10); // 2025
```

And all this using only one function: `genericSum`.

Functions that *accept other functions* as arguments (as is the case with our `genericSum` function) are called *higher-order functions (HOF)*. And functions that can be *passed* as an argument are called *first-class functions*.

Having this combination of higher-order and first-class functions in JavaScript allows us to create very expressive and highly-abstracted constructions, which help us to solve *complex* tasks in an *easier* manner, *conveniently reusing the code*.

That covers the theory. Let's see what we can do in practice.

## Arrays: extra processing

The pattern described above gives us an almost unlimited number of ways to carry out *generic processing of arrays*. Thus, as we said above all the boring details of applying this processing is hidden from us. Instead of repeating for `k ... length` every time, we concentrate on the task itself, leaving the non-interesting (*lower-abstracted*) details behind the scenes. JavaScript has several HOFs for parameterized array processing. They are all available on the `Array.prototype` object and therefore available on every array instance. Let's consider these methods.

### forEach

The most frequent one of these methods you'll encounter, which corresponds to *parameterized looping* over an array is the `forEach` method. It simply applies a function on each element in the array. This means that *only existing elements* are visited and handled. For example:

```
[1, 2 ,3, 4].forEach(alert);
```

Here, “*the function passed in as an argument is applied to each item in the array*”, which in this case is an alert. So what is the difference between this and a casual for...length loop such as:

```
var array = [1, 2, 3, 4];

for (var k = 0, length = array.length; k < length; k++) {

    alert(array[k]);

}
```

Since we can't refer to an array without a variable, we use an additional variable array; for the loop counter we also use the variable k. And the code itself becomes longer because we are repeating the for...length loop over and over again. We could of course use another iteration (e.g. while) and wrap the code into a function (thereby hiding helper variables and not polluting the global scope), but obviously this is less abstract than the forEach approach.

If we replace the action function with for example console.log, we get another interesting result:

```
[1, 2 ,3, 4].forEach(console.log);
```

```
// Result:
```

```
// 1, 0, [1, 2, 3, 4]
// 2, 1, [1, 2, 3, 4]
// 3, 2, [1, 2, 3, 4]
// 4, 3, [1, 2, 3, 4]
```

The Debug function console.log (which works with Opera Dragonfly or Firebug) can accept any number of arguments: here *three* arguments are passed to every call of console.log by the forEach function.

It's not hard to see that these arguments are: the *current item*, the *index* of the item, and the *array itself*. We can provide any function of three arguments and perform required actions with these arguments:

```
var sum = 0;

[1, 2, 3, 4].forEach(function (item, index, array) {
```

```
    console.log(array[index] == item); // true

    sum += item;

  });

  alert(sum); // 10
```

Thus we get the first generic higher-order method of arrays, whose signature is defined as:

```
array.forEach(callback,[ thisObject])
```

The first argument is already known to us — it's a function of three arguments, which is applied for items. The second argument is a *context object* (or a *this value*), which will be used as a value of *this* in the code of the applied function. It can be useful, for example when we want to use a method of an object as a processing function:

```
var database = {

  users: ["Dmitry", "John", "David"],

  sendEmail: function (user) {

    if (this.isValidUser(user)) {

      /* sending message */
    }
  },

  isValidUser: function (user) {
    /* some checks */
  }

};

// send an email to every user

database.users.forEach( // for each user in database
  database.sendEmail,   // send email
  database               // using context (this) as database
);
```

Let's discuss what is going on here. Inside the `sendEmail` activation function, the `'this'` value is set to a `database` object, and `this.isValidUser` refers to the required function. If we didn't pass this second argument, the `'this'` value would be set to the *global object* (in browsers it's `window`) or even to `undefined` in strict mode.

Let's show again, that *only existing* items are handled (i.e. “holes” are *not included* into the process):

```
var array = [1, 2, 3];

delete array[1]; // remove 2
alert(array); // "1,,3"

alert(array.length); // but the length is still 3

array.forEach(alert); // alerts only 1 and 3
```

## map

Sometimes we might want to get the *transformation* or the *mapping* of the original array. JavaScript provides a HOF for that too: `map`. This function has a signature, as follows:

```
array.map(callback,[ thisObject])
```

This method also applies callback functions for each element of an array (again, only in the required context of `this`, and only for existing items). It does however also *return* the *transformed* (*mapped*) array as a result. Take a look at this example:

```
var data = [1, 2, 3, 4];

var arrayOfSquares = data.map(function (item) {

    return item * item;

});

alert(arrayOfSquares); // 1, 4, 9, 16
```

In practice we may use this technique to get any transformation of a list. For example, if we have a list of user objects, we can get the list of their email addresses:

```
var users = [

    {name: "Dmitry", "email": "dmitry@email.com"},

    {name: "John",   "email": "john@email.com"},

    {name: "David",  "email": "david@email.de"},

    // etc
```



```
];  
  
var emails = users.map(function (user) { return user.email; });  
  
alert(emails); //["dmitry@email.com", "john@email.com", "david@email.de"]
```

## filter

Instead of the basic mapped result, we may want to only get certain entries that *satisfy a certain condition*, for example ones that have an email address that starts with "d". We can create a filter for exactly this kind of purpose, which will exclude items that don't pass our conditions. The filter method can be used to do this quickly and easily.

The signature is quite similar to that of map:

```
array.filter(callback,[ thisObject])
```

The callback function of the filter should return the *boolean* value (either true or false). true means that the filter is *passed*, and false means that an item shouldn't be included in the result set.

Considering the previous example, we can select a subset of users or emails, for example only emails that are registered in the com domain:

```
var comEmails = users //from users ...  
  
    // get emails ...  
    .map(function (user) { return user.email; })  
  
    // and remove non-needed, leaving only "com"-emails  
    .filter(function (email) { return /com$/.test(email); });  
  
alert(comEmails); //["dmitry@email.com", "john@email.com"]
```

Note how we used the *chained pattern* of method invocations. This is quite a normal practice in JavaScript — the map method returns an array so we can then directly call the next method of the array, i.e. filter. In the latter we use a regular expression to check whether passed email addresses end with the com string (the \$ sign means “the end of the testing string”). Note also that in the former case we accept the user *object*, whereas in the second case we already have the email *string* available.

It's not hard to see that, even though we have highly-abstracted handling here, we nevertheless have an *inefficient operation*. Indeed, we go through the whole array *twice*. It would be great if we could do all the needed checks and mapping in one pass. There is some syntactic sugar available to do that: map + filter. This is called *array comprehensions*. Currently it's implemented only in Firefox, but it is still worth covering here:

```
var comEmails = [user.email for each (user in users) if (/com$/.test(user.email))  
];
```

```
alert(comEmails); // ["dmitry@email.com", "john@email.com"]
```

The code snippet above basically says “*build an array of user emails if the email ends with the string com*”. If we don’t have array comprehensions available, we can always fall back to the simple for enumeration:

```
var comEmails = [];  
  
var email;  
  
for (var k = 0, length = users.length; k < length; k++) {  
  
    email = user.email;  
  
    if (/com$/.test(email)) {  
  
        comEmails.push(email);  
  
    }  
  
}  
  
alert(comEmails); // ["dmitry@email.com", "john@email.com"]
```

The choice is ours. Sometimes it is convenient (for example when the operation of transformation is not known in advance) to use array processing HOFs, and sometimes it’s more useful and efficient to use an old-school way.

**Note:** One thing to note in the above example is that we left two “garbage” variables intact after our actions had completed: `k` and `email`. This wouldn’t happen if we used `forEach` and array comprehension, so we need should consider this also in our choice.

## some

Often we want to know whether *some* or *all* items of a collection satisfy a specified condition. JavaScript provides two array methods allowing us to create easy solutions to such problems: `some` and `every`. We’ll tackle `every` in the next section; we’ll look at `some` first:

```
array.some(callback,[ thisObject])
```

This method accepts the function of three arguments and the context object. However, the result of the `some` function is *boolean*. It returns `true` if *some* (that is, *at least one*) of the items satisfies the condition. The condition is determined by the callback function, which also should return a Boolean result.

For example, we might be storing test results, and want to test whether some user's scores are higher than a certain threshold:

```
var scores = [5, 8, 3, 10];

var current = 7;

function higherThanCurrent(score) {

    return score > current;

}

if (scores.some(higherThanCurrent)) {

    alert("Accepted");

}
```

This code gives the result "Accepted", since the `some` method determines that the second element of the `scores` array (value 8) is higher than the current item, value 7. The processing therefore stops, returning `true`.

This technique can be used for performing a *complex search* (i.e. meeting several conditions at once) of the first found element in the array:

```
var found = null;

var points = [

    {x: 10, y: 20},

    {x: 15, y: 53},

    {x: 17, y: 72}

];

points.some(function (point) {

    if (point.x > 10 && point.y < 60) {

        found = point; //found

    }

})
```

```
        return true;
    }

    return false;

});
if (found) {
    alert("Found: " + found.x + ", " + found.y); // Found: 15,53
}
```

We could also use `forEach` for searching; however `forEach` wouldn't stop on the first found element: we'd need to throw a special exception to exit from it.

We'll look at testing an element using just the `===` operator to provide a simple search below.

## every

By contrast, if we instead want to test whether all the scores are higher than the threshold, we can use the `every` method, which looks like this:

```
array.every(callback,[ thisObject])
```

Our updated example looks like so:

```
if (scores.every(higherThanCurrent)) {

    alert("Accepted");

} else {

    alert("Not all scores are higher than " + current);

}

// change our value to 2
current = 2;

// now it's OK
alert(scores.every(higherThanCurrent)); // true
```

## indexOf

Another frequent task we'll run into is testing whether an element is present in a collection. There are two convenient methods to do this: `indexOf` and `lastIndexOf`, which simply search an element, testing it with *strict equality* `===` operation. The `indexOf` definition is as follows:

```
array.indexOf(searchElement[, fromIndex])
```

This method results an *integer* index of a searched element in the list. In a case where an item is not found, the value -1 is returned. The `fromIndex` parameter is optional — if this is passed then the search starts from this index. If omitted, the default value 0 is used (i.e. the whole array is searched):

```
var data = [2, 5, 7, 3, 5];

alert(data.indexOf(5)); // 1

alert(data.indexOf(5, 3)); // 4 (start search from 3 index)

alert(data.indexOf(4)); // -1 (not found)

alert(data.indexOf("5")); // -1 (also not found since 5 !== "5")
```

## lastIndexOf

`lastIndexOf` is very similar to `indexOf`, except that it searches the element starting *from the end* of the array. The `lastIndexOf` definition is as follows:

```
array.lastIndexOf(searchElement[, fromIndex])
```

The `fromIndex` parameter is again optional; the default value for it is the array length - 1:

```
var data = [2, 5, 7, 3, 5];

alert(data.lastIndexOf(5)); // 4

alert(data.lastIndexOf(5, 3)); // 1 (start search from 3 index)

if (data.indexOf(4) == -1) {

    alert("4 is not found");

}
```

## reduce

The last two new methods we'll discuss allow us to *reduce* an array into a single value: they are `reduce` and `reduceRight`. The former starts its analysis from the beginning, while the latter starts it from the end. These methods were introduced into JavaScript later than others: at version 1.8.

We'll discuss `reduce` first, and then go on to `reduceRight` in the next section. `reduce` has the following definition:

```
array.reduce(callback[, initialValue])
```

The callback function accepts *four* arguments: *previous value*, *current value*, *index*, and again the *array* itself. The `initialValue` parameter is optional and, if omitted, is set to *the first element* of the array. Consider the following example:

```
// reduce the array to sum of elements

var sum = [1, 2, 3, 4].reduce(function (previous, current, index, array) {
    return previous + current;
});

alert(sum); // 10
```

Here we go through the array elements and get:

1. The *previous* value of every callback, which initially is *the first element* since it's equal to the default `initialValue`
2. The *current* value of every callback, which at first call is 2
3. The two last arguments — `index` and `array`

We then return the sum of our previous and current values, which becomes the *previous value* of the *next iteration*, and the current value is set to the next element, i.e. to 3. The process loops until the end of the array:

```
// initial set

previous = initialValue = 1, current = 2

// first iteration

previous = (1 + 2) = 3, current = 3

// second iteration

previous = (3 + 3) = 6, current = 4

// third iteration

previous = (6 + 4) = 10, current = undefined (exit)
```

The resulting value is not required to be a primitive value. With `reduce` we can, for example, transform two-dimensional arrays into flat vectors:

```
var matrix = [
```

```
[1, 2],  
  
[3, 4],  
  
[5, 6]  
  
];  
  
alert(matrix[0][1]); // 2  
  
// and now get the flatten array  
  
var flatten = matrix.reduce(function (previous, current) {  
    return previous.concat(current);  
});  
  
alert(flatten); // [1, 2, 3, 4, 5, 6]
```

## reduceRight

The definition of `reduceRight` is as follows:

```
array.reduceRight(callback[, initialValue])
```

This function works in the same way as `reduce`, except that it processes an array from the end. Let's have look at an example:

```
var data = [1, 2, 3, 4];  
  
var specialDiff = data.reduceRight(function (previous, current, index) {  
  
    if (index == 0) {  
  
        return previous + current;  
  
    }  
  
    return previous - current;  
  
});  
  
alert(specialDiff); // 0
```

This results in a value of zero. I'm going to leave the explanation for you as an exercise: draw every step of the process, like we did in the previous example.

## Generic nature

One of the biggest advantages of the array methods discussed in this article is the fact that they are all *generic* with respect to the objects on which they operate. In other words, *it's not required* that the object to process should be an array. The object just needs the `length` property, and numeric indices.

This means we can *reuse* the functionality of arrays, applying it to other kinds of objects, for example strings:

```
// get the reference to the map method

var map = Array.prototype.map;

// and call it for a string

var hello = map.call("hello world", function (char) {

    return char + "*";

});

alert(hello.join("")); // "h*e*l*l*o* *w*o*r*l*d*"
```

Here we apply the `map` function to the "hello world" string, then get the result as an *array* (yes, the `map` function has converted the string into an array) and then convert the array into another string — "h\*e\*l\*l\*o\* \*w\*o\*r\*l\*d\*". This is of course only one solution, included to show the generic nature of the methods: we could instead solve these using regular expressions, or with a combination of `split` and `join` functions.

This approach can work in the opposite way too — here's how we can reuse a string method to handle an array:

```
// reuse "toUpperCase" method
var toUpperCase = String.prototype.toUpperCase;

var upper = toUpperCase.apply(["foo", "bar"].split(","));

alert(upper); // ["FOO", "BAR"]
```

In Firefox these generic methods are duplicated for constructors as well, as a *non-standard extension*: this provides an even more convenient generic application:

```
// reuse array's "map" method for a string
Array.map("foo", String.toUpperCase).join(""); // "FOO"

// reuse string's "toLowerCase" method for an array
String.toLowerCase(["F", "O", "O"].split(",")); // ["f", "o", "o"]
```



Another example — the arguments object isn't an array and hasn't got such methods available intrinsically. However, it has length and properties-indices: here's a better way to handle passed arguments:

```
// function "foo" accepts
// any number of arguments

function foo(/* arguments */) {
    var every = Array.prototype.every;
    var allNumbers = every.call(arguments, function (arg) {
        return typeof arg == "number";
    });

    if (!allNumbers) {
        throw "Some argument is not a number";
    }

    /* further handling */
}

foo(1, 2, 3); // OK
foo(1, 2, "3"); // Error
```

We can also call a method of an array for the DOM nodes collection, even though the DOM NodeList collection is not an array and has none of the discussed methods natively:

```
var paragraphs = document.querySelectorAll("p");

[].forEach.call(paragraphs, console.log);
```

In this example we select all paragraphs in the document and then log every paragraph to the console. Note how we reuse the forEach method of the array — the empty array is created only to get the reference to the forEach method.

We can use all the other array methods discussed in this article in exactly the same way.

## Summary

Higher-order methods of arrays provide convenient and elegant ways to process different collections. Having the ability to parameterize the action applied on the element of a sequence increases the abstraction and therefore makes our code cleaner and shorter, with easier handling of complex structures.

At the same time, sometimes it can be more efficient to fall back to a lower abstraction level. Bear in mind that every new abstraction level can bring some kind of performance penalty, in exchange or providing more convenient ways of programming. We can use different programming styles depending on our situation and needs.

