First Edition

Tiny.txt

Assets That Will Improve Your Website

Written by **Alexander Dawson**

Tiny.txt: Assets That Will Improve Your Website, First Edition

© Copyright 2023, Alexander Dawson, All rights reserved.

Find me on the web at: https://alexanderdawson.com/

To report errors, send me a message at: alex@hitechy.com

Editor: Alexander Dawson

Writer: Alexander Dawson

Publisher: Alexander Dawson

First published and showcased through the website AssetConfig.com, of which I am the sole creator. The idea for this book is based on a series of articles written (by myself) and published many years ago on a now defunct web development blog. Code examples use the libraries provided on the AssetConfig platform and are offered to you under the same (MIT) open source license as the website.

Set in 45/20/18/14/10 Geomanist.

Chapter Images use Icon54 Classic.

Printed and published in the United Kingdom.

Tiny.txt

Assets That Will Improve Your Website

By
Alexander Dawson

Table of Contents

"For the long-suffering professionals who work tirelessly to make the web a better place, and those individuals who continue to strive for a more inclusive, sustainable, standards compliant Internet."

Preface

Hello there, I'm Alexander Dawson.

I've been building websites and applications for a long time, over 20 years (if you want to put a number on it). I'm based in the UK (Brighton) and I currently work as a freelancer designing websites, developing products (and open source tools), and authoring content for well-known publications. I also specialize in Web standards, Inclusive design, Web sustainability, and front-end performance.

Acknowledgements

Before we get started, I just wanted to give thanks to everyone who has chosen to grab a copy of this book. I realize there are plenty of web development titles out there, so I hope that you find this one useful on your journey and worthy of your time. If you have feedback or want to get in touch, just visit my website (it's listed on the inside cover and the back of the book) and drop me a friendly email. I do try to reply to everyone who contacts me (time permitting).

Introduction:

Quick files that will improve your website

Writing a book, just like building a website or app, is a commitment. You’re dedicating several months to ploughing your experience and brainpower into something that (hopefully) other people can follow and will in-turn find useful. This particular book has been written with one goal in mind, tackling those often neglected and misunderstood small web assets that can be bolted onto your website or app to provide added functionality, with very little development overhead.

Some of these tiny files improve the user-experience, some improve search engine optimization, and others enable your website to run as a single page application. It’s all pretty exciting stuff, yet not enough focus is given to these files, which is why I'm taking the time to cover them in detail, all in one place, where you can reference them at any time and build them in any order to boost your site's usefulness.

Your Expectations

Within this book, we shall tackle each file within its own chapter to ensure that you don’t get overwhelmed. You shouldn’t feel the need to include every file within every web project you launch, as some will ultimately be unsuitable (depending on your sites requirements). Though, it’s worth knowing about each of them on the off chance you may need one of them either in the future or within another job.

You can always return to that chapter and create the file when you require it. In fact, this book is designed, so you can jump to the chapter (document) you want to build and not need to have read anything before getting started (with a few exceptions). Though, as with many things Web development, it's easier on a workflow to decide to integrate and use a technology before you begin a project.

If you wish to know what you should expect from this book:

- A comprehensive guide to assets used in web development.
- Lots of hand coding examples based on living specifications.
- Easy-to-follow steps shown in a logical format for each file type.

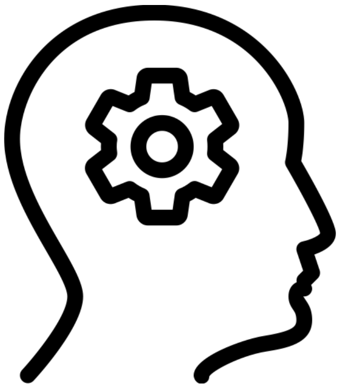
If you want to know what you shouldn't expect from this book:

- This isn't a guide to HTML, CSS, JS, or to building a full website.
- This book expects you to know basic front-end development.
- You can use any toolchain you like, but don't expect automation.

