HTML links - lets build a web

From Web Education Community Group

Contents

- 1 Introduction
- 2 What are links?
- 3 The anatomy of an anchor link
- 4 The href attribute
- 5 Creating in-page navigation with id attributes
- 6 Don’t leave any ambiguity about what you’re linking to
  - 6.1 Providing extra information with a title attribute
  - 6.2 Linking to non-HTML resources — don’t make people guess
  - 6.3 External vs. internal links
- 7 Frames and popups — just say no
- 8 Benefits of outbound and inbound links
- 9 Link wording
- 10 Link styling
- 11 HTML5: block level linking
- 12 Summary
- 13 Exercise questions

Introduction

In this article you’ll learn all about one of the most ground-breaking inventions in the history of the Web — links. Links allow the reader of a document to follow them to another document and jump from server to server without having to disconnect and connect all over again. A lot has happened since they were first invented but one thing stayed the same: links are a very important part of the web experience and you can make accessing your content and functionality easy or hard for your web site’s visitors, depending on how you use them.

After you’ve gone through this article you’ll know how to create links that are easy to understand and function in any environment. Furthermore you’ll learn how linking affects your search engine popularity and you’ll get some tips about wording links. As usual, there is an accompanying zip file to this tutorial (http://dev.opera.com/articles/view/18-html-links-let-s-build-a-web/links_code.zip) , which contains several files we’ll refer to as we go along.

What are links?

Links are parts of a web site that point to other resources — other HTML documents, text files, PDFs, etc. There are links that are followed automatically by the browser, created using `<link>` elements (you’ve already encountered some of those in earlier articles—they were used to import CSS files into an HTML document) and then there are links that are optional for the user to activate. These are called **anchors** and you can add them to the document using the `<a>` element.

The anatomy of an anchor link

You can turn any inline element in the document into an anchor link by adding an <a> element around it. For example, in the following HTML document the text *Opera Software* gets turned into a link (linkexample.html (http://dev.opera.com/articles/view/18-html-links-let-s-build-a-web/linkexample.html)).

```
<!DOCTYPE html>

<html lang="en-GB">
<head>
  <meta charset="utf-8">
  <title>Link Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>A link to Opera</h1>
  <p><a href="http://www.opera.com">Opera Software</a></p>
</body>
</html>
```

Visitors activating this link (either by clicking it with a mouse, or activating it with the keyboard or voice in some cases) will leave the current site and go to the Opera home page. There are more changes happening to the link itself, and we'll see about them later when we talk about link styling.

The anchor has several attributes you can add:

- `href` — the resource it points to (either an external file or an anchor ID).
- `id` — an ID to uniquely identify the link, for example if you want to style just that link differently to all the others. You can also use an `id` attribute to make an link into a page anchor, and link to it from other <a> elements.
- `class` — a class to identify this link and others (and different elements, if you really want to), for example if you want to style certain links on the page with styles, but not others.
- `title` — extra information about the external resource.

Let's go through the most important attributes first and then talk about what you can do to make things easy to grasp for your visitors.

The href attribute

An <a> element can play several roles depending on which attributes are set. The most common attribute you'll use is the `href` attribute, which defines what resource the link points to. This attribute can contain different values:

- A URL in the same folder (help.html), relative to the current folder (for example "../../help/help.html" — 2 dots means "go up one level in the site folder hierarchy") or absolute to the server root (for example "/help/help.html" — having a forward slash at the front of the address means the address starts at the root of the computer the page is on.)
- A URL on a different server altogether (for example http://wait-till-i.com or ftp://ftp.opera.com/ or http://developer.yahoo.com/yui).
- A fragment identifier or id name preceded by a hash (for example "#menu"). This points to a target inside the same document.
- A mixture of URLs and fragment identifiers — you can link directly to a section of a different document by pointing the `href` attribute to a URL followed by a fragment identifier (for example http://dev.opera.com/articles/view/new-structural-elements-in-html5/#aside).