~~~ End of Article ~~~



Session 15: Functions and Objects

JavaScript examples using the DOM

Source <http://bobpeers.com/technical/dhtml1>

The xmlHTTP object

This uses the xmlhttp object to get data from a server but without refreshing the page. Usually to get data from the server in response to a form or other user input we need to submit the form which reloads the entire page. This is very useful when we need to update part of a page with fresh data but don't want the current view to alter or page to reload.

Another great advantage is that we can carry out server side processing using PHP or any other language and return this data back to the page using this method. There seems to be a lot of talk about this right now and possibly some over use since the results can be pretty cool, but be aware that if you rely on this and your users have JavaScript turned off then your cool page will fail completely.

Some of the better known sites that use the xmlhttp object are Gmail which gives the site a very 'snappy' feel since the whole page does not need to be reloaded when clicking on links, Google Suggest which returns possible searches as you type based on the most likely results and Google Maps which allows you to 'drag' the map and load new maps in the background. I have also used this method on my contact page so that an email can be sent without navigating away from the contact page to the usual 'Thanks for contacting us' page, which is otherwise quite useless. If you have JavaScript turned off then you can still email me as the form will then be submitted as normal.

Note that although it is called xmlhttp you can return any data you like, in my example I return a simple number, but you could equally return xml, maybe something like below in my example.

```
<?xml version='1.0' standalone='yes'?> <convert> <temp>69.8</temp>
</convert>
```

Of course if you decide to send xml data from your script make sure you also sent the correct Content-type of text/xml, this is done with PHP like this:

```
header('Content-Type: text/xml');
```

Also remember to use xmlhttp.responseXML instead of xmlhttp.responseText when getting the data. I decided not to return xml data as then I would have to parse the returned xml using javascript which was an unnecessary overhead in this simple case.

In this example the temperature and conversion required are sent via POST to the page 'add' on this server. The output from this script is then sent back to the page and the value displayed in the last field. The PHP script is very simple, it just takes the POST variables and uses them to convert from one temperature scale to another returning the resulting conversion. This code tests for window.XMLHttpRequest (used by Mozilla, Safari, Opera) and if this doesn't work then uses the

Microsoft version of new ActiveXObject("Microsoft.XMLHTTP"), if both fail the function returns false. This created object is then set to the xmlhttp variable.

```
var xmlhttp=false;
if (window.XMLHttpRequest)
{
xmlhttp = new XMLHttpRequest();
}
else if(window.ActiveXObject)
{
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
else
return false;
```

Change the value of the result box label depending on whether we are converting to Fahrenheit or Celsius.

```
if(document.getElementById("f").checked)
{
document.getElementById("resulttext").replaceChild
(document.createTextNode("Temp in
C"),document.getElementById("resulttext").childNodes[0])
}
else
{
document.getElementById("resulttext").replaceChild
(document.createTextNode("Temp in
F"),document.getElementById("resulttext").childNodes[0])
}
```

Get the values of the text boxes and checkbox and send these values in a querystring to the page add using the GET method.

```
xmlhttp.open("GET", "test/add?temp=" +
document.getElementById("temp").value + "&scale=" +
document.getElementById("c").checked,true);
```

Here we wait until the data is received. We set up a function to be called when the state of the xmlhttp object changes, when readystate==4 then we show the data using xmlhttp.responseText (or xmlhttp.responseXML if you are getting XML data from the server).

```
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4)
```

```
{
document.getElementById("result").value=xmlhttp.responseText
document.close()
}
}
xmlhttp.send(null)
}
```

Enter temp

This is °C ☐

This is °F ☐

Converted temp

Regular Expression tester

This uses JavaScript to test a regular expression. Enter the expression to match against the 'RegExp match' field and the test string in the 'Text to compare' field. Click 'Compare' and either true or false will appear in the 'Is match?' field.

```
function test()
{
var strtest=document.getElementById("input").value;
var Exp=document.getElementById("strExp").value;
var myreg=new RegExp(Exp);
var ans=myreg.test(strtest);
document.getElementById("output").value=ans;
}
```

RegExp match string

Test string

Is match?

Traversing DOM nodes

This code assigns all the <DIV> elements to the variable tags using document.getElementsByTagName("div"). Next it loops through all the elements of this array until it finds an element with an id of 'pagebody', which is the main page content div on this page. Next it loops through all the.childNodes of this div looking for all the h2 tags, then changes the background color to #228800 or #fff and text color #fff or #000 depending on how many times the button has been pressed. Note that I also test for Internet Explorer using if (document.all) since it seems that IE requires uppercase tag names whereas Mozilla uses lowercase, as I might have expected.

Press to toggle the background color of all <H2> tags.

```
var toggle=0; function highlight()
{ var div="div"; var h2="h2"; if (document.all)
{ div="DIV"; h2="H2";
  }
var col; (toggle%2==0)?col="#228800":col="#fff";
(toggle%2==0)?textcol="#fff":textcol="#000";
toggle++;
var tags=document.getElementsByTagName("div");
for(var x=0;x<tags.length;x++) {
if(tags[x].id=="pagebody")
{
var newtags=tags[x].childNodes;
for(var z=0;z<tags[x].childNodes.length;z++) {
if(newtags[z].nodeName==h2) {
newtags[z].style.backgroundColor=col;
newtags[z].style.color=textcol;
} } } }
}
```

~~~ End of Article ~~~



## Session 16: Building a Mobile Web Application

### Mobile Application Development

|               |                                                                                                                                                                   |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source</b> | <a href="http://www.nitrix-reloaded.com/2012/04/07/mobile-application-development/">http://www.nitrix-reloaded.com/2012/04/07/mobile-application-development/</a> |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Mobile Applications are rapidly developing segment in the global mobile sector. Developing mobile applications targeting different mobile platforms such as Windows Phone, iOS, Android, Blackberry and Bada (Samsung's Proprietary OS) is the trend setter now.

**Mobile application development** is the process by which application software is developed for small low-power handheld devices such as personal digital assistants, enterprise digital assistants or mobile phones or smart phones. These applications are either pre-installed on phones during manufacture, downloaded by customers from various mobile software distribution platforms, or web applications delivered over HTTP which use server-side or client-side processing (e.g. JavaScript) to provide an "application-like" experience within a Web browser.

#### **Why is it so different from other application development?**

Or

#### **Why do we need to give special care when doing mobile development?**

Mobile devices have certain limitations in terms of hardware, they have limited screen display, limited space for applications to operate and network capabilities. These limitations will vary in different models as the Mobile ecosystem is very large and there is innumerable amount of devices in the world. So we need to give special care in developing the mobile applications ensuring the compatibility across all the platforms.

*Some of the things we need to consider when we do mobile development are:*

- **Small screen size and mobile ecosystem** – This makes it difficult or impossible to see text and graphics dependent on the standard size of a desktop computer screen.
- **Lack of windows/multi-tasking** – On a desktop computer, the ability to open more than one window at a time allows for multi-tasking and for easy reverts to a previous page. Historically on mobile web, only one page can be displayed at a time, and pages can only be viewed in the sequence they were originally accessed. Latest mobile platform releases such as iOS, Android and Windows Phone are supporting multi-tasking capability, which allows you to run more than one applications parallel, yet only one application can be displayed on the screen.

- **Navigation** – Most mobile devices do not use a mouse like pointer, but rather simply an up and down function for scrolling or touch inputs, thereby limiting the flexibility in navigation.
- **Hardware and Resource limitation** – all the upcoming mobile devices are having Dual Core or Quad core mobile processors with decent amount of memory. But still if you need your device to function well and be responsive, you should develop application in such a way it will be using low memory footprint.
- **Targeting Or Supporting wide variety Mobile Platforms/ Execution environments**– this is the most complicated or important feature set we should be taking in to consideration when developing an application. Android, iOS, BlackBerry, HP webOS, Symbian OS, Bada from Samsung, and Windows Mobile support typical application binaries as found on personal computers with code which executes in the native machine format of the processor (the ARM architecture is a dominant design used on many current models).

Also read this article on 7 Limitations when designing for mobile:

<http://baymard.com/blog/mobile-design-limitations>

For Technical point of view we can differentiate mobile applications in to three categories

### **1. Native Applications**

Native applications developed against native platform APIs and It would be having full(or limited for some platforms) access to the device capabilities. Each of the platforms for mobile applications also has an integrated development environment which provides tools to allow a developer to write, test, and deploy applications into the target platform environment.

*Examples* are using Applications developed using Native/Platform Mobile SDK's such as iOS SDK, Android SDK, Windows Phone SDK etc. Windows Mobile, Android, HP webOS and iOS offer free SDKs and integrated development environments to developers. These applications will be able to utilize or interact with all device capabilities.

Five reasons for developing Native Applications:

- Performance
- Offline Mode
- Findability / Discoverability (through a central location such as Apple's AppStore, Google's Market, Windows Phone Market etc.)
- Device Attributes
- Monetization



The **disadvantage** is obviously the development cost. No two mobile platforms can share the same mobile application, and there are too many Mobile operating systems (or platforms) existing in the market. If you develop a mobile application to market it widely, you need to develop that in Symbian, Mac iPhone, Android, Blackberry and Windows mobile.

## **2. Mobile Web Applications/Browser based Applications**

Mobile Web applications are web sites that are scaled down/optimized to display on a mobile web browser. Mobile web applications are developed in such a way keeping in mind less payload delivery per each request. These applications are typically a scale down version of actual website which you can browse over any desktop browser such as Internet Explorer, Chrome, Firefox and Opera. Every action mobile web application makes a round trip to server.

### **The advantages of having mobile web application is that:**

- Server Driven – we have full control on the website and we can customize it any time, so that user will get the latest look and feel.
- Targeting multiple platforms & connectivity – most of the mobiles and smartphones have a browser. One-third of humankind currently has access to the Internet through a mobile device. That makes it easy to deliver our applications in any platform.
- It enables services to take advantage of mobile device capabilities such as clicking on a phone number to call it or add it to the device address book. It can provide location-sensitive content. Location technologies can enable location-sensitive information be provided to a user. This can reduce the steps required for the user to reach useful content, and so makes it accessible with the least amount of effort.
- For users, they don't have to download an application or any maintenance updates, but instead "call up" a URL via their mobile browser which instantly delivers the most up-to-date application to their device.
- Cross platform compatibility.
- Low development cost.

The **disadvantage** is the bandwidth limitations and the limitations of Mobile websites, which does not access your Phone's components like your Address book, Camera, etc.

The **Mobile Web** refers access to the World Wide Web, i.e. the use of browser-based Internet services, from a handheld mobile device, such as a smartphone, a feature phone or a tablet computer, connected to a mobile network or other wireless network.

Traditionally, access to the Web has been via fixed-line services on large-screen laptops and desktop computers. The shift to mobile Web access has been accelerating with the rise since 2007 of larger multi-touch smartphones, and of multi-touch tablet computers since 2010.

Both platforms provide better Internet access and browser- or application-based user Web experiences than previous generations of mobile devices have done.

Mobile Web access today still suffers from interoperability and usability problems. Interoperability issues stem from the platform fragmentation of mobile devices, mobile operating systems, and browsers. Usability problems are centered around the small physical size of the mobile phone form factors (limits on display resolution and user input/operating).

### **3. Hybrid Mobile Applications**

Hybrid Mobile Applications are Applications that use BOTH browser interfaces and native mobile components. With HTML5 and JavaScripts, now the browsers are becoming capable of accessing a phone's built in features like contacts, camera etc.

There are mobile frameworks based on HTML5 and JavaScript's that would be able to access the device features such as Accelerometer, Compass, Contacts, Camera and device file system etc. Some popular such mobile frameworks are **PhoneGap**, **Appcelerator's Titanium** etc. These frameworks helps in developing applications that would able to target most of the mobile platforms such as "iOS", "Android", "Windows Phone", "Symbian" and "Blackberry".

*Appcelerator Titanium* is a platform for developing mobile, tablet and desktop applications using web technologies.

#### **Key to successful Hybrid Apps:**

- Make the experience feel like a native application.
- Take advantage of the enhanced features.
- Don't simply release a hybrid version of the mobile web site.
- Optimize performance

Mobile world is growing and you can see lots of opportunities for innovation. If you are a fan of building mobile applications go ahead and develop something that is in your dreams and leave it over to the rest of the world to carry forward.

#### **Resources for Developing mobile applications:**

- **PhoneGap** (was called by the name **Apache Callback**, but now **Apache Cordova**) is an open-source mobile development framework produced by Nitobi, purchased by Adobe Systems. It enables software programmers to build applications for mobile devices using JavaScript, HTML5 and CSS3, instead of lower-level languages such as Objective-C. The resulting applications are hybrid, meaning that they are neither truly native (all layout rendering is done via the webview instead of the platform's native UI framework) nor purely web based (they are not just web apps but packed for appstore distribution, and have access to part of the device application programming interface).  
<http://phonegap.com/>

- **Appcelerator Titanium** is a platform for developing mobile, tablet and desktop applications using web technologies. Appcelerator Titanium is developed by Appcelerator Inc.  
Appcelerator Titanium Mobile is one of several phone web based application framework solutions allowing web developers to apply existing skills to create native applications for iPhone and Android. Yet, while using the familiar JavaScript *syntax*, developers will also have to learn the Titanium API, which is quite different from familiar web frameworks such as jQuery.  
<http://www.appcelerator.com/>
- **Android Application Development:** <http://developer.android.com/index.html>
- **Apple iOS Application Development:** <https://developer.apple.com/>
- **Windows Phone Development:** <http://create.msdn.com/en-US/>  
[http://msdn.microsoft.com/en-us/library/ff402535\(v=vs.92\).aspx](http://msdn.microsoft.com/en-us/library/ff402535(v=vs.92).aspx)
- **Blackberry Application Development:**  
<https://bdsc.webapps.blackberry.com/devzone/>
- **Bada Development:** <http://developer.bada.com>
- **Symbian Development:** <http://www.developer.nokia.com/Devices/Symbian/>
- **HP WebOS Development:** <https://developer.palm.com/>
- **jQuery Mobile:** Touch-Optimized Web Framework for Smartphones & Tablets  
<http://jquerymobile.com/>
- **Sencha Touch:** *Sencha Touch* was the first HTML5 mobile JavaScript framework for web applications that feel native on Android, BlackBerry, and iOS.  
<http://www.sencha.com/products/touch>  
and more...