Chapter 1:

Pathways within the document head

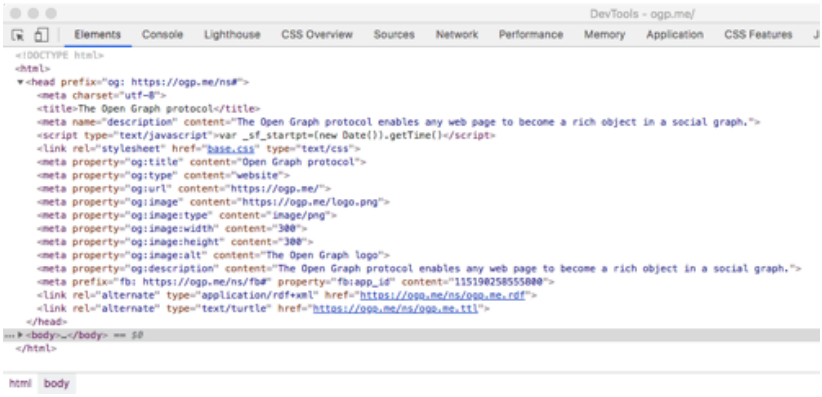
Build an inventory of what you need in the head of your HTML document using links, metadata and more.



<head>

Whilst this isn't a file format in itself (in fact, it exists within every HTML file you produce), I feel it's important enough to dedicate a chapter to this particular segment before we jump into the rest of the book.

The head element of every web page or app will contain a multitude of elements, many of which will contain references to the very files we'll be talking about throughout this title. Often these references are necessary for the browser to sniff them out and utilize them, so it's critical you get to know them and include them in your documents.



```
<!DOCTYPE html>
<html>
▼<head prefix="og: https://ogp.me/ns#">
    <meta charset="utf-8">
    <title>The Open Graph protocol</title>
    <meta name="description" content="The Open Graph protocol enables any web page to become a rich object in a social graph.">
    <script type="text/javascript">var _sf_startpt=(new Date()).getTime()</script>
    <link rel="stylesheet" href="base.css" type="text/css">
    <meta property="og:title" content="Open Graph protocol">
    <meta property="og:type" content="website">
    <meta property="og:url" content="https://ogp.me/">
    <meta property="og:image" content="https://ogp.me/logo.png">
    <meta property="og:image:type" content="image/png">
    <meta property="og:image:width" content="300">
    <meta property="og:image:height" content="300">
    <meta property="og:image:alt" content="The Open Graph logo">
    <meta property="og:description" content="The Open Graph protocol enables any web page to become a rich object in a social graph.">
    <meta prefix="fb: https://ogp.me/ns/fb#" property="fb:app_id" content="115190258555800">
    <link rel="alternate" type="application/rdf+xml" href="https://ogp.me/ns/ogp.me.rdf">
    <link rel="alternate" type="text/turtle" href="https://ogp.me/ns/ogp.me.ttl">
  </head>
... ▶<body>…</body> == $0
</html>
```

These references usually come in the form of either meta tags or link elements; and while neither holds more importance than the other, there's a long history of overuse and abuse of meta tags, with both search engines and browsers ignoring proprietary tags. It's a pretty diverse environment that's determined more on convention and on popularity (and adoption) than best practice (like other elements of the semantic web). This is because search engines, social networks, apps and we as developers determine their support through usage.

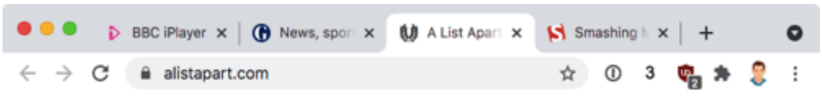
Reference For a guide to the elements which can be included in the head, check the HTML5 specification at https://www.w3.org/TR/html52/document-metadata.html

Generic Elements

Before we start looking at meta and link tags, let's first examine the other elements you'll find within the head that should be mentioned.