Using an absolute or relative URL will make an `<a>` element into a link as it will point to somewhere else.

Creating in-page navigation with id attributes

You can also put an `id` attribute on an `<a>` element, to make it into a page anchor. You then reference that ID in the `href attribute of another link to link to it. For example:`

```
<h2><a id="sec1">Section #1</a></h2>
```

Could be linked to by

```
<a href="#sec1">Section One</a>
```

But most browsers you'll want to support these days allow you to write a shortcut for this, and put the ID directly on the element you want to link to, for example:

```
<h2 id="sec1">Section #1</h2>
```

This is much simpler, so we'd recommend that you stick to this.

The following HTML has examples of all the different types of links in it (linkexamples.html (http://dev.opera.com/articles/view/18-html-links-let-s-build-a-web/linkexamples.html)):

```
<!DOCTYPE html>

<html lang="en-GB">
<head>
  <meta charset="utf-8">
  <title>Different Links Example</title>
  <link rel="stylesheet" href="linkexamplestyles.css">
</head>
<body>
  <h1>Different Links</h1>

  <h2>Example of in-page navigation with fragment identifiers, links and anchors</h2>
  <div id="nav">
    <ul id="toc">
      <li><a href="#sec1">Section One</a></li>
      <li><a href="#sec2">Section Two</a></li>
      <li><a href="#sec3">Section Three</a></li>
      <li><a href="#sec4">Section Four</a></li>
      <li><a href="#sec5">Section Five</a></li>
    </ul>
  </div>

  <div id="content">
    <div>
      <h2 id="sec1">Section #1</h2>
      <p><a href="#toc">Back to menu</a></p>
    </div>
    <div>
      <h2 id="sec2">Section #2</h2>
      <p><a href="#toc">Back to menu</a></p>
    </div>
    <div>
      <h2 id="sec3">Section #3</h2>
      <p><a href="#toc">Back to menu</a></p>
    </div>
    <div>
      <h2 id="sec4">Section #4</h2>
      <p><a href="#toc">Back to menu</a></p>
    </div>
    <div>
```

```
      <h2 id="sec5">Section #5</h2>
      <p><a href="#toc">Back to menu</a></p>
    </div>
  </div>

  <h2>Some other link examples</h2>

  <ul>
    <li><a href="http://dev.opera.com">Opera Developer Network</a></p></li>
    <li><a href="http://www.wait-till-i.com/stuff/JavaScript-DOM-Cheatsheet.pdf">Dom Cheatsheet</a></p></li>
    <li><a href="ftp://get.opera.com/pub/opera/win/">Download different Opera versions</a></li>
    <li><a href="http://farm1.static.flickr.com/56/188791635_0b8bdd808d.jpg?v=0">Photo of my book</a></li>
  </ul>

  </body>
</html>
```

Open this file in your browser of choice and experiment with it. You'll find that activating any of the links in the first list will jump to the appropriate section of the document. You might also have realized that the URL in the location bar of your browser changed and now shows the fragment identifier at the end of it, which means visitors can bookmark this section or email the link to other people to send them exactly where they should go. To recap:

- anchor links can have a fragment identifier as the value of the `href` attribute — this fragment identifier must start with a hash sign (#).
- When activated, this link will jump to any HTML element with an `id` of this value. The IDs on each page must be unique.
- IDs follow certain naming conventions. Most importantly is that they must start with an alphanumeric character and must not have any spaces in them.

That covers the menu and the different sections in the example but what about the other links? If you try them out you'll see that they point to different targets — one goes to another site, another displays a photo and the third one shows a specific section of another web page (found by jumping to a specific ID). If all of that worked for you, great — but what if you or your browser couldn't understand some of these resources?