~~~ End of Article ~~~



Mobile Development and Compilers HTML5

Source

<http://technews-cloud-computing-mobile-bi-services.wikispaces.com/Mobile+Development+and+Compilers+HTML5>

HTML4 is the conventional that has taken technique execution since the beginning of the century. In addition to JavaScript and CSS 2, these three groundwork support beams have formed web style over the last several years.

One key unable of the HTML4, CSS and JavaScript sacred trinity, has been a lack of movement and multi-media incorporation. Although mootools and jquery JavaScript collections address the movement issues to an level, movie has generally only been easily corner online browser / corner groundwork possible with Display.

Mobile Application Development Services: HTML5 is the conventional to substitute HTML4, however sites known as in HTML5 will only operate on the newest editions of Opera, Opera or Chrome 4 and IE 9 only has partially execution of the new requirements. A reward is that it will operate on the planet of cellular phone devices such as iPhone, Rim and Operating system centered mobile phones. Mobile sites developed and known as in HTML5 not only appreciate many of the new functions but also perform on different devices. In 2011 it is now plenty of a chance to start thinking about changing HTML4 with HTML5, which brings many plug-ins to the current requirements including:-

- in built movie and media handling
- a far more powerful set of style rules in CSS3
- the material item to handle illustrating, and display requirements

In inclusion to stuffing the gap currently repaired by Adobe Display technological innovation, HTML5 presents a whole new set of style and other functions in its own right including:-

- procedure storage
- local information storage
- SQL information storage

This collaboration of tools strongly ensures HTML5 as database incorporation groundwork, although even with the inclusion of this tremendous customer part toolset, hosting server part groundwork of some information (php, coffee or .net) will almost certainly be needed to create a complete program. Unfortunately presently, there is a problem with pc surfers as many users have old surfers and some do not adhere to the newest requirements. Drupal is recommended for sites developed for pc surfers presently.

Using HTML5 for Mobile Internet

Unlike personal computers, new technological innovation mobile phones usually meet requirements and so HTML5 can be used in the growing community of the mobile phone, illustrations are the iPhone (iOS), Rim and Operating system centered mobile phones. Perhaps even furthermore HTML5 is the one conventional that holds these different technological innovation (e.g. the iOS and the Operating system OS) together. Thus cellular sites known as in HTML5, not only appreciate many of the new functions, but also perform on different devices.

Cross Producing HTML5 Programs.

Whilst this use of HTML5 for cellular sites is in itself remarkable, there is an even more big benefit, with the use of compilers such as adAPPt Titanium and Phonegap, HTML5 applications can be corner collected to perform on the iPhone, Operating system, Rim and Windows cellular.

Conclusion

Whilst HTML5 (unlike Drupal) may not be taking the community of web style and pc exploring by weather, it has completely changed the community of cellular app progression. Not only does HTML5 provide a very affordable way to develop 'cross platform', but it also considerably decreases the price of maintaining applications at some factor. Companies (like DrupalTheme and adAPPt.co.uk) who are at the leading edge of android progression and use this new technological innovation are well placed for the long run, as cellular surfers are set to surpass pc surfers by 2013.

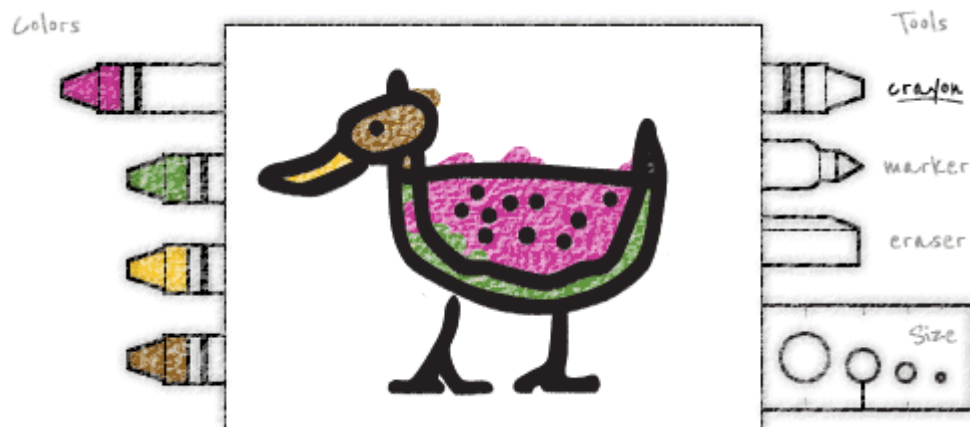
~~~ End of Article ~~~



## Session 17: Canvas and JavaScript

### CREATE A DRAWING APP WITH HTML5 CANVAS AND JAVASCRIPT

**Source** <http://www.williammalone.com/articles/create-html5-canvas-javascript-drawing-app/>



This tutorial will take you step by step through the development of a simple web drawing application using HTML5 canvas and its partner JavaScript. The aim of this article is to explore the process of creating a simple app along the way of learning:

- How to draw dynamically on HTML5 canvas
- The future possibilities of HTML5 canvas
- The current browser compatibility of HTML5 canvas

Each step includes a working demo; if you want to skip ahead:

- Simple Demo
- Colors Demo
- Sizes Demo
- Tools Demo
- Outline Demo
- Complete Demo

#### *Define Our Objective*

Let's create a web app where the user can dynamically draw on an HTML5 canvas. What will our users use? A coloring book comes to mind; that means crayons. Our first tool is a crayon. Although the real world doesn't agree, I think we should be able to erase crayons. Our second tool will be an eraser (sorry reality). And because I have always have been a Sharpie® fan our final tool will be a marker.

Our tools could use colors (except maybe our eraser). Let's keep it simple, so we give our user 4 different colors to choose from.

Similarly let's also give our user 4 different sizes to draw with, because we can. To recap our app should have the following:

- 3 tools: crayon, marker, eraser
- 4 colors to choose from (except eraser)
- 4 sizes to choose from

Like a coloring book, let's give our user something to "color". I have chosen a favorite of mine: **Watermelon Duck** by Rachel Cruthirds.

### *Prepare HTML5 Canvas: Markup*

---

We only need a line of markup; everything else will be in scripting.

```
<canvas id="canvasInAPerfectWorld" width="490" height="220"></canvas>
```

*Wait...* HTML5 is still new and some browsers (pssst... that means you Internet Explorer) don't support the canvas tag, so let's use this line of markup instead:

```
<div id="canvasDiv"></div>
```

### *Prepare HTML5 Canvas: Scripting*

---

To prepare our canvas, we would hope to use:

```
context =  
document.getElementById('canvasInAPerfectWorld').getContext("2d");
```

However IE doesn't know what the canvas tag means, and if we used that in our markup, IE would serve us an entrée of error. Instead we create a canvas element in JavaScript and append it to our div we called `canvasDiv`.

```
var canvasDiv = document.getElementById('canvasDiv');
```

```
canvas = document.createElement('canvas');

canvas.setAttribute('width', canvasWidth);

canvas.setAttribute('height', canvasHeight);

canvas.setAttribute('id', 'canvas');

canvasDiv.appendChild(canvas);

if(typeof G_vmlCanvasManager != 'undefined') {

    canvas = G_vmlCanvasManager.initElement(canvas);

}

context = canvas.getContext("2d");
```

For Internet Explorer compatibility we will also have to include an additional script: [ExplorerCanvas](#).

### *Create a Simple Drawing "Canvas"*

---

Before we add any options, let's tackle the basics of dynamically drawing on an HTML5 canvas. It will consist of 4 mouse events and two functions: `addClick` to record mouse data and `redraw` which will draw that data.

**Mouse Down Event:** When the user clicks on canvas we record the position in an array via the `addClick` function. We set the boolean `paint` to true (we will see why in a sec). Finally we update the canvas with the function `redraw`.

```
$('#canvas').mousedown(function(e) {

    var mouseX = e.pageX - this.offsetLeft;

    var mouseY = e.pageY - this.offsetTop;

    paint = true;

    addClick(e.pageX - this.offsetLeft, e.pageY - this.offsetTop);

    redraw();
```



```
});
```

*Mouse Move Event:* Just like moving the tip of a marker on a sheet of paper, we want to draw on the canvas when our user is pressing down. The Boolean `paint` will let us know if the virtual marker is pressing down on the paper or not. If `paint` is true, then we record the value. Then redraw.

```
$('#canvas').mousemove(function(e) {  
  
    if(paint){  
  
        addClick(e.pageX - this.offsetLeft, e.pageY - this.offsetTop,  
true);  
  
        redraw();  
  
    }  
  
});
```

*Mouse Up Event:* Marker is off the paper; `paint` boolean will remember!

```
$('#canvas').mouseup(function(e) {  
  
    paint = false;  
  
});
```

*Mouse Leave Event:* If the marker goes off the paper, then forget you!

```
$('#canvas').mouseleave(function(e) {  
  
    paint = false;  
  
});
```

Here is the `addClick` function that will save the click position:

```
var clickX = new Array();

var clickY = new Array();

var clickDrag = new Array();

var paint;

function addClick(x, y, dragging)

{

    clickX.push(x);

    clickY.push(y);

    clickDrag.push(dragging);

}
```

The `redraw` function is where the magic happens. Each time the function is called the canvas is cleared and everything is redrawn. We could be more efficient and redraw only certain aspects that have been changed, but let's keep it simple.

We set a few stroke properties for the color, shape, and width. Then for every time we recorded as a marker on paper we are going to draw a line.

```
function redraw(){

    canvas.width = canvas.width; // Clears the canvas

    context.strokeStyle = "#df4b26";

    context.lineJoin = "round";

    context.lineWidth = 5;

    for(var i=0; i < clickX.length; i++)

    {

        context.beginPath();

        if(clickDrag[i] && i){

            context.moveTo(clickX[i-1], clickY[i-1]);
```

```
        }else{  
            context.moveTo(clickX[i]-1, clickY[i]);  
        }  
        context.lineTo(clickX[i], clickY[i]);  
        context.closePath();  
        context.stroke();  
    }  
}
```

### *Simple Drawing Canvas Demo*

---

Give it a try:

Clear the canvas: [Clear](#)

### *Add Colors*

---

Let's give our user some color choices. To do so, all we need to do is add declare a few global variables and update our redraw function.

Declare four color

variables: `colorPurple`, `colorGreen`, `colorYellow`, `colorBrown` with corresponding hex color values, a variable to store the current color: `curColor`, and an array to match the chosen color when the user clicked the canvas `clickColor`.

```
var colorPurple = "#cb3594";  
var colorGreen = "#659b41";  
var colorYellow = "#ffcf33";  
var colorBrown = "#986928";  
  
var curColor = colorPurple;  
var clickColor = new Array();
```

The `addClick` function needs to be updated to record the chosen color when the user clicks.

```
function addClick(x, y, dragging)
{
    clickX.push(x);
    clickY.push(y);
    clickDrag.push(dragging);
    clickColor.push(curColor);
}
```

Since the color can vary now, we need to update the `redraw` function so it references the color that was active when the user clicked. We move the line about `strokeStyle` into the for loop and assign it to color value in the new array `clickColor` that corresponds to the user's clickage.

```
function redraw() {
    /* context.strokeStyle = "#df4b26"; */
    context.lineJoin = "round";
    context.lineWidth = 5;

    for(var i=0; i < clickX.length; i++)
    {
        context.beginPath();
        if(clickDrag[i] && i){
            contex.moveTo(clickX[i-1], clickY[i-1]);
        }else{
            context.moveTo(clickX[i]-1, clickY[i]);
        }
        context.lineTo(clickX[i], clickY[i]);
        context.closePath();
        context.strokeStyle = clickColor[i];
        context.stroke();
    }
}
```

## Demo Colors

---

Try it with color choices:

- Clear the canvas: Clear
- Choose a color: PurpleGreenYellowBrown

## Add Sizes

---

Just like we added colors, let's add some sizes to choose from: "small", "normal", "large", and "huge".

We need a couple more global variables: `clickSize` and `curSize`.

```
var clickSize = new Array();  
var curSize = "normal";
```

The `addClick` function needs to be updated to record the chosen size when the user clicks.

```
function addClick(x, y, dragging)  
{  
    clickX.push(x);  
    clickY.push(y);  
    clickDrag.push(dragging);  
    clickColor.push(curColor);  
    clickSize.push(curSize);  
}
```

Update the `redraw` function to handle the new sizes.

```
function redraw(){  
    context.lineJoin = "round";  
  
    class="highlight delete">/* context.lineWidth = 5; */  
    for(var i=0; i < clickX.length; i++)  
    {  
        context.beginPath();  
        if(clickDrag[i] && i){  
            context.moveTo(clickX[i-1], clickY[i-1]);  
        }else{  
            context.moveTo(clickX[i]-1, clickY[i]);  
        }  
    }
```

```
context.lineTo(clickX[i], clickY[i]);
context.closePath();
context.strokeStyle = clickColor[i];
context.lineWidth = radius;
context.stroke();
}
}
```

## Demo Sizes

---

Try it with different sizes:

- Clear the canvas: Clear
- Choose a color: PurpleGreenYellowBrown
- Choose a size: SmallNormalLargeHuge

## Add Tools

---

Crayon, Marker, Eraser. Three tools. Let's make them.

The two global variables we need are: `clickTool` and `curTool`.

```
var clickTool = new Array();
var curTool = "crayon";
```

The `addClick` function needs to be updated to record the chosen tool when the user clicks.

```
function addClick(x, y, dragging)
{
    clickX.push(x);
    clickY.push(y);
    clickDrag.push(dragging);

    if(curTool == "eraser"){
        clickColor.push("white");
    }else{
```

```
    clickColor.push(curColor);  
}  
clickColor.push(curColor);  
clickSize.push(curSize);  
}
```

Update the `redraw` function to handle the new tools.

```
function redraw(){  
  
    context.lineJoin = "round";  
  
    for(var i=0; i < clickX.length; i++)  
    {  
  
        context.beginPath();  
  
        if(clickDrag[i] && i){  
  
            context.moveTo(clickX[i-1], clickY[i-1]);  
  
        }else{  
  
            context.moveTo(clickX[i]-1, clickY[i]);  
  
        }  
  
        context.lineTo(clickX[i], clickY[i]);  
  
        context.closePath();  
  
        context.strokeStyle = clickColor[i];  
  
        context.lineWidth = radius;  
  
        context.stroke();  
  
    }  
  
    if(curTool == "crayon") {  
        context.globalAlpha = 0.4;  
        context.drawImage(crayonTextureImage, 0, 0, canvasWidth,  
canvasHeight);  
    }  
    context.globalAlpha = 1;  
}
```

## Demo Tools

---

Try it with different tools:

- Clear the canvas: Clear
- Choose a color: PurpleGreenYellowBrown
- Choose a size: SmallNormalLargeHuge
- **Choose a tool:** CrayonMarkerEraser

## Add Outline

---

Coloring books provide an outline of something to color: a cute puppy or a hopping bunny. I chose a watermelon duck.

First declare a variable `outlineImage`.