Element	Required	Description
<base>	No	This element dictates the base directory of anchor references (for relative linking to locations). There can be only one per file!
<noscript>	No	This element provides you an opportunity to offer an accessible fallback for if JavaScript fails.
<script>	No	This element allows you to either link to or embed JavaScript within the page or app.
<style>	No	This element embeds a block of CSS style accessible only to that page or app.
<title>	Yes	This element describes your app or webpage within the browser's title bar.

As you can see, the only one of the above you're required to include in all of your documents is a title tag. There's no set rule as to what you should include within it but ensure you make it relevant to your website's brand and be consistent throughout all of your pages.



Regarding the script and style tags, they can occasionally be useful to inject code that needs to be run immediately before render-blocking occurs. While you can use async with JavaScript, CSS currently has no such feature (without some JS assistance) so directly embedding can help with above the fold preloading. If you do use JavaScript, be sure to use the noscript element to offer a fallback because even today, cases of no-JS availability can occur (and break your site).

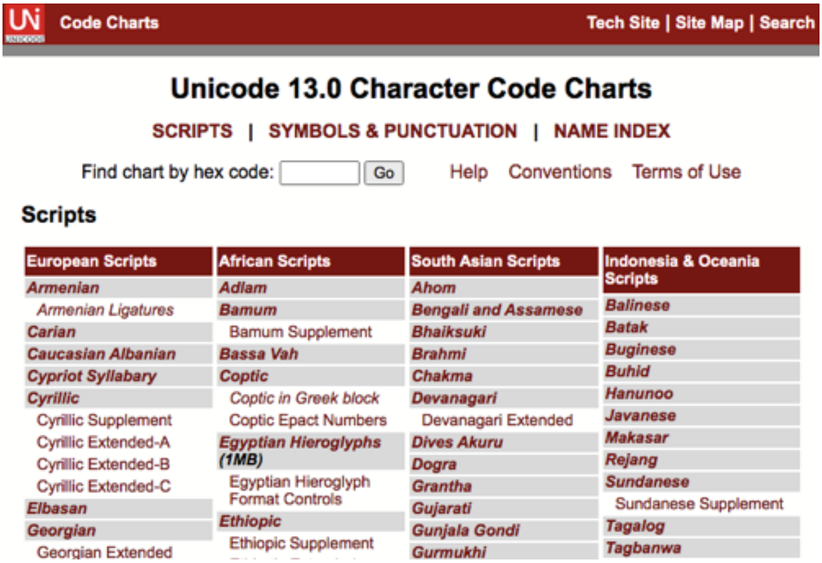
<meta>

Of all the tags you can include within the head of your documents, the most notorious are the meta tags. Over the years they have been a bit of a wild west of proprietary formats, non-adoption, and abuse with literally thousands appearing (or being misinterpreted) and vying for widespread attention. The truth is nobody really knows how much search engines or social networks factor them in, and with a couple of exceptions, their value has been very much diminished.

Reference	Google provides a handy reference guide to which meta tags their search engine actively recognizes https://developers.google.com/search/docs/advanced/crawling/special-tags

Before we begin, let's address the most important meta tag, and the only one which is required to be included within every document; the charset meta element. Unless you require some unique encoding format, best to stick with unicode (utf-8) and just include the below as-is within your files, ensuring it's the first thing below the head.

Element	Description
<meta charset="utf-8">	Defines the encoding for your app or website (utf-8 is the default).



Code Charts

Tech Site | Site Map | Search

Unicode 13.0 Character Code Charts

SCRIPTS | SYMBOLS & PUNCTUATION | NAME INDEX

Find chart by hex code: Go Help Conventions Terms of Use

Scripts

European Scripts	African Scripts	South Asian Scripts	Indonesia & Oceania Scripts
Armenian	Adlam	Ahom	Balinese
Armenian Ligatures	Bamum	Bengali and Assamese	Batak
Carian	Bamum Supplement	Bhaiksuki	Buginese
Caucasian Albanian	Bassa Vah	Brahmi	Buhid
Cypriot Syllabary	Coptic	Chakma	Hanunoo
Cyrillic	Coptic in Greek block	Devanagari	Javanese
Cyrillic Supplement	Coptic Epact Numbers	Devanagari Extended	Makasar
Cyrillic Extended-A	Egyptian Hieroglyphs (1MB)	Dives Akuru	Rejang
Cyrillic Extended-B	Egyptian Hieroglyph Format Controls	Dogra	Sundanese
Cyrillic Extended-C	Ethiopic	Grantha	Sundanese Supplement
Elbasan	Ethiopic Supplement	Gujarati	Tagalog
Georgian		Gunjala Gondi	Tagbanwa
Georgian Extended		Gurmukhi	