Don't leave any ambiguity about what you're linking to

The most important thing to remember about links is that they are a substantial part of your relationship with your visitors. They trust that when you offer them a link, they can follow it and get good, relevant information. If your links don't work because the linked resource is not available or in a format the visitor cannot consume you will betray that trust and lose credibility. Don't let that happen.

Providing extra information with a title attribute

Like almost any other HTML element you can add a `title` attribute to an `<a>` element to add some extra information. Browsers will show a so-called tooltip when visitors hover their mouse cursor over the link. This tooltip then tells them what the link is about. For example you could give a small introduction to the content and location of the linked document (titleexample.html (http://dev.opera.com/articles/view/18-html-links-let-s-build-a-web/titleexample.html)):

```
<!DOCTYPE html>

<html lang="en-GB">
<head>
  <meta charset="utf-8">
<title>Adding extra information with a title attribute</title>
<link rel="stylesheet" href="linkexamplestyles.css">
</head>
<body>
  <h1>Adding extra information with a title attribute</h1>
```

```
  <ul>
    <li>Find more information on the <a title="The Yahoo Developer Network is the main hub for all the develo
     tools Yahoo offers, including the Yahoo User Interface Library (YUI) and the Design Patterns repository"
     href="http://developer.yahoo.com">Yahoo Developer Network</a>.</li>
  </ul>

  </body>
</html>
```

However, you cannot expect visitors to have enough patience and hand-eye coordination to rely on this for crucial information. Visually impaired users, who cannot see the page at all, are very likely not to be able to reach this information. While screen readers have the option to read out `title` attributes for the end user it is turned off by default, which is why you should never use the `title` attribute for crucial information about the link.

Crucial information might be:

- Linking to non-HTML resources like PDF files, images, videos, sound files or downloads.
- Leaving the current site and linking to another server (external vs internal links).
- Linking to a document that'll open in a different frame or a popup.

Linking to non-HTML resources — don't make people guess

It can be very annoying when you click on a link and your browser does not know what to do with the content the link points to. It is unfortunately all too common to see web sites link to images, PDF documents and videos without warning their visitors to be prepared. Videos especially are very often a cause for browser crashes. Furthermore, the resource might be on the larger side (20MB PDF anyone?), which means that visitors might prefer to download it rather than opening it inside the browser, or just not access it at all.

One of the biggest success factors of a web product is not keeping people guessing what happens when they perform an action and instead tell them flat out what effects their action will have. In the case of links all you need to do to prevent a lot of frustration is to tell your visitors what the linked resource is. Here are some examples (linkingnonhtml.html (http://dev.opera.com/articles/view/18-html-links-let-s-build-a-web/linkingnonhtml.html)):

```
<!DOCTYPE html>

<html lang="en-GB">
<head>
  <meta charset="utf-8">
<title>Linking non-HTML resources</title>
<link rel="stylesheet" href="linkexamplestyles.css">
</head>
<body>
  <h1>Linking non-HTML resources</h1>

  <ul>
    <li>Find more information on the <a href="http://developer.yahoo.com">Yahoo
      Developer Network site (external)</a></li>
    <li>Download the <a href="http://www.wait-till-i.com/stuff/JavaScript-DOM-Cheatsheet.pdf">
      Dom Cheatsheet (PDF, 85KB)</a></li>
    <li>Pick and <a href="ftp://get.opera.com/pub/opera/win/">download different Opera
      versions from their FTP (external)</a></li>
    <li>Check out a <a href="http://farm1.static.flickr.com/56/188791635_0b8bdd808d.jpg?v=0">
      Photo of my book (JPG, 200KB)</a></li>
  </ul>

  </body>
</html>
```

By providing such information about linked files and their nature you leave the decision of what to do with them to your visitors rather than expecting them to have certain browser settings or installed software. If you mix that with clever styling you can even make such links look pretty and intuitive (for example, by giving

different types of link different easily recognisable icons - find more out about this in Styling lists and links). If you want to be very safe, also offer a help section that explains what the different file formats are and where you could get the software needed to display them.