```
var outlineImage = new Image();
```

Load the outline image.

```
function prepareCanvas() {  
  
    ...  
  
    outlineImage.src = "images/watermelon-duck-outline.png";  
}
```

Update the `redraw` function to draw the outline image using the canvas context's `drawImage` method. Its parameters are the image object we loaded, the position we want to draw the image, and the dimensions of the image.

```
function redraw() {  
  
    ...  
  
    context.drawImage(outlineImage, drawingAreaX, drawingAreaY,  
        drawingAreaWidth, drawingAreaHeight);  
}
```



```
}
```

## Demo Outline

---

Give it try:

- Clear the canvas: Clear
- Choose a color: PurpleGreenYellowBrown
- Choose a size: SmallNormalLargeHuge
- Choose a tool: CrayonMarkerEraser

## Final Details

---

We are almost there! These last details are optional: restrict the drawing area to a rectangle on the canvas and use the rest of the canvas for our GUI (aka buttons).

Let's mask the canvas so all drawing is within the drawing area. We use the `clip` method with the `save` and `restore` methods. This method is not currently supported by Internet Explorer.

```
function redraw()

{
    ...

    context.save();
    context.beginPath();
    context.rect(drawingAreaX, drawingAreaY, drawingAreaWidth,
drawingAreaHeight);
    context.clip();

    var radius;
    var i = 0;
    for(; i < clickX.length; i++)
    {
        ...
    }
    context.restore();
    ...
} // end redraw function
```

The last detail is to move all those buttons on our canvas. It involves loading images and displaying them based on our user interaction. I used standard JavaScript techniques so I won't bore you with the details (but it is included in the source code if you are interested). And there you have it!

### *Complete Demo*

---

Now that we have created an HTML5 canvas drawing app, let's take a break and draw!

### *Download Source Code*

---

- Download HTML5 Canvas Drawing App (zip format): [html5-canvas-drawing-app.zip](#)

~~~ End of Article ~~~



Session 18: HTML5 Web Storage

HTML5 localStorage for Offline Web Applications

Source <http://www.linuxforu.com/2012/04/html5-localstorage-offline-web-applications/>

This article explains the local (client-side, browser) storage feature in HTML 5.

Since the introduction of HTML5, the technology is making an equal impact in the desktop and mobile space. It has completely changed what is possible with a Web interface. The new features are so powerful they bring Web pages closer to Apps. The new features include a plugin-free paradigm, with the introduction of tags like `<video>`, semantic markup like `<header>`, new form elements like `email`, and client-side storage. There is a constant drive to make browser-specific incompatibilities disappear.

The need for client-side storage has been felt with the increasing use of Web for e-commerce, with the aim to enhance the user experience by storing user preferences, shopping cart items and user-session information. Cookies have served Web developers for long, but have size limitations (4 KB each) and need to be transferred from client to server. The importance of offline storage has increased with Web app development using HTML5. There are times when the user is not connected to the Internet (e.g., composing an email in an airplane using Internet Web app to send later), or when data needs to be saved across sessions (e.g., a Web gaming app needs to store the session state, like the board positions for chess, so that the game can be resumed later).

With HTML5, two new objects are introduced for client-side storage. They are `localStorage` and `sessionStorage`. We will focus on `localStorage` in this article. New object `sessionStorage` is to store data for one session. Before introduction of `localStorage` in HTML5, developers had to choose from various vendor- or browser-specific options like Google Gears, `userData` in IE, and Local Shared Objects used in Adobe Flash Player.

An introduction to localStorage

`localStorage` is also called Web Storage or DOM storage. It is a database that can store only key-value pairs, and the value data-type should always be a string. At first, this might seem a serious limitation, but using along with JavaScript arrays and JSON we can achieve quite a bit. The good news is that this feature is supported in the latest versions of all browsers, both on desktop and mobile environments (Android and iOS). Also, the developer gets a huge 5 MB of storage per domain.

The code snippet below checks if the browser supports `localStorage`. The code in this article has been tested on Google Chrome 17.0.963.56 and Firefox 11.0. There is a small difference in behavior when the code below is executed in IE: if the HTML file is opened locally

(file:///C:/Web/mylocaltodo.html), IE throws an error — but if the same file is hosted on a server (even localhost), it just works!

In the following code, `localStorage` is an object under `window` and can be referenced via the `window` object, or directly as `localStorage`.

```
function browserCheck() {
    if (typeof(localStorage) == 'undefined') {
        alert('Your browser does not support HTML5 localStorage. Try upgrading.');
```

localStorage support on browsers/platforms

This table lists the minimum version of each browser required to support `localStorage`:

| IE | Firefox | Safari | Chrome | Opera | iPhone | Android |
|------|---------|--------|--------|-------|--------|---------|
| 8.0+ | 3.5+ | 4.0+ | 4.0+ | 10.5+ | 2.0+ | 2.0+ |

Operations on localStorage

The methods of `localStorage` include `setItem()`, `getItem()`, `removeItem()` and `clear()`.

Here is sample code to add a record to `localStorage` using `setItem()`:

```
// max limit reached exception
try {
    localStorage.setItem(itemId, "Sample Value");
} catch (e) {
    if (e == QUOTA_EXCEEDED_ERR) {
        alert('Quota exceeded!');
    }
}
```

We will use `getItem()` and `removeItem()` in the Web app that we will develop. Note that `localStorage.clear()` removes all entries from the database, so use it with caution!

Debugging localStorage

The quick and easy way to view the `localStorage` database is in Chrome, as part of Developer Tools. Select the wrench menu at the top-right corner, go to *Tools* → *Developer Tools*. Click the *Resources* tab, select *Local Storage* and then *Local Files*. The right pane should display key-value pairs (Figure 1), if the database has data. For debugging in Firefox, you need the Firebug extension.

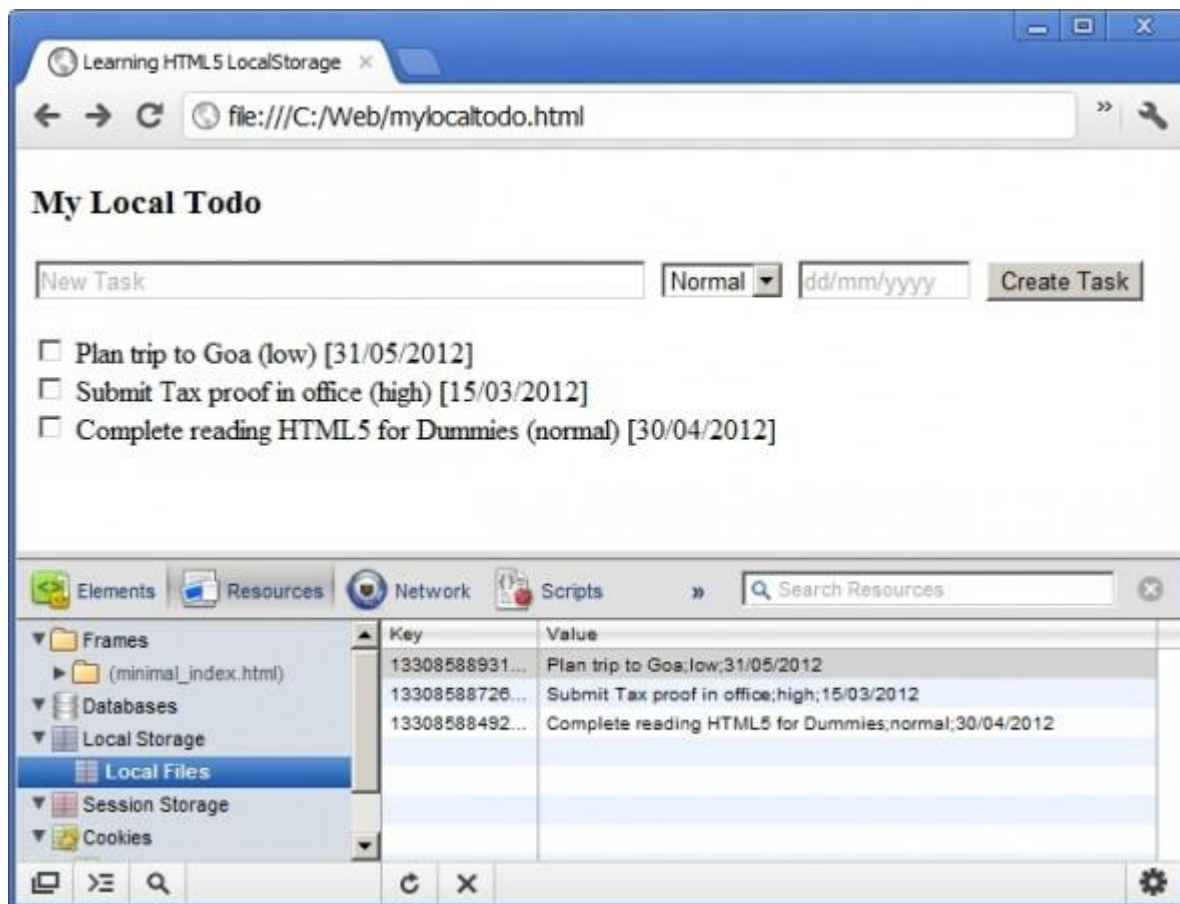


Figure 1: Database view

MyLocalTodo: An example of the use of localStorage

Let's try implementing a My Local Todo Web application, which can work offline, with minimal code. Besides `localStorage`, we will also use HTML5 features such as `placeholder` and `required`.

Application UI



Figure 2: Task listing

In this simple to-do application, the first (text) field “New Task” is the task description; the second is task priority, and third the due date. Only the first is mandatory. If the user clicks “Create Task” without entering it, the browser shows a tooltip below the text field (Figure 3).

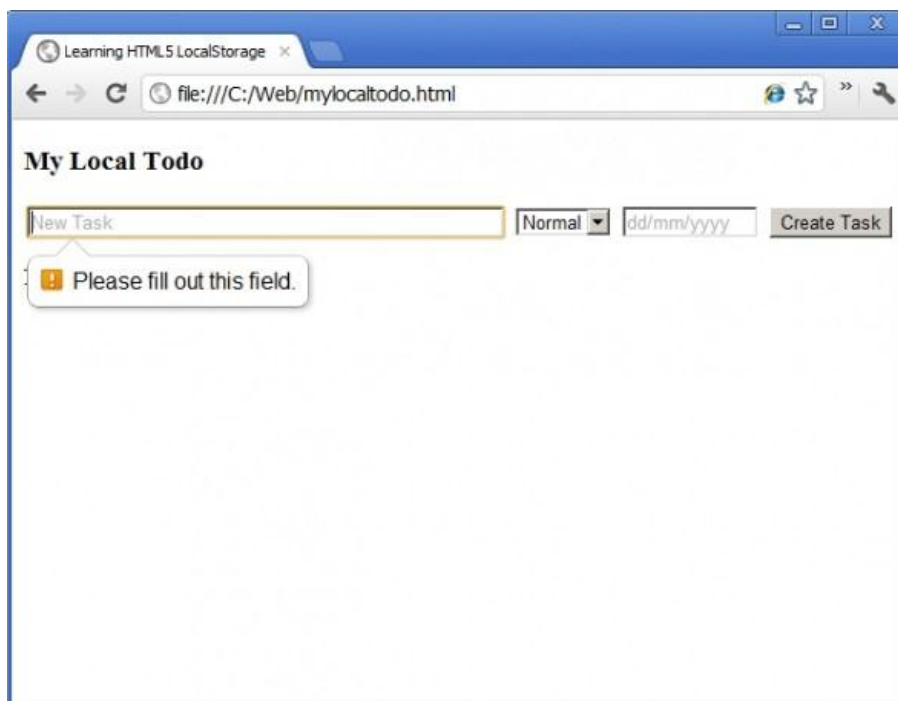


Figure 3: Tooltip notification for mandatory entry

The default task priority is “Normal” and the due date is today.

```

1  <body onLoad="browserCheck();getTasks();">
2      <h3>My Local Todo</h3>
3      <form name="todoForm" onSubmit='addTask();'>
4          <input type="text" size="50" name="task" placeholder="New Task" value=""
5          autofocus required>
6              <select id="priority">
7                  <option value="high">High</option>
8                  <option value="normal" selected>Normal</option>
9                  <option value="low">Low</option>
10             </select>
11             <input type="text" size="10" name="duedate" placeholder="dd/mm/yyyy">
12             <input type="submit" value="Create Task">
13         </form>
14     <p id="theTaskList">Nothing much to do!</p>
15 </body>

```

- Line 1: We check if the browser supports `localStorage`. If not, display an alert. We also display tasks already present in `localStorage`.
- Line 3-12: Form with fields to accept task text, priority and due date. Form to be submitted via the “Create Task” button, which will add the task to `localStorage`.
- Line 4: Compared to older HTML text fields, you find three new attributes: `placeholder`, `autofocus` and `required`. The first attribute displays a hint to the user in grey text, about the expected purpose/format of the input. The `autofocus` attribute places the cursor in the text field before and after form submission. The `required` attribute makes input mandatory before the form can be submitted.
- Line 5-9: Task priorities drop-down; “High”, “Normal” (default) and “Low” entries.
- Line 10: Text field for due date; format suggested via `placeholder`.
- Line 11: Form submission button “Create Task”.
- Line 13: Space to list existing tasks. If no tasks are found, “Nothing much to do!” is displayed.