http-equiv Elements

There are two types of optional meta tags we should examine. Firstly, we shall examine http-equiv tags which can impact the browser.

Element	Description
<meta http-equiv="Content-Security-Policy">	Control where resources (after the tag) are loaded from.
<meta http-equiv="content-type">	Serves the same purpose as the charset tag (defining encoding).
<meta http-equiv="default-style">	Set the default stylesheet to be used by this website or app.
<meta http-equiv="imagetoolbar">	This deprecated tag only affects IE6 users as they hover images.

Element	Description
<meta http-equiv="msthemecompatible">	This deprecated tag only affects Windows XP (IE6+) users GUI's.
<meta http-equiv="refresh">	Avoid this tag which redirects to a new page. See chapter 2.
<meta http-equiv="window-target">	Require the document or app to appear in a specific frame.
<meta http-equiv="x-dns-prefetch-control" content="off">	Totally opt out of browser based DNS prefetching.
<meta http-equiv="x-ua-compatible" content="IE=edge">	Force Internet Explorer 8+ to use its latest rendering engine.

Name Elements

Next we shall examine the name meta tags which can impact your sites SEO if they are utilized by search engines or social networks:

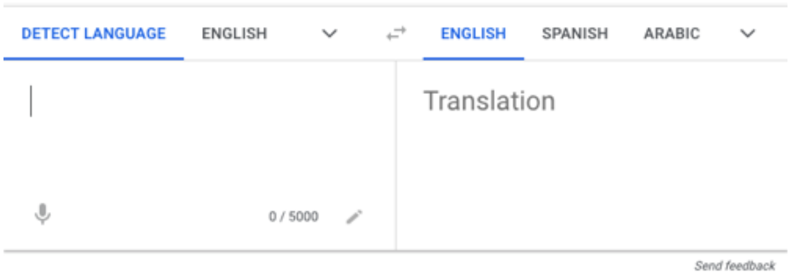
Reference This wiki provided by the WHAT-WG provides a comprehensive guide to all the known name meta tags https://wiki.whatwg.org/wiki/MetaExtensions

Element	Description
<meta name="abstract">	A summary of the content - useful for long documents.
<meta name="apple-mobile-web-app-capable" content="yes">	Triggers a web app running on iOS to run in full screen.

Element	Description
<meta name="apple-mobile-web-app-title">	Set a friendly title for apps when added to the iOS home-screen.
<meta name="apple-mobile-web-app-status-bar-style" content="default">	If in fullscreen, possible values you can use are light-content, dark-content, hidden or default.
<meta name="application-name">	Short name of your web app for when added to home screens.
<meta name="author">	The individual or organization who published the website.
<meta name="color-scheme">	Indicate if your site supports light, dark mode or just one palette.
<meta name="contact">	A phone number, email, or physical location for the website.
<meta name="copyright" >	Provide copyright details or a link to a specific license agreement.
<meta name="creator">	Can be used in place of author, it identifies who produced the site.
<meta name="description">	A description of your page, which is limited to up to 156 chars.
<meta name="designer">	Give credit where due via a link to the designer (See chapter 15).
<meta name="format-detection" content="telephone=no">	Disable auto-formatting of phone numbers on handheld devices.
<meta name="generator">	Promote the software used to create the website or application.