External vs. internal links

One of the biggest fears of business decision makers when it comes to their company's web sites is people leaving prematurely. This is often the reason for never offering third party links (unless the third parties pay money for the privilege of having web traffic directed towards them). We'll come back to this error in judgment later on; for now let's talk about what people do to avoid visitors leaving their site and how these measures affect the site's success.

Frames and popups — just say no

The fear of losing visitors to other sites while still wanting to link to them gave us some inventions in web development that have been a thorn in the side of usability for years: frames and popups.

Using HTML frames means you separate the page shown in the browser into several different documents. The benefit is that the document seemingly stays the same even when you load parts of it either from your own server or from third party servers. This is where the usefulness ends however — frames are a terrible user experience and actually harmful:

- Search engines can never index a whole page but instead might show up parts of a page in search results that don't make sense out of context.
- Visitors cannot bookmark the page — the next time they open their bookmark they'll get the initial state of the frameset and not the page as they left it.
- Visitors dependent on assistive technology have a very hard time navigating around framesets.
- Third party sites might not like to be shown inside a frameset and use "framebreaker" scripts that replace framesets with the real URL when you try to embed them. This is to stop criminals luring Internet users into entering for example credit card information into a web site that seemingly looks like a bank (so called "phishing").

Links inside a frameset use the `target` attribute of the anchor to target the correct frame. Each frame in a frameset gets a certain name and activating the link would open the document defined in the `href` attribute in that frame. If the frameset is not available (for example when a visitor found the document with the links via a search engine) each link opens in a new browser instance.

Opening a new browser instance is another common way to link to third party sites — either with a scripted pop-up window or with a `target` attribute with a value of `_blank`. The fact that every modern browser comes with a pop-up blocker should give an indication of how safe it is to rely on this technique in this day and age. It's not!

In short: **do not use the `target` attribute when you create links, unless you really know what you are doing**. It is an outdated idea anyhow — these days most browsers have tabbed interfaces, so users can open third party sites in the background for later reading while they stay on your site. Under certain circumstances you may want to indicate the difference between external and internal links, but always leave it to the discretion of the visitor what they want to do with them.

Benefits of outbound and inbound links

There are several good reasons for linking to third party sites even when they are competitors.

- It gives you credibility in the eyes of your visitors — you are so sure of the quality of your content that you don't shy away from the competition.
- It is an opportunity to deliver a full service — you can link to content and articles or even products on other sites that you don't cover but that might be of interest for those visitors who want to dive deeper into the topic at hand.
- You can prove a point by building on an older article by a third party and offering a better or different solution and referencing the old article as a proof via a link.

The usefulness of inbound links (links pointing from a third party site to yours) is less debated. The more often that valid and high quality sites link to yours with relevant keywords, the higher you'll rank in search engines such as Google. Before that happens however you need to prove that you don't shy away from linking to others either.

The relevant keywords bring us to another very important part of creating good links: how to word them.

Link wording

I've covered this partly in the section about linking to non-HTML resources, but it is good to remind ourselves that links are not only part of the page copy but also interactive elements in the document.

Some assistive technologies will offer a list of links instead of the whole document to allow visitors to quickly navigate their way through it and find the link they want, which means that your link text needs to make sense out of context as well as in context. You can easily check this in Opera by opening any web site and choosing Tools > Links from the menu or pressing Ctrl + Shift + L. You'll get a new tab that shows all the links in the document and where they point to.

You should also make sure that there are not links that have the same wording but point to different resources. The classic mistake here is "click here" links, worded for example like "To download the latest version of our tool **click here**". It is much better to use a link text that explains what it points to — in the case of "You can download the latest version of our tool and try it out for yourself", turn the words "download the latest version of our tool" into a link.

The same applies to "more" links. You'll find these in news sites where you get a heading and some teaser text and a "more" or "full story" link to follow. The solution to this problem is to either use a linked "more" image and give it a unique alternative text or to add a span inside the link after the "more" and hide it with CSS.