Operations and attributes of `localStorage`

There are three functions that perform operations on `localStorage`:

- `addTask()`: Called when the user enters task details and clicks “Create Task”.
- `getTasks()`: Retrieves all records from `localStorage`; called when the HTML page is loaded, and when a task is added, to update the list.
- `deleteTask()`: Called when the user marks the task complete (ticks the check-box). The task is removed from `localStorage`.

```

1  function addTask()
2  {
3      var values = new Array();
4      var newDate = new Date();
5      var itemId = newDate.getTime();
6      var duedate = "";
7      if (!document.forms["todoForm"]["duedate"].value)
8      {
9          duedate = fillDuedate();
10     }
11     else
12     {
13         duedate = document.forms["todoForm"]["duedate"].value;
14     }
15     values.push(document.forms["todoForm"]["task"].value);
16     values.push(document.forms["todoForm"]["priority"].value);
17     values.push(duedate);
18     try {
19         localStorage.setItem(itemId, values.join(';'));
20     } catch (e) {
21         if (e == QUOTA_EXCEEDED_ERR) {
22             alert('Quota exceeded!');
23         }
24     }
25     getTasks();
26 }

```

- Line 3: Each record is stored as a key-value pair; fields are separated by a semicolon (;). A `values()` array is used to store each task record.
- Line 4-5: Key is generated using the current time in milliseconds, on the assumption that there will always be at least a few milliseconds delay between the creation of two tasks — so, this is always unique.
- Line 7-14: Check if the due date is entered; if not, insert today's date in dd/mm/yyyy format using the function `fillDuedate()`. If the user entered a due date, use it as-is, without validation, since our focus is on trying `localStorage`.
- Line 18-24: Insert the new record in the `localStorage` database. If the 5 MB quota allotted by the browser is exceeded, an exception is thrown. Better storage management ideas are needed!
- Line 25: Fetch tasks already in `localStorage` and display, including the newly created one.

```

1  function getTasks()
2  {
3      var i = 0;

```



```

4     var currentTaskInHTML = "";
5     var allTasksInHTML = "";
6     if (localStorage.length == 0)
7     {
8         document.getElementById('theTaskList').innerHTML = "Nothing much
to do!";
9     }
10    else
11    {
12        for (i=0; i<localStorage.length; i++)
13        {
14            var itemKey = localStorage.key(i);
15            var values = localStorage.getItem(itemKey);
16            values = values.split(";");
17            currentTaskInHTML = '<input type="checkbox\"
18                                name="\"task\" value="\"' + itemKey + '\"
19                                onClick="\"deleteTask(' + itemKey + ')\"/> ' +
20                                values[0]+ ' (' +values[1]+ ' ) [' +values[2]+
21                                ']/><BR>';
22            allTasksInHTML += currentTaskInHTML;
23        }
24        document.getElementById('theTaskList').innerHTML =
allTasksInHTML;
25    }
26 }

```

- Line 6: If there are no tasks in `localStorage`, it displays “Nothing much to do!”.
- Line 12-23: Iterates through each record in the database, prepares HTML for check-box and task details.
- Line 12: `localStorage.length` returns the number of records present in the database.
- Line 14: Fetches the unique key using the function `localStorage.key()`.
- Line 15: Fetches the complete record — fields separated by semi-colon, using the unique key.
- Line 16: Extracts fields, separating them using `split()`.
- Line 17-21: Creates an HTML form element for each task (check-box followed by task details). Inserts the function `deleteTask()` as part of the actions when the user completes a task and ticks the check-box. To delete a task from the database, we need the unique key — which is the value of each check-box.
- Line 22: Appends tasks, and creates one HTML string that will be output below the “New Task” form, replacing the previous task list with the new one using the `innerHTML` attribute.

```

1  function deleteTask(key)
2  {
3      localStorage.removeItem(key);
4      getTasks();
5  }

```

- Line 3: When the user clicks the check-box to mark it as complete, the record key is passed `todeleteTask()`, which uses `localStorage.removeItem()` to remove the record.
- Line 4: The list of pending tasks is refreshed after the deletion.

Extending MyLocalTodo to a full application

Here are some ideas that could be implemented to build on the skeleton app:

- Add color to tasks: Differentiate tasks with different priorities with colors. Tasks past the due date can be in RED.
- Make the due date “human-friendly”: Add flexibility to accept human-friendly due dates like “today”, “tomorrow”, “by next week”, “end of April”, etc.
- Export/import tasks: Due to `localStorage` limitations, the task database is restricted to one browser. Extend the app to export records as a CSV file to make tasks portable across browsers, and for backup purposes.
- Import CSV: The ability to import a CSV file into `localStorage` could allow the import of tasks from Outlook, Google Tasks, etc.
- Integration with Phonebook: A friendlier to-do app could integrate with the Phonebook application. Tasks that involve communication (a phone call or email) can fetch the phone number/email address and add it to the task, to enable calling or emailing the person. For example: for a task like “call @Rohit on Sunday”, it displays “call Rohit (9911991199) on Sunday”, so that you can dial by clicking the number.
- Add tags to group tasks: Each task can have a tag such as “personal”, “work”, “call” or “email”, to group the related tasks to be performed together.

Develop other applications using localStorage

Some quick ideas:

- Phonebook: The MyLocalTodo example app can be modified to develop a Phonebook application.
- Project Time Tracker: You can [check out the monkeeTime application](#), which is a Project Time Tracker.

~~~ End of Article ~~~



## HTML5 - Web Storage (DOM Storage)

**Source**

<http://yinyangit.wordpress.com/2012/01/27/html5-huong-dan-ve-web-storage-dom-storage/>

HTML5 provides a data storage features in the client with limited capacity greater than cookies. This feature is called the Web Storage, and is divided into two objects are **localStorage** and **sessionStorage**. This article will help you understand the full knowledge of the two objects used in web programming.

### Introduction

Currently, each cookie only allows storage of up to 4KB and a few dozen cookies for a domain. So only use cookies to store information simple and concise as email, username, helps users automatically log into the site. This makes the site to improve performance by caching at the client data can hardly be realized.

The emergence of Web Storage as a focus for a great web applications have interoperability and load immediate data on the browser. One is how much more efficient transmission over a network can be substantially reduced. For example, an application online reference books, the books have been investigated will be saved on the user's machine. When you need to check again, your users will not need to connect to the server to reload the old data.

With the web application can compact the database, the programmer can make the cache 'stealth' database to the client and then the user can comfortably without lookup request to the server.

### Storage Support Web Test

The following table is abridged from: <http://dev-test.nemikor.com/web-storage/support-test/>

| Browser     | localStorage | sessionStorage |
|-------------|--------------|----------------|
| Chrome 5.0  | 2:49 M       | unlimited      |
| Chrome 6.0  | 2:49 M       | 2:49 M         |
| Firefox 3.0 | none         | unlimited      |
| Firefox 3.5 | 4.98 M       | unlimited      |

| Browser                  | localStorage | sessionStorage |
|--------------------------|--------------|----------------|
| IE 7.0                   | none         | none           |
| IE 8.0                   | 4.75 M       | 4.75 M         |
| IE Platform Preview 10.0 | 4.75 M       | 4.75 M         |
| 7.0 IEMobile             | none         | none           |
| Opera 10:54              | none         | none           |
| Opera 10.63              | unlimited    | 4.92 M         |
| Opera 11:00              | 380 k        | 504 k          |
| Opera 11:51              | unlimited    | 4.92 M         |
| Opera 11:52              | 7:42 M       | 4.92 M         |
| Opera 11.60              | 1.85 M       | 4.92 M         |
| Opera 11.61              | unlimited    | 4.86 M         |
| Opera Mini 5.18741       | none         | none           |
| Opera Mobile 11:00       | unlimited    | 504 k          |
| Opera Tablet 11:50       | 380 k        | 504 k          |
| Safari                   | 2:49 M       | unlimited      |
| Safari 4.0               | unlimited    | unlimited      |
| Safari 4.1               | none         | none           |
| Safari 5.0               | 2:49 M       | unlimited      |

## Storage Interface

Storage interface

```
{  
  readonly attribute unsigned long length;  
  DOMString? key (unsigned long index);  
  DOMString getter GetItem (DOMString key);  
  SetItem creator void setter (key DOMString, DOMString value);  
  void RemoveItem Deleter (DOMString key);  
  void clear ();  
};
```

As you can see, the data is stored in the storage string with the other data types such as integer, real number, and bool. You need to perform type conversions. Each Storage object is a list of key / value pairs, this object includes properties and methods:

- Length: number of key / value pairs in the object.
- Key (n): Returns the name of the nth key in the list.
- GetItem (key): Returns the value associated with key.
- SetItem (key, value): More or assign a key / value pairs to the list.
- RemoveItem (key): remove key / value pairs from the list.
- Clear: clear all data in the list.

In addition, Storage objects can also be accessed via the key attribute is the list.

For example:

```
localStorage.abc = "123";  
// Equivalent to:  
// localStorage.setItem ("abc", "123");
```

## Storage and Storage Local Session

Two objects are created from Storage interface, you use two objects in JavaScript through two variables are made available as window.localStorage and window.sessionStorage. Two advantages they bring are:

- Ease of use: you can access two data objects through properties or methods. Data is stored for each key / value pairs and do not need any work or preparing any initialization.
- Large storage capacity: Depending on your browser, you can store from 5MB to 10MB per domain. According to Wikipedia, this figure is 5MB in Mozilla Firefox , Google Chrome, and Opera, 10MB of Internet Explorer.

### Scope:

**-sessionStorage** : limited to a window or browser tab. A web page is opened in two cards of the same browser cannot access data from each other. So, when you close your web page, the data stored in the current sessionStorage also is deleted.

**-localStorage** : can access each other between the browser window. Data will be stored without time limit.

**For localStorage:**

"Each domain and subdomain has its own separate local storage area. Domains can access the storage areas of subdomains, and subdomains can access the storage areas of parent domains. For example, localStorage ['example.com'] is accessible to example.com and any of its subdomains. The subdomain localStorage ['www.example.com'] is accessible to example.com, but not to other subdomains, word such as mail.example.com.

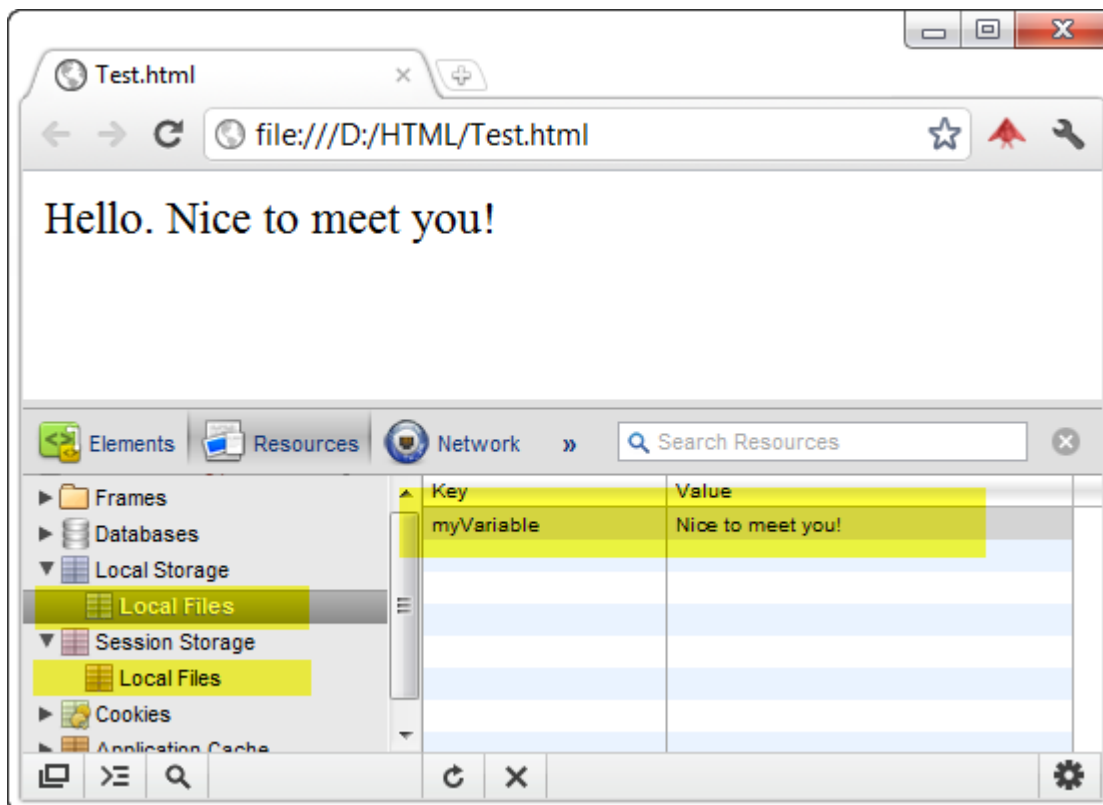
[http://msdn.microsoft.com/en-us/library/cc197062 \(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/cc197062 (VS.85).aspx)

**Use**

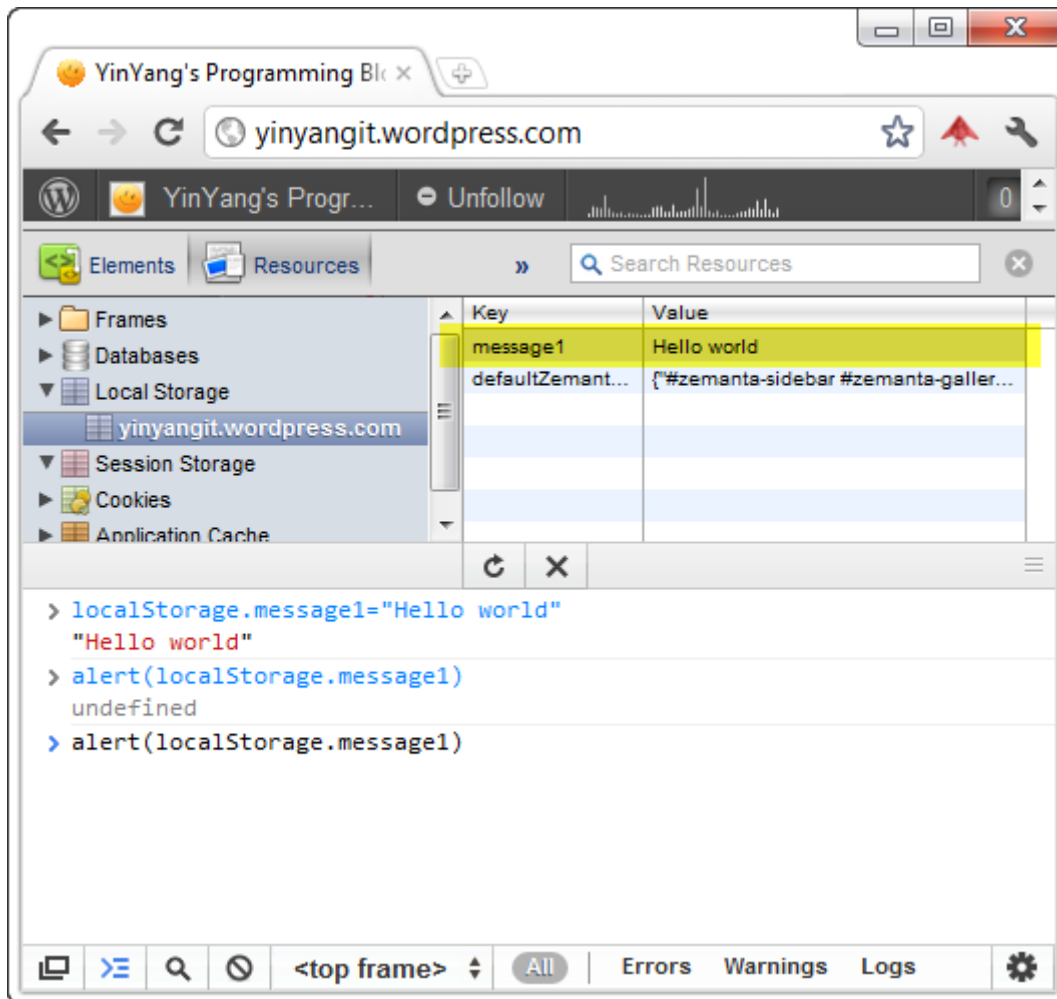
You can create an HTML file with the following content to run in your browser. Here I use Chrome as it is provided with viewing window in Google Chrome Resources Developer Tools (Ctrl + Shift + I).The contents of the HTML file as follows:

```
01 <!DOCTYPE html>
02 <html>
03 <body>
04
05 <script type="text / javascript">
06
07 sessionStorage.myVariable = "Hello.";
08 localStorage.myVariable = "Nice to meet you!";
09
10 document.write (sessionStorage.myVariable);
11 document.write (localStorage.myVariable);
12
13 </script>
14
15 </body>
16 </html>
```

The results show:



In the view Resources, you can open the JavaScript console to type commands to interact with existing website. For example, here I add the new value into the localStorage and use alert () to show them up.



## Storage event

Window Object in JavaScript provides an event as "storage". Event is triggered when storageArea changed.

interface **StorageEvent** : Event

```
{
  readonly attribute DOMString key ;
  readonly attribute DOMString? OldValue ;
  readonly attribute DOMString? newValue ;
  readonly attribute DOMString url ;
  readonly attribute Storage ? storageArea ;
};
```

This Event cannot work when you view the HTML file in the local computer and is active only in the window / card. That is when you open multiple browser windows to access the same domain, if you change an object the Storage window, the remaining windows will be activated event "storage" and the current window will not nothing happens.