Element	Description
<meta name="geo.placename">	The city or town where the app or site is based (see chapter 14).
<meta name="geo.position">	Provide a longitude and latitude (semicolon separated) position.
<meta name="geo.region">	Two digit region code to indicate your current location (such as US).
<meta name="google" content="nositelinkssearchbox">	Tells Google not to show the SiteLinks search box for your site.
<meta name="google" content="notranslate">	Tells Google not to translate your document via Google Translate.
<meta name="google" content="nopagereadaloud">	Prevent voice assistants from reading the body of this page.
<meta name="googlebot">	Tell Google if this file should be indexed (see chapter 24).
<meta name="ICBM">	Use comma separated latitude and longitude to show a location.
<meta name="keywords">	Comma separated words relating to the site (won't affect SEO).
<meta name="mobile-web-app-capable" content="yes">	Adds a web app on Android to the home screen.
<meta name="msapplication-config">	Offers a link to Internet Explorer Tile images (See Chapter 27).
<meta name="monetization">	Allow your site to be monetized using payment stream providers.
<meta name="pinterest" content="nopin">	Stop Pinterest users from saving bits from your site to their boards,

Element	Description
<meta name="publisher">	Credit the company that allowed your content to be shown.
<meta name="rating">	Keep your site child-friendly with content labelling (see Chapter 21).
<meta name="referrer">	Control how referrer information is passed with HTTP requests.
<meta name="robots">	Tell search engines if a file should be indexed (see chapter 24).
<meta name="skype_toolbar" content="skype_toolbar_parser _compatible">	Disable Skype from automatically formatting and detecting phone numbers on the desktop.
<meta name="subject">	Gives a brief overview of what subject your site or app relates to.
<meta name="theme-color">	The color you'd prefer a browser UI to be styled to match an app.
<meta name="viewport" content="width=device-width, initial-scale=1">	Used by mobile devices to indicate how they scale and size. This tag must be declared first.



OpenGraph Elements

These days, being connected to social media is an essential part of any business. Ensuring a website is compatible with the major social giants is a must-have. The below meta tags ensure that any sharing of your pages will look and behave better on networks they're shared.

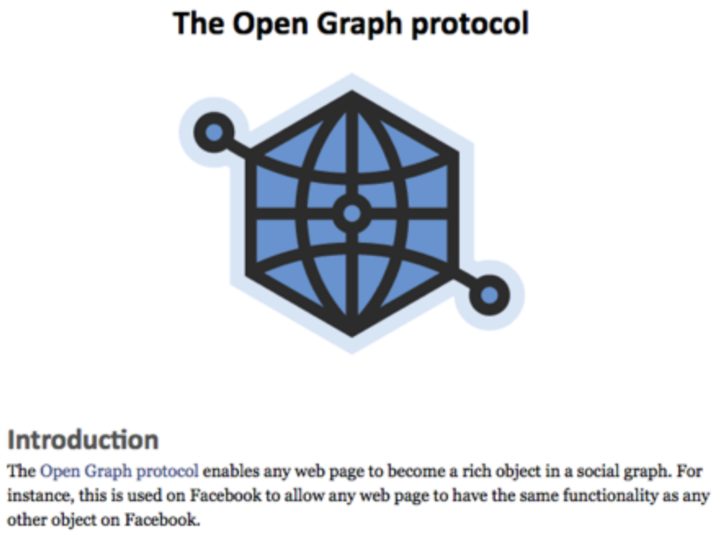
Element	Description
<meta property="og:description">	Describes your content briefly, as it'll appear as a snippet.
<meta property="og:image">	An image to show your post on social media (min 1080px wide).
<meta property="og:image:alt">	Alternative text for if your image fails to load for accessibility.
<meta property="og:locale">	Defaults to en_US but if you need a different locale, use one.
<meta property="og:title">	The title of your content without any branding (site name).
<meta property="og:type">	The default is website, but you can specify music or video.
<meta property="og:url">	This should be the base URL of your main website or application.
<meta property="og:video">	If you want streaming video to play, use this element as well.
<meta name="twitter:card" content="summary">	The content for this is always summary, as it's a base tag.
<meta name="twitter:site">	The @username of the business or app which the page relates.