Link styling

We won't give this a full treatment until the [CSS section of the course (http://www.w3.org /wiki/Web_Standards_Curriculum#CSS)], but it is useful to consider at this point that the way links look is very important and there are several different link states to consider. The link states (which relate to CSS pseudo-selectors — this sounds complex, but it isn't) are:

- `<link>` — the default state — it defines what links should look like in a certain part of the document. By default, unvisited links are coloured blue.
- `<visited>` — the style of a link that has already been visited before (and might already be in the browser cache). By default, already visited links are coloured purple.
- `<hover>` — the style of a link whilst the mouse cursor is hovering over it.
- `<focus>` — the style of a link whilst it has been given focus (or highlighted) via another alternative control mechanism, most commonly the keyboard.
- `<active>` — the style of the link while it is activated, ie while the connection to the other site is in

progress; it is also the style of the last activated link when you arrive at the document by going back in your browser.

HTML5: block level linking

In HTML4, the <a> element was restricted to just turning other inline elements into links. This was ok for most situations, but it became annoying when you wanted to, for example, turn a whole entire advertising banner containing images and paragraphs into a link. For the code to remain valid, you'd have to wrap the different bits of text and images in their own separate links, which is horribly repetitious, and confuses assistive technology users ("why are there multiple links going to the same place?") Putting an inline element round a load of block level content also makes styling behave weirdly, unless you set it to display like a block level element using CSS.

HTML5 removes this restriction, allowing you to put a link round any amount of content you want. To get it to behave properly, you'll currently still have to set the link to behave like a block level element, but this is immediately a lot more flexible than what went before. Let's look at an example:

```
<!DOCTYPE html>

<html lang="en-GB">
<head>
  <meta charset="utf-8">
  <title>Block level link Example</title>
  <link rel="stylesheet" href="styles.css">
  <style>
    a {
      display: block;
      background-color: blue;
      text-decoration: none;
      color: white;
      width: 300px;
      height: 100px;
    }

    a:hover {
      background-color: red;
    }

  </style>
</head>
<body>
  <a href="http://www.opera.com">
    <h1>A link to Opera</h1>
    <p>Opera Software</p>
  </a>
</body>
</html>
```

Here you can see that I've got the <a> element wrapping both a heading and a paragraph. To make this work correctly I've set it to `display: block;` in the CSS. If you [try the example out yourself (http://devfiles.myopera.com/articles/373/blocklevellink.html)], you'll see that the entire block is part of the clickable link area. I have added a colour changing hover effect to make this more visible.

Summary

We covered a lot this time, but it is very important to remember how links work and what they should do. You will learn a lot of tricks and techniques in your career as a web developer to override this default behaviour and I hope you'll stop and wonder if what you are trying to do is really necessary.

Exercise questions

- What is wrong with the following link: `<<a href="report.pdf" title="report as PDF, 2.3MB">get our latest report</a>>`?
- What is the `target` attribute in links for and are there any good uses for it?
- I've talked about link relationships and connections between links and anchors. Is there an attribute for links that describes relationships between documents, too?
- How can you write a link that sends the visitors to an element further down the page when they click it? What do you need to make sure of beforehand?

Note: This material was originally published as part of the Opera Web Standards Curriculum, available as 18: HTML links - let's build a web! (http://dev.opera.com/articles/view/18-html-links-let-s-build-a-web/) , written by Christian Heilmann. Like the original, it is published under the Creative Commons Attribution, Non Commercial - Share Alike 2.5 (http://creativecommons.org/licenses/by-nc-sa/2.5/) license.

Retrieved from "http://www.w3.org/community/webed/wiki/HTML_links_-_lets_build_a_web"
Categories: Tutorials | WSC | HTML

- This page was last modified on 28 January 2012, at 22:52.