```
01 <!DOCTYPE html>
```

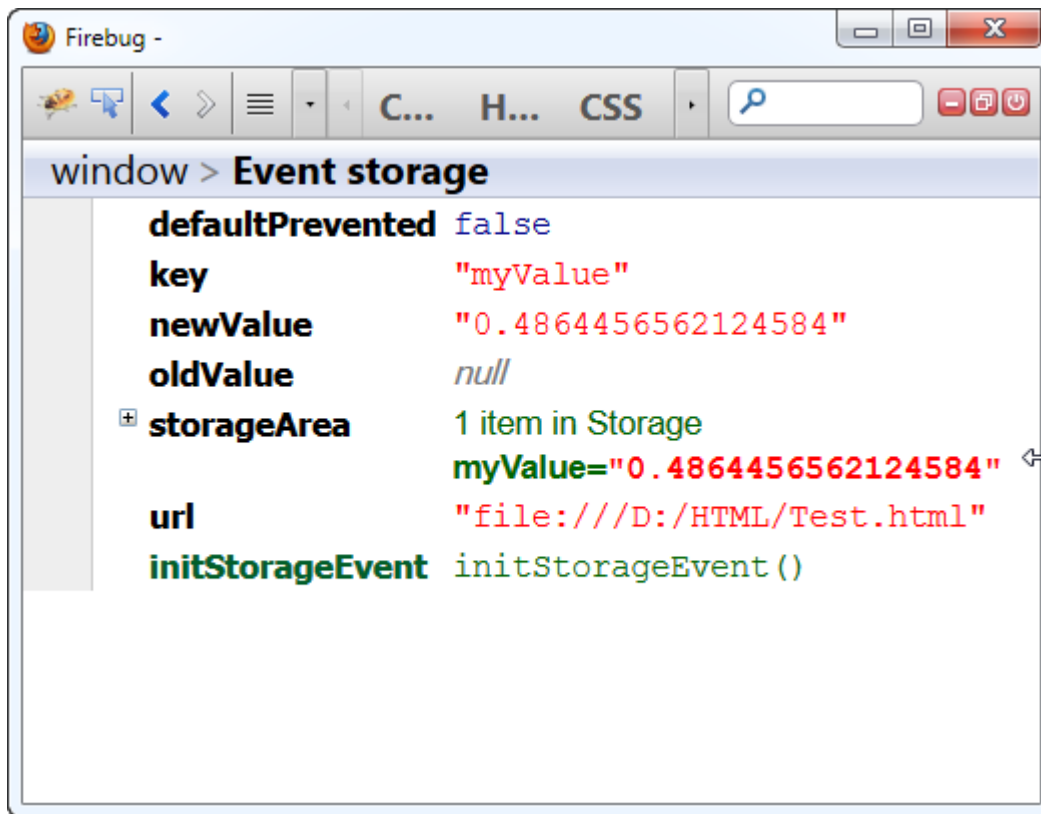
```
02 <html>
```

```
03 <body>
```



```
04
05 < button onclick = "changeValue ();" > Change Value </ button >
06
07 < script type = "text / javascript" >
08 localStorage.clear ();
09 console.log (localStorage);
10 if (window.addEventListener)
11     window.addEventListener ('storage', storage_event, false);
12 else if (window.attachEvent) / / IE
13     window.attachEvent ('onstorage', storage_event, false);
14
15 storage_event function (event) {
16     console.log (event);
17 }
18
19 changeValue function ()
20 {
21     localStorage.myValue = Math.random ();
22 }
23
24 </ script >
25
26 </ body >
27 </ html >
```

The value of the object StorageEvent in the above example (already reduced):



## Add methods to Storage

The method of storage cannot satisfy your requirements, so you can add a few new methods to use when needed. For example I will add a method `search ()` to filter out the value stored from the key search.

```

01 Storage.prototype.search = function (keyword)
02 {
03     var array = new Array ();
04     var re = new RegExp (keyword, "gi" );
05     for ( var i = 0; i < this . length; i + +)
06     {
07         var value = this . GetItem ( this . key (i));
08         if (value.search (re)> -1)
09             Array.push (value);
10     }
11     return array;

```

```
10 }
```

This approach uses regular expressions to search for the two options "g" (to search the entire string) and "i" (not case sensitive). String.search method () returns the position of first character matched the search keywords, in contrast to -1 if not found.

We have complete source code for this example:

```
01 <! DOCTYPE html>
02 <html>
03 <body>
04
05 <Script type = "text / javascript" >
06 Storage.prototype.search = function (keyword) {
07     var array = new Array ();
08     var re = new RegExp (keyword, "gi" );
09     for ( var i = 0; i < this . length; i + +) {
10         var value = this . GetItem ( this . key (i));
11         if (value.search (re)> -1)
12             Array.push (value);
13     }
14     return array;
15 }
16
17 localStorage.clear ();
18
19 localStorage.key1 = "Abbey" ;
20 localStorage.key2 = "Abbie" ;
21 localStorage.key3 = "Abby" ;
22 localStorage.key4 = "Abe" ;
23 localStorage.key5 = "Abie" ;
24 localStorage.key6 = "Acton" ;
```

```
25 localStorage.key7 = "Adair" ;  
26  
27 document.write (localStorage.search ( "ab" ));  
28  
29 </ Script>  
30  
31 </ Body>  
32 </ Html>
```

**Results:**

Abby, Abie, Abbie, Abe, Abbey

Also this way, you can add methods to store the JSON object or other data types.  
Add:

- It is another object globalStorage but this is not the instance of the Storage interface. Currently in the popular browsers, only Firefox support this object.

~~~ End of Article ~~~



Developer's Guide - Client-side Storage (Web Storage)

Source

<https://developers.google.com/web-toolkit/doc/latest/DevGuideHtml5Storage#UsingStorage>

Starting with GWT 2.3, the GWT SDK provides support for HTML5 client-side storage, also known as "web storage". Using the GWT library's Storage capability enables your web application to take advantage of these storage features when running in an HTML5-compliant browser.

What is HTML5 Storage?

The HTML5 (web) storage spec is a standardized way of providing larger amounts of client-side storage and of more appropriately "partitioning" session storage and locally persistent storage. The HTML5 spec also provides for storage events to be generated and handled by interested listeners. The full impact of these features provided by HTML5 storage can best be seen by looking at client-side storage in the non-HTML5 world.

Without HTML5, client-side storage for web applications is limited to the tiny storage provided by cookies (4KB per cookie, 20 cookies per domain) unless proprietary storage schemes are used, such as Flash local shared objects or Google Gears. If cookies are used they provide both session and locally persistent storage at the same time, and are accessible by all browser windows and tabs. Domain cookies are sent with every request to that domain, which consumes bandwidth. The "mechanics" of processing cookies are also a bit cumbersome.

In contrast, HTML5 storage provides a much larger initial local storage (5MB per domain), unlimited session storage (limited only by system resources) and successfully partitions local and session storage so that only the data you want to persist is persisted in local storage and data you want to be transient stays transient. Moreover, session storage is only accessible from its originating tab or window; it is not shared between all browser windows and tabs. Accessing session and local storage is simple, consisting in simple reads and writes of key-value strings. Finally, local and session storage are client-side only; they are not sent with requests.

Why Use HTML5 Storage?

With HTML5 local storage, a larger amount of data (initially, 5MB per application per browser) can be persistently cached client-side, which provides an alternative to server downloads. A web application can achieve better performance and provide a better user experience if it uses this local storage. For example, your web application can use local storage to cache data from RPC calls for faster startup times and for a more responsive user interface. Other uses include saving the application state locally for a faster restore when the user re-enters the application, and saving the user's work if there is a network outage, and so forth.

Note: The 5MB maximum applies to local storage only, not to session storage, which is limited only by system memory.

Here is a short list of some of the benefits and uses of local storage:

- Reduce network traffic
- Significantly speed up display times
- Cache data from RPC calls
- Load cached data on startup (faster startup)
- Save temporary state
- Restore state upon app reentry
- Prevent work loss from network disconnects

Note: unlike cookies, items in Storage are not sent along in requests, which helps reduce network traffic.

Details You Should Know About HTML5 Storage

To use HTML5 storage features, you need to know about lifespan (persistence) of local and session storage, about their scope--which windows and tabs can access the storage, and which tabs and windows can listen for storage events.

LocalStorage and SessionStorage

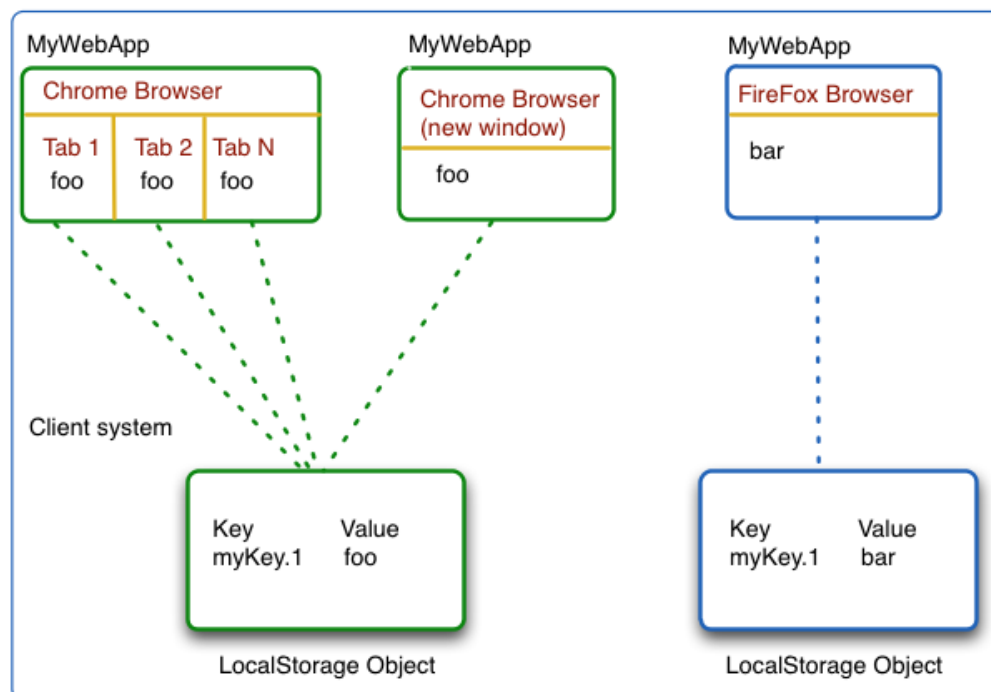
HTML5 Web Storage defines two types of key-value storage types: sessionStorage and localStorage. The primary behavioral difference is how long the values persist and how they are shared. The following table shows the differences between the two types of storage.

| Storage Type | Max Size | Persistence | Availability to other Windows/tabs | Data Type Supported |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|------------------------------------------------------------------------|---------------------------------|
| LocalStorage | 5MB per app per browser.
According to the HTML5 spec , this limit can be increased by the user when needed; however, only a few browsers support this | On disk until deleted by user (delete cache) or by the app | Shared across every window and tab of one browser running same web app | String only, as key-value pairs |
| SessionStora | Limited only by | Survives only as | Accessible only within | String only, as key- |

| | | | | |
|----|---------------|---------------------------------------------|--------------------------------------|-------------|
| ge | system memory | long as its
originating
window or tab | the window or tab that
created it | value pairs |
|----|---------------|---------------------------------------------|--------------------------------------|-------------|

How Local Storage Is Shared by the Browser

One LocalStorage per web application, with a max size of 5MB, is available for a given browser and is shared by all windows and tabs of that browser. For example, suppose you have MyWebApp running in a Chrome browser on the client. If you run MyWebApp in multiple tabs and windows, they all share the same LocalStorage data, subject to a max limit of 5MB. If you were to then open that same application in another browser, say FireFox, then the new browser would get its own LocalStorage to share with all its own tabs and windows. This is shown in the following figure:



Local Storage is String Storage

HTML5 local storage saves data in string form as key-value pairs. If the data you wish to save is not string data, you are responsible for conversion to and from string when using LocalStorage. For proxy classes used with the GWT RequestFactory, you can use `RequestFactory#getSerializer()` to do string serializing. For non-proxy objects, you could use [JSON stringify and parse](#).

Note: Just like cookies, LocalStorage and sessionStorage can be inspected using browser tools such as Developer Tools in Chrome, Web Inspector in Safari and so forth. These tools allow a user to remove storage values and see what values are being recorded by a web site the user is visiting.

LocalStorage is Not Secure Storage

HTML5 local storage saves data unencrypted in string form in the regular browser cache. It is not secure storage. It should not be used for sensitive data, such as social security numbers, credit card numbers, logon credentials, and so forth.

How Storage Events Work

When data is added to, modified, or removed from LocalStorage or SessionStorage, a StorageEvent is fired within the current browser tab or window. That Storage event contains the storage object in which the event occurred, the URL of the document to which this storage applies, and both the old and the new values of the key that was changed. Any listener registered for this event can handle it.

Note: Although the HTML5 spec calls for Storage events to be fired in all tabs of the same browser or all windows of the same browser, few browsers currently implement this.

HTML5 Storage Support in GWT

GWT support for the HTML5 storage feature consists of the following:

- `com.google.gwt.storage.client.Storage` (required import)
- `LocalStorage` (local storage)
- `SessionStorage` (session storage)
- `StorageEvent` (event generated by session or local storage changes)
- `StorageEvent.Handler` (interface for storage event handlers)
- `StorageMap` (exposes the Storage object as a standard Map)

Syntactic details for these can be found in the Storage feature javadoc.

How to Use HTML5 Storage in Your Web Application

You get the storage object by invoking `Storage.getLocalStorageIfSupported()` or `Storage.getSessionStorageIfSupported()`, depending on the type of storage you want to access. Because your web app might be accessed from a browser that does not support HTML5, you should always check before accessing any of the HTML5 storage features. If the storage feature is supported, you get the storage object and then write data to it or read data from it, depending on your needs. If you want to delete one key-value pair from the storage, you can do that or you can clear all of the data from the storage object.