Element	Description
<meta name="twitter:creator">	The @username of the person who owns or runs this site.
<meta name="twitter:description">	In 200 characters or less, detail the content of the document.
<meta name="twitter:title">	In 70 characters or less, provide a title for the article on this page.
<meta name="twitter:image">	An image for this article less than 5mb (JPG, PNG, WEBP or GIF).
<meta name="twitter:image:alt">	Make your image accessible with alternative text.
<meta name="twitter:dnt" content="on">	If you embed tweets, use this to keep your site's data private.

Reference	Read the specs at https://developers.facebook.com/docs/sharing/webmasters (Facebook), https://developer.twitter.com/en/docs/twitter-for-websites/cards/overview/markup (Twitter) and https://ogp.me/

Dublin Core Elements

Below are some more name meta tags which utilize the DCMI (Dublin Core Metadata Initiative) standard, which is pretty widely adopted:



The Open Graph protocol

Introduction

The Open Graph protocol enables any web page to become a rich object in a social graph. For instance, this is used on Facebook to allow any web page to have the same functionality as any other object on Facebook.

Element	Description
<meta name="dc.contributor">	If you have multiple contributors to a resource, you can include them here.
<meta name="dc.coverage">	Include place names or coordinates to show the scope of the document.
<meta name="dc.creator">	The person or business responsible for the app or site will be added here.
<meta name="dc.date">	Using YYYY-MM-DD you can provide a timestamp for content creation.

Element	Description
<meta name="dc.description">	Offer a brief explanation or a table of contents of what the content is about.
<meta name="dc.format">	If the content isn't in HTML (default), use this to offer a format MIME type.
<meta name="dc.identifier">	Provide a link to a URL, DOI, ISSN or ISB which references the content.
<meta name="dc.language">	Provide details of the locale in which th content is offered to visitors.
<meta name="dc.publisher">	Provide details of the organization or individuals responsible for content.
<meta name="dc.relation">	If there are any links which connect to the content, provide the reference.
<meta name="dc.rights">	License information, copyright details o links to agreements are used here.
<meta name="dc.source">	If the content cites any sources, you car credit the references with this tag.
<meta name="dc.subject">	Offer details of the content's subject using semicolon separated strings.
<meta name="dc.title">	Give the article title without branding sc it can be read by aggregators.
<meta name="dc.type">	Shows content categories, functions, genres, or aggregation levels.

In chapter 8, I'll describe how to utilize the Dublin Core to provide an easy-to-index file for a project that offers a rich amount of metadata about your website or app that search engines can use if they aren't already taking advantage of newer schemas like microdata. Aside from that, using DC terms within individual pages has a similar effect.

Summary

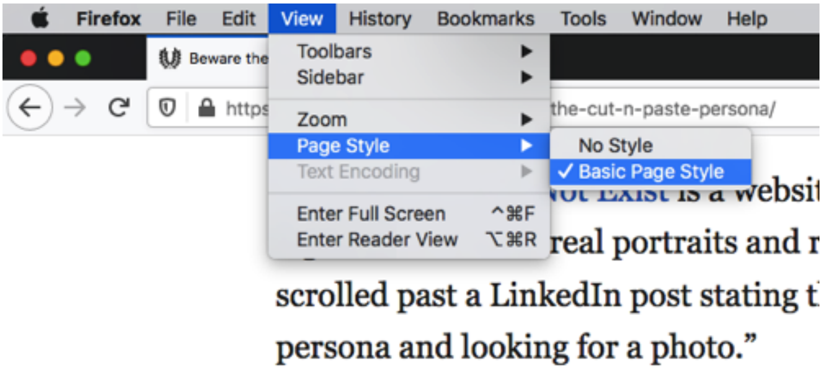
Granted, all of this is a lot to take in. You're not likely to want to include them all into your pages because otherwise they'll become bloated and probably quite repetitive (as some tags just say the same thing). So which do I recommend? I've included a template below that suits most static web pages and apps, feel free to adapt it as needed.

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-
scale=1">
<meta name="description" content="Description.">
<meta name="referrer" content="no-referrer">
<meta name="apple-mobile-web-app-title" content="Title">
<meta name="theme-color" content="#ffffff">
<meta name="msapplication-config"
content="browserconfig.xml">
<meta property="og:title" content="Title">
<meta property="og:description" content="Description.">
<meta property="og:type" content="website">
<meta property="og:locale" content="en_us">
<meta property="og:url" content="https://example.com/">
<meta property="og:image" content="images/image.png">
<meta property="og:image:alt" content="Logo">
```

Reference For more information about DCMI, check the spec at https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ or read this handy usage guide at https://www.dublincore.org/specifications/dublin-core/usageguide/2000-07-16/simple-html/

<link>

While meta tags are an important part of the head of any document, the element which holds the most interest for us regarding this book as it connects documents to their web assets is the link element. You'll be using this particular joining force to activate various web files within your sites and apps so that devices, browsers, social networks and search engines can use them. This is where things get innovative, as it's the backbone for how many of the books assets function.



real portraits and r

scrolled past a LinkedIn post stating t

persona and looking for a photo."

Let's get the most common one out-of-the-way first, you'll be using this one every time you build a website or app. It's the stylesheet link, which gives your pages style to go along with their substance. Often you'll have a primary stylesheet which contains your primary visual details, but you can also offer alternative ones that can be chosen by the end-user, such as for themes (IE and Firefox users have this feature built-in but Chrome users require an extension to take advantage).

Element	Description
<link rel="stylesheet" href="styles.css">	Apply an external, primary CSS stylesheet to the document.
<link rel="alternate stylesheet" href="theme.css" title="Theme">	Offer a secondary stylesheet which the user can trigger upon request. Useful for offering website themes.

With the evolution of browsers you can provide stylesheets with numerous conditions, such as the ability to wrap them in conditional comments (to target Internet Explorer), target them using media queries for specific resolutions, or target device or user preferences (such as dark mode). You can even provide a special stylesheet for printers or voice assisted users. We will cover this later in the book.

Reference	If you want to target a specific version of IE using conditional comments, read this Wikipedia article: https://en.wikipedia.org/wiki/Conditional_comment

URL Links

Next, let's look at all the other link types that can be used to provide useful information via URL's for browsers or search engines.

Element	Description
<link rel="amphtml">	Link to a Google AMP version of the page.
<link rel="archives">	A collection of records or historical docs.
<link rel="canonical">	Avoid duplication issues with your content.
<link rel="contents">	Opera navigation link to table of contents.
<link rel="dns-prefetch">	Resolve the DNS quickly for performance.
<link rel="edit">	Indicate a location to edit the document.
<link rel="EditURI">	Used by 3rd party tools to maintain a blog.
<link rel="first">	The first document in a series of articles.
<link rel="glossary">	Opera navigation link to a list of definitions.
<link rel="home">	Opera navigation link to the home page.
<link rel="index">	The top-level file in the website's structure.