1. Check for browser support
2. Get the Storage object for your browser
3. Read data from Storage
4. Write data to Storage

5. Delete data from Storage
6. Handle Storage Events

Checking for Browser Support

GWT provides a simple way to determine whether the browser supports HTML5 storage--a built-in check when you get the storage object. You use `Storage.getLocalStorageIfSupported()` or `Storage.getSessionStorageIfSupported()`, depending on which type of storage you want to use. The storage object is returned if the feature is supported, or, if not, null is returned.

```
import com.google.gwt.storage.client.Storage;
private Storage stockStore = null;
stockStore = Storage.getLocalStorageIfSupported();
```

Getting the Storage Object

If the browser supports HTML5 storage, the `Storage.getLocalStorageIfSupported` method creates the storage object if it doesn't exist yet, and returns the object. If the storage object already exists, it simply returns the already-existing object. If the browser doesn't support HTML5 storage, this returns null. The `Storage.getSessionStorageIfSupported()` method works in the same way.

Getting the storage object and checking for browser support of HTML5 storage are done at the same time, so the code snippet for doing this should look familiar:

```
import com.google.gwt.storage.client.Storage;
private Storage stockStore = null;
stockStore = Storage.getLocalStorageIfSupported();
```

Reading Data from Storage

Data is stored as key-value string pairs, so you need to use the key to get the data. You either have to know what the key is, or you'll need to iterate through the storage using indexes to get keys. Picking good naming conventions for keys can help, and the use of `StorageMap` can be useful as well. If the data needs to be converted from string, you are responsible for doing that.

The following snippet shows iteration through the contents of storage, with each item in storage then being written to a separate row in a `FlexTable`. For simplicity, this assumes all of storage is used only for that `FlexTable`.

```
import com.google.gwt.storage.client.Storage;

private FlexTable stocksFlexTable = new FlexTable();
private Storage stockstore = null;
```

```
stockStore = Storage.getLocalStorageIfSupported();
if (stockStore != null){
    for (int i = 0; i < stockStore.getLength(); i++)
    {
        String key = stockStore.key(i);
        stocksFlexTable.setText(i+1, 0, stockStore.getItem(key));
        stocksFlexTable.setWidget(i+1, 2, new Label());
    }
}
```

Using StorageMap to do a Quick Check for Specific Key or Value

If you want to quickly check whether a specific key or a specific value is present in the storage, you can use the `StorageMap` object by supplying the storage to the `StorageMap` constructor, then using its `containsValue()` or `containsKey()` methods.

In the following snippet, we use a `StorageMap` to see if a certain value is already in the storage, and if it is not yet stored, we write the data to storage.

```
stockStore = Storage.getLocalStorageIfSupported();
if (stockStore != null)
{
    stockMap = new StorageMap(stockStore);
    if (stockMap.containsValue(symbol) != true){
        int numStocks = stockStore.getLength();
        stockStore.setItem("Stock."+numStocks, symbol);
    }
}
```

Writing Data to Storage

To write data, you supply a key name and the string value you wish to save. You can only write string data, so you need to do any conversions from other data types or from objects. (If the object is a proxy used with the GWT RequestFactory, you can use `RequestFactory#getSerializer()` to do string serializing. For non-proxy objects, you could use [JSON stringify and parse](#).)

Judicious use of naming conventions can help with processing storage data. For example, in a web app named `MyWebApp`, key-value data associated with rows in a UI table named `Stock` could have key names prefixed with `MyWebApp.Stock`.

In the following snippet, which is part of an Add button click handler, a text value is read from a textbox and saved, with the key name concatenated from a prefix and the current number of items in the storage.

```
import com.google.gwt.storage.client.Storage;

final String symbol = newSymbolTextBox.getText().toUpperCase().trim();
stockStore = Storage.getLocalStorageIfSupported();
if (stockStore != null)
```

```
{  
    int numStocks = stockStore.getLength();  
    stockStore.setItem("Stock."+numStocks, symbol);  
}
```

Deleting Data from Storage

You can delete a single key-value pair of data from the storage or you can delete all the data all at once.

Deleting a Specific Key-Value Pair

If you want to delete specific piece of data and you know the key name, you simply supply the key name to the `removeItem` method like this: `myStorage.removeItem(myKey)` ;

If you don't know the key, or need to process a list of keys, you can iterate through the storage using the `key` method, like this: `myStorage.key(myIndexValue)` ;

Clearing the Entire Storage

To clear the storage used by your web app, invoke the `clear()` method, like this: `myStorage.clear()` ;

The following sample snippet provides an example of one way to integrate this method with a UI, in this case a FlexTable that displays items from the storage. The user clears the UI and the storage by clicking on a Clear All button. In the button-click handler we just use the count of items in the storage to iterate through and remove rows from the UI, and when that is done, we delete all the storage data. (To keep things simple, we used the storage only for populating the FlexTable.)

```
import com.google.gwt.storage.client.Storage;  
import com.google.gwt.event.dom.client.ClickEvent;  
import com.google.gwt.event.dom.client.ClickHandler;  
import com.google.gwt.user.client.ui.Button;  
import com.google.gwt.user.client.ui.FlexTable;  
import com.google.gwt.user.client.ui.Widget;  
  
// Listen for mouse events on the Clear all button.  
clearAllButton.addClickHandler(new ClickHandler()  
{  
    public void onClick(ClickEvent event)  
    {  
        // note that in general, events can have sources that are not Widgets.  
        Widget sender = (Widget) event.getSource();  
        //If HTML5 storage is supported, clear all rows from the FlexTable UI,  
        //then clear storage  
        if (sender == clearAllButton)  
        {  
            stockStore = Storage.getLocalStorageIfSupported();  
            if (stockStore !=null)  
            {  
                for (int ix =0; ix < stockStore.getLength(); ix++)
```

```

    {
        stocksFlexTable.removeRow(1);
    }

    stockStore.clear();
}
}
} // if sender is the clear all button
});

```

Handling Storage Events

You can register storage event handlers with a storage object, and these are invoked **for the current window or tab** when data is written to or deleted from the storage. Although the HTML5 spec states that storage events fire in all windows and tabs of the browser, this should not be assumed because few browsers implement this. Notice that if the storage is cleared, the event does not contain any information about the deleted key-value pairs.

The storage event handlers get a storage event object that contains useful information, such as the old value and the new value, in the case of an update to an existing key-value pair. The following can be obtained from the StorageEvent object:

| Method | Description |
|----------------|-------------------------------------------------------------------------------------------------------------------------------|
| getKey | Returns the key being changed. |
| getNewValue | Returns the value of the key after the change, or null if not changed or if it is the result of a Storage.clear() operation. |
| getOldValue | Returns the value of the key before the change, or null if not changed or if it is the result of a Storage.clear() operation. |
| getStorageArea | Returns the SessionStorage or LocalStorage object where the event occurred. |
| getURL | The address of the document in which the change occurred. |

The following snippet shows a sample event handler registered with storage, where the changes from the incoming events are displayed in a UI label.

```

import com.google.gwt.storage.client.Storage;
import com.google.gwt.storage.client.StorageEvent;
private Storage stockstore = null;

```

```
stockStore = Storage.getLocalStorageIfSupported();
if (stockStore != null)
{
    stockStore.addStorageEventHandler(new StorageEvent.Handler()
    {
        public void onStorageChange(StorageEvent event)
        {
            lastStockLabel.setText("Last Update: "+event.getNewValue() +": "
            +event.getOldValue() +": " +event.getUrl());
        }
    });
};
```

GWT 2.5 RC - This documentation is for a release candidate of GWT 2.5. For the most recent stable version, see GWT 2.4.

~~~ End of Article ~~~



Session 19: HTML5 Geolocation and APIs

Find me! Geolocation-enabled Opera build

| | |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source | http://dev.opera.com/articles/view/labs-find-me-geolocation-enabled-opera-build/ |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Another great Opera technology release is with us — we are delighted to release the first build of Opera with Geolocation API support — you can use this to expose the browser's geographical position, and make use of it in your applications.

This article gives you some background information, plus usage details and links to all the files you need to start playing with this exciting new functionality now.

Please note that the geolocation feature is now included in stable Opera release.

History of geolocation

Geolocation on the web is not new — many sites already use the IP address of the browser to serve targeted content, mostly in the form of advertisements. You have doubtless seen banners along the lines of *"Find a Friend in [your city]"* while surfing the web. However, the IP address method is notoriously inaccurate and cannot be reliably used for more advanced geolocation services.

You can get a much more accurate location reading if the device the browser is running on has a GPS available, or can triangulate using wireless access points or cell towers. And even if the device doesn't have the right hardware for this, a location provider web service can be used to provide a fairly accurate location reading.

The modern web has many applications that can make great use of this data - geographical information can really enrich your data set. It would be possible to use this API with services like Google Maps (for finding services around your current location), Dopplr (for registering your current location as part of a trip) and Flickr_ (for finding pictures around your current location).

This kind of functionality has been made a lot easier with the introduction of the Geolocation API.

The Geolocation API

The Geolocation Working Group of the W3C has recently released the first working draft of the geolocation API specification. The API is used via JavaScript, with commands being used to retrieve the current latitude and longitude of the browser. The following snippet shows how a web page would request the browser's location:

```
// One-shot position request:

navigator.geolocation.getCurrentPosition(showMapCallback);
function showMapCallback(position) {
    // Show a map centered at (position.coords.latitude, position.coords.longitude).
}
```

As you can see, the API is very simple; it doesn't get really much more complicated than this, even with more advanced examples (check out examples from the specification_).

More importantly, leaving it to the browser to transmit its location means that the user can remain in control of their privacy. The W3C Geolocation API is likely to become a widely adopted standard, and Opera is providing this early implementation of the API to let developers and users start experimenting with it.

How the Opera Geolocation build works

- Once you have downloaded and installed the build, check out the [Geolocation API spec](#), and start playing with the functionality.
- This build uses the Skyhook web service to retrieve the location information - this allows you to use the API even if you don't have access to a GPS or triangulation device. To develop applications that make use this service you need to register your site on loki.com .
- Additionally, if you're running Windows XP you will also need to run svcsetup.exe, which ensures that wifi scanning will not be affected by the various "wifi managers" shipped with many laptops. All this won't be necessary in future releases.

This Opera build will prompt the user to make sure they agree to send their location, every time a site requests it. Your actual location (and under the right circumstances this build can readily identify your location to an accuracy of a few meters) is extremely sensitive information. Our current limited UI is designed to allow you to test the build as easily as possible, while protecting your location information — we take the security and privacy of users very seriously. For normal usage, you should not give your location information out anywhere except to trusted applications from trusted developers.

~~~ End of Article ~~~



## Geolocation of your users based on IP address (how to)

|               |                                                                                                                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Source</b> | <a href="http://developer karma.com/blog/andrew-m-riley/geolocation-your-users-based-ip-address-how">http://developer karma.com/blog/andrew-m-riley/geolocation-your-users-based-ip-address-how</a> |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Have you ever needed to find out where a site visitor physically was for something beyond analytics? I'm talking about things like finding your user on a map or locations of Store X within 50 miles of the user's current location. Well, I have, and I've found there's not really one solution that fits all users. Sometimes I find myself having to mix methods to provide the best overall user experience and/or changing up my execution depending on my needs.

For the most part you can break it down to three questions:

1. Am I using this on mobile? (Go HTML5)
2. Do I want to spend money? (If No: Go Google.load() with backup)
3. Do I need to process this data at a later time (batch processing) or don't want to rely on outside services? (Go Local Database)

One note about mobile: Most of the services below will look up the IP to the address of who's registered it (the ISP) instead of where it is being served from. The best example is using a Local Database with my phone; it claims that I'm in Alabama when I'm really in Florida. It turns out the IP address is registered to some place in Alabama.

### Local Database

Local databases do have a lot to offer if you are willing to invest both time and money. I tend to use this type of service when I need to do back end processing on the IP addresses. It's perfectly fine to use this for real time IP address geolocation lookups, but there are other options that are cheaper and easier if that is all you are using it for. If you are using this method for mobile, I would recommend that the finest resolution you can trust is Country.

This method requires you purchasing a database dump or CSV file and then importing it to your local database server. The company you are purchasing from will usually provide an API of how to pull the data from your database and or do special conversions. Every few months or so the company will email you to notify you there is an update. At this point you'll need to go through and repeat the process to update your database. If this paragraph scares you, then please don't go with this method.

Some providers will also provide you with binary files to avoid using a local database. I usually skip this method since having hundreds of thousands users hitting my server in a day all read from a series of files on my file system scares me.

#### Example:

maxmind (link) / ip2location (link)

#### Pros:

Unlimited queries if you host in your own database, Returns Latitude, longitude and other information\*. Can use this one off-line (aka, not real time while the visitor is on your site). Don't rely on third party services.

#### Cons:

Costs money, Needs to be updated on a regular basis, Can have wrong location for mobile

#### When to use:

When you have a lot of lookups to do and don't really care about mobile, or you are going to be doing your lookups as part of a batch process behind the scenes.



## HTML5

HTML5 is now really a conglomerate of web based technologies that are supposed to make our lives easier. The geolocation API is one of those technologies.

GPS is the reason this method truly shines in mobile. Most phones will allow the geolocation API pull directly from GPS and then fall back to tower triangulation. Either way, you get a fairly accurate location. Do keep in mind, though, that this technology only provides latitude and longitude. You would have to use a separate service to find out what State/County/Country the user is located in.

The only big downside I see is on most phones/browsers it pops down a dialog that asks the user if they want to divulge their location. This sometimes runs into issues since users can just hit "no". It's also not a great user experience since now they have to hit yet another button just to use your site.

**Example:**

<http://html5demos.com/geo>

**Pros:**

Free, Works well on mobile, unlimited queries since it's client side

**Cons:**

Needs a modern browser, asks the user to determine your location, Only returns latitude & longitude

**When to use:**

Use this when you are using mobile without a doubt. There are libraries that will make it easy to use on most phones. You can use it in desktop browsers but remember that not all browsers support it.

## Google.load()

Google.load() was not created to do geolocation lookups. In fact, it is created to load handy JavaScript libraries from a central CDN. It just happens to pass back an object that contains latitude, longitude, city, state and country.

I enjoy using this method on desktop based websites since I already have to use a library like jQuery. I just tell it to load jQuery from google.load() and I happen to get the location data tacked on as part of an object. This provides a good user experience since they don't have to click on anything and a good experience for my server since I don't have to do yet another request to my server (or anybody's server for that matter).

This method does have one huge drawback; you can't trust it to work reliably. On cell phones if the object is empty (aka, it didn't work), I use the HTML5 method to I cover my bases and provide a good all round user experience. On the desktop if the object is empty, I fall back to using a Local Database call.

Note: I have tested this method on multiple phones with mixed results. You can't trust that this will work on ALL phones with ALL versions.

**Example:**

<http://files.developerkarma.com/geo.html>

**Pros:**

Free, Unlimited queries, Returns latitude, longitude and other information

**Cons:**

It works on some mobile phone browsers and not others. International results may or may not work.

**When to use:**

When you're doing desktop development and you want a fast result without having to do a second call (most of the time). It's a bonus that you can load your favorite javascript library as part of this.

## ***In Closing***

Please remember that Geographical Information Systems are just large databases of rather poor data created by humans. There will be errors, omissions and oddities no matter which method you use. Even if you somehow manage to use all the methods listed above I couldn't guarantee 100% accuracy or even 100% results. Write your code to fail gracefully and give your users options to work around these issues when you can.

*~~~ End of Article ~~~*



# HTML5 Drag and Drop

**Source** <http://dev.opera.com/articles/view/drag-and-drop/>

## Introduction

HTML5 includes the **Drag and Drop API**, which gives us the ability to natively drag, drop, and transfer data to HTML elements. Up until now, JavaScript libraries have commonly been used to achieve something similar, such as jQuery UI's Draggable and Droppable plugins or Dojo.dnd. What libraries such as jQuery UI or Dojo can't do, however, is interact with other windows, frames, and applications (to and from the file system) or have a rich drag data store.

Marketers like to brand all things HTML5 as shiny and new — disappointingly then, the HTML5 Drag and Drop model has roots in Internet Explorer 5 (and below). Ian Hickson, the HTML5 specification editor, reverse engineered the Drag and Drop API from Internet Explorer. The best thing it had going for it was the fact that it was already working in IE, and to some extent Safari and Firefox had implemented it. It may not be the most elegantly designed part of HTML5, but the paths had already been paved.

Before you get too excited, a word of caution: the current cross-browser landscape isn't pretty. Chrome, Firefox, Safari, and Internet Explorer all implement some form of Drag and Drop, but violate the spec in many different ways. Certain parts of the API, like the dropzone attribute, are only implemented in Opera. Cross-browser drag and drop is going to be tricky (though not impossible) until all the browsers work through their implementation bugs.

Drag and drop is supported now in Opera 12 and to some extent in current versions of Chrome, Safari, Firefox and will be in IE10. With time, we hope all browsers will move towards compatible implementations, or that the spec will be changed to match implementation reality. As always, please remember to file bugs with the browser vendors!

## The Drag and Drop API

Without further ado, let's dive into the main Drag and Drop API features, and see what you can do with them.

### The draggable attribute

The quickest way to get started with drag and drop is to add `draggable=true` to an element.

The `draggable` attribute indicates to the browser that an element is eligible for dragging with three possible values: `true`, `false`, and `auto`. For example, in the following list of items, the first is draggable, the second isn't, and the third may or may not be depending on the user agent's default behavior.

- You can drag me.
- You can't drag me.
- You might be able to drag me.

Keep in mind that text selections are always draggable by default, as are images and links. You can set their `draggable` attributes to `false` to prevent this, which may not be very nice for your users.

Here's the markup for the previous example:

```
<ul>

  <li draggable=true>You can drag me.</li>

  <li draggable=false><a href="http://opera.com/">You can't drag me</a>.</li>

  <li draggable=auto>You might be able to drag me!</li>

</ul>
```

## The dropzone attribute

The dropzone attribute allows you to indicate in markup *where* draggable elements may be dropped — and to some extent what to do with them *when* they are dropped.

The dropzone attribute takes a list of space-separated items with the following possible values: copy, move, link, “string:”, and “file:”

The copy, move, and link values, which correspond mostly to `dataTransfer.dropEffect`, indicate the kind of operation that is allowed as the result of a drop.

The string: and file: values allow the author to restrict the type of data that can be dropped by adding a data type— it has to be at least one letter long, and can't contain spaces:

```
<div id=droppy dropzone="copy string:text/html file:image/png"></div>
```

This example will put data from text/html content that gets dragged in, or from PNG images dragged from the file system or otherwise obtained via the File API, into the drag data store.

Even with a dropzone element declared, in order to accept a drop the drop target needs to listen for the drop event.

```
var div = document.getElementById('droppy');

div.addEventListener('drop', doSomethingAmazing);
```

## The drag data store

What makes HTML5 Drag and Drop more interesting than a run-of-the-mill JavaScript drag and drop library is the concept of the *drag data store*. At the core of every drag and drop operation lies a bucket of information that can be passed around. The data store consists of a data store item list, listing each piece of dragged data. These drag items all have a kind (either “Text” or “File”), a type (a string, usually given as a MIME type), the actual data (a unicode or binary string), and some other information that can be used by the browser to give UI feedback.

The `dataTransfer` object of a drag event is how we get access to read from, write to, and clear the data in this data store. Each `dataTransfer` object is also linked to a `dataTransferItemList` object (accessible via the `itemsgetter`), which contains one or more `dataTransferItem` objects (accessible via the index of the `dataTransferItemList` object).

## Drag and drop events

Drag and drop introduces seven new events that HTML elements can listen for, all of which bubble. With the exception of `dragleave` and `dragend`, they are all cancellable.

Drag and drop events are basically mouse events with a data store. Note that you can both read and write to the data store during a `dragstart` event, but it is read-only during drop.

For items that are dragged, the events are:

- `dragstart`: dispatched when the user begins dragging a draggable element (or selection).
- `drag`: dispatched when the draggable element is moved during a drag (e.g., with the mouse).
- `dragend`: dispatched when the user ends the drag and drop operation.

For any element that may receive a drop, the events are:

- `dragenter`: dispatched when a draggable object is first dragged over an element.
- `dragleave`: dispatched when a draggable object is dragged outside of an element.
- `dragover`: dispatched when a draggable object is moved inside an element.
- `drop`: dispatched when a draggable object is dropped.

These are fairly intuitive, but let's examine the event sequence exhibited during a Drag and Drop sequence, to help us understand exactly how it works.

## The Drag and Drop event sequence

When the user starts dragging something, the `dragstart` event fires on the source element, which can be an element with `draggable=true` set on it, or any elements that are draggable by default, such as links, images, text selections, etc. The `dragstart` event's handler can now add data to the data store, to be dropped later on. Once the `dragstart` event has fired, a series of events fire every few hundred milliseconds, or every time the mouse moves. They are:

1. `drag`: firing at the source element
2. `dragover`: firing at the current target element

This keeps happening until the mouse passes over a new element. At this point, we see a change: `dragenter` then fires immediately after the last drag event. If `dragenter` is cancelled, then this new element becomes the current target, `dragleave` fires on the previous target to indicate that it is no longer the target, and the `dragover` event will fire on the new target. If `dragenter` is not cancelled, then the `<body>` becomes the current target. If the `dragover` event is cancelled as well, then the current target element will be allowed to receive the drop event so the user can now drop the payload being dragged. If not, then dropping will be prevented, as per the spec.

To be compatible with other browsers, we've changed our implementation to now allow the element to become the current target even if `dragenter` is not cancelled. IE still (correctly, per spec) requires this.

So to summarize, the usual repeated event sequence is:

1. `drag`
2. `dragenter` (if needed)

3. `dragleave` (if the current target has been changed)
4. `dragover`

When the user drops what they were dragging, if dropping was allowed in the last repeated sequence, then `drop` fires at the current target. The `drop` event is allowed to see all of the data that was dropped — the only other event that can do so is the `dragstart` event that created it. If it is not cancelled, then the browser will be allowed to do whatever it would normally do in that case (for example when dropping a file, if you don't cancel the `drop` event the browser should try to open the file). After the drag has ended, it fires a `dragend` event at the source element (even if the drop was not allowed).

This series of events targeted at both the source and target elements allows drags to start in one document and end in another, with both documents being able to monitor the state of the drag. If the drag starts outside the window, or passes from inside it to outside it, then only the events that should be targeted at the relevant document will fire, and the others are assumed to have fired and been allowed or cancelled as needed. If multiple documents are involved, then each receives the relevant events from the sequence.

## The `effectAllowed` and `dropEffect` properties

`dataTransfer.effectAllowed`: this determines which of the three "effects" the user will be allowed to choose from when dropping. Possible values are:

none, all, copy, link, move, copyLink, copyMove, and linkMove.

The "compound" values like `copyMove` mean copy *or* move.

`dataTransfer.dropEffect`: determines which "effect" will be used when the user drops (or in the case of the `drop` and `dragend` events, which one was already used when the user dropped). This will be either the user's choice (if they pressed a modifier key), or the default (depending on what type of element/selection is being dragged), or none (if the user selected a type of drop effect that was not allowed, or dropped somewhere where dropping was not allowed, in which case the `drop` event will not fire). Possible values are none, copy, link or move.

The script events can also write to this property, to override the user's choice. If the final `dropEffect` (the last one seen in a `dragover` event) is not one that is compatible with `effectAllowed`, the drop will not be allowed (`dropevent` will not fire, `dragend` event sees "none" as the `dropEffect`).

So what's the point of all this? The idea is to allow drag and drop scripts to offer functionality like an OS file manager. You know, drag a file from one folder into another. Does it:

- create a "copy" of the file?
- "move" the file?
- create a "link"/shortcut to the file?

The user can choose by pressing modifier keys (they are different on different OSes, but Ctrl, Shift and Alt all do something on Windows). With drag and drop, the idea is that the script looks at the property, and offers relevant functionality. For example, if I offer a remote file manager UI using drag and drop, I can allow the user to press keys to select what should happen, and my script could see what they chose and act accordingly.

## Some simple examples

Now we've explored all the theory, let's put it into practice and look at some working examples.

To see the below examples in action, have a look at our Drag and Drop demo.

## A simple drag source

The simplest example using the normal events just uses a simple source element (you'll need to use a link if you want it to work in IE as well) with the `draggable="true"` attribute.

```
<div draggable="true">Ain't life a drag.</div>
```

The draggable `<div>` should listen for the `dragstart` event, and put some data into the data store when it is fired. It can also set `dataTransfer.effectAllowed`, to state whether the user will be allowed to select any of “copy” “move” or “link” using modifier keys. These will be used by the operating system if that's where the drag ends up, or be passed to the drop handler, if the user chooses a permitted effect that is compatible with the drop target's `dataTransfer.dropEffect` property.

```
<script>

var div = document.getElementsByTagName('div')[0];

div.ondragstart = function (event) {

    //browsers may populate the text/html data type for you,
    //but Firefox will not allow a drag without setting some data
    event.dataTransfer.setData('text/plain', 'Hello world');
    event.dataTransfer.effectAllowed = 'copyLink'; //copy or link
}
</script>
```

Any potential drop target should then listen for and cancel the `dragenter` and `dragover` events, and then also listen for the all-important drop event, and in most cases cancel that too. Working with the dropped data is done in the drop event handler.

```
<div>Drop here.</div>

<script>

var droptarget = document.getElementsByTagName('div')[1];

droptarget.ondragenter = droptarget.ondragover = function (event) {

    event.preventDefault();

}

}
```

```
droptarget.ondrop = function (event) {  
  
    event.preventDefault();  
  
    this.innerHTML = event.dataTransfer.dropEffect + ' ' +  
  
        event.dataTransfer.getData('text/plain');  
  
};  
  
</script>
```

## Text selections

Dragging text selections is easier to handle. The dragstart event fires at the parent element, but it does not need a handler to create data. The selected text is added as text/plain data. Therefore if we just want to deal with text selections, we can simply remove the dragstart event handler from the above example, and just assume there is some text that the user can select and start dragging.

## Microdata

Any micro data items attached to a dragged element, or intersected by a dragged selection, will be included as a JSON string with the mime type application/microdata+json. When parsed, this is represented as a series of objects and arrays, such as:

```
parsedJSON.items[0].properties.foo[0] == 'item property data'
```

Currently only Opera supports this.

## Files

When files are dropped from the system into a web document (or otherwise obtained via the File API), they appear in the dropped data's dataTransfer.files collection. Each entry in the collection is a File object, as specified in the File API spec. The contents can then be read using a FileReader object.

```
<script>  
  
droptarget.ondrop = function (event) {  
  
    event.preventDefault();  
  
    if( event.dataTransfer.files.length ) {
```



```
this.textContent = 'File: '+event.dataTransfer.files[0].name;

var reader = new FileReader();

reader.onload = function () {

    var fileAsString = reader.result;

    ...

}

reader.readAsBinaryString(event.dataTransfer.files[0]);

}

}

</script>
```

## Security

Nothing in the HTML5 specification currently mentions the security implications of drag and drop operations. At Opera, we've taken a few measures to make things safer for our users.

## Uploads

In Opera's implementation, if a file from the file system is dragged into a page the drop won't happen unless the user explicitly allows it. Figure 1, taken from [fontdragr.com](http://fontdragr.com) shows an example of this:



Figure 1: Our dialog box, which prompts the user to allow a page to interact with a dropped file from the filesystem.

## Exposing event origin

In addition to UI considerations, we've proposed (and implemented) the following extensions to `event.dataTransfer`:

```
interface DataTransfer {
    ...

    readonly attribute DOMString origin;

    void allowTargetOrigin(DOMString targetOrigin);
};
```

`dataTransfer.origin` is a read-only property that returns the protocol, domain and optional port of the origin where the drag started. If the drag was not started on an origin (such as a dragged file from

the desktop), or on a URL that is not a scheme/host/port tuple, the value should be the string value “null”.

`dataTransfer.allowTargetOrigin(targetOrigin)` defines an origin match for sites which may receive the dropped data. If this method is not called, then all sites and applications may be considered dropzones.

Just like `postMessage`, `dataTransfer.allowTargetOrigin` takes one of the three arguments: “\*” (matches all origins), “/” (matches current origin), or an absolute URL.

The origin must match permitted target origins on the data store, otherwise the dragenter, dragover, dragleave and drop events are never fired.

`dataTransfer.allowTargetOrigin` is intended to be used by a site (we can call it the source site) that is putting something sensitive inside the data store, and wants to ensure it will not be leaked to untrusted sites. `dataTransfer.origin` is intended to be used by a site that will activate a sensitive operation when data is dropped, and wants to ensure that only trusted sites started the drag. That is, it is intended for use by the target site.

To demonstrate this, we've created a [sample security page](#). The top-level window includes the following bit of code:

```
box.addEventListener('dragstart', function(event){  
  
    ...  
  
    event.dataTransfer.allowTargetOrigin('http://people.opera.com');  
  
});
```

This ensures that the drop can only happen on a document with `http://people.opera.com` as its origin.

The embedded `iframe` includes this:

```
drop.addEventListener('drop', function(event){  
  
    event.preventDefault();  
  
    if (event.dataTransfer.origin == "http://miketaylr.com") {  
  
        var secrets = event.dataTransfer.getData('text');  
  
        ...  
  
    }  
  
});
```

Before we operate on the data, we check the origin of the drop and make sure we trust it.

Both `dataTransfer.allowTargetOrigin` and `dataTransfer.origin` may be used, or a site may just use one. It all depends on the use case, and what needs to be protected — the data, the action, or both.

## Summary

This concludes our tour of HTML5 Drag and Drop. Pat yourself on the back if you've made it this far, you deserve it. Please let us know what you think in the comments, and help us improve things by submitting bugs.

*~~~ End of Article ~~~*