<link rel="last">	The last document in a series of articles.
<link rel="me">	Information about the document author.
<link rel="micropub">	Post to your domain via a micropub client.
<link rel="modulepre-load">	Triggers high-priority and early loading of module-based scripts (JavaScript).
<link rel="next">	The next document in a series of articles.

Element	Description
<link rel="openid.delegate">	The site to identify an OpenID account.
<link rel="openid.server">	The server to validate a user's details.
<link rel="pingback">	Lets blogs identify when pages linked to.
<link rel="preconnect">	Connect to a URL in readiness for requests.
<link rel="prefetch">	Cache a file before it’s requested by a user.
<link rel="preload">	Force the browser to download an asset.
<link rel="prerender">	Load an entire page in the background.
<link rel="prev">	The previous file in a series of articles.
<link rel="self">	When files have multiple references.
<link rel="shortlink">	Provide a shortened URL for this resource.
<link rel="subresource">	Prefetch a file but with the highest priority.
<link rel="up">	Link to a page higher in the URL structure.
<link rel="webmention">	Notifies a URL when you’ve linked to it.

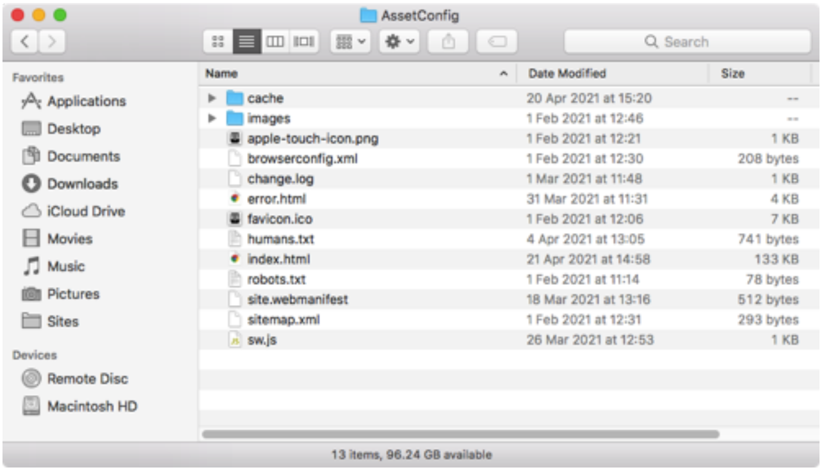
OpenID
The Internet Identity Layer
Home » Welcome to OpenID Connect
Welcome to OpenID Connect

Asset Links

Finally, we get to the best stuff, the assets which you can link to that will give your site a tiny but useful benefit. It's the core of the book:

Element	Description
<link rel="alternate">	Offer RSS & Atom feeds [See chapter 25].
<link rel="apple-touch-icon">	A 180px Image for iOS users [See chapter 11].
<link rel="apple-touch-startup-	A splash screen for iOS apps [See chapter 11].
<link rel="author">	Give authors credit [See chapters 8 & 13-15].
<link rel="help">	Offer support or guidance [See chapter 21].
<link rel="icon">	An SVG favicon for browsers [see chapter 11].
<link rel="license">	Link to the website's license [see chapter 17].
<link rel="manifest">	Give your PWA an app icon [See chapter 27].

Element	Description
<link rel="mask-	Only use this deprecated SVG in old Safari.
<link rel="P3Pv1">	Set privacy rules for old IE (See chapter 20).
<link rel="search">	Search within the browser (See chapter 19).
<link rel="shortcut icon">	A deprecated favicon standard which has been replaced. (see chapter 11).



Name	Date Modified	Size
cache	20 Apr 2021 at 15:20	--
images	1 Feb 2021 at 12:46	--
apple-touch-icon.png	1 Feb 2021 at 12:21	1 KB
browserconfig.xml	1 Feb 2021 at 12:30	208 bytes
change.log	1 Mar 2021 at 11:48	1 KB
error.html	31 Mar 2021 at 11:31	4 KB
favicon.ico	1 Feb 2021 at 12:06	7 KB
humans.txt	4 Apr 2021 at 13:05	741 bytes
index.html	21 Apr 2021 at 14:58	133 KB
robots.txt	1 Feb 2021 at 11:14	78 bytes
site.webmanifest	18 Mar 2021 at 13:16	512 bytes
sitemap.xml	1 Feb 2021 at 12:31	293 bytes
sw.js	26 Mar 2021 at 12:53	1 KB

13 items, 96.24 GB available

Summary

This concludes the list of stuff you can include within the head of your document. With great power comes great responsibility, and as with the META tags it's probably a lot to take in. As to what you should include, I recommend only including the bare minimum you need to reduce HTTP requests and keep your project performant. To give you a sample of what might work for a website, I've provided some link tags below that I commonly use within most of my work.

```
<link rel="preload" as="font" type="font/woff2" crossorigin
href="cache/font.woff2">
<link rel="stylesheet" href="cache/style.css">
<link rel="author" type="text/plain" href="humans.txt">
<link rel="apple-touch-icon" sizes="180x180" href="apple-touch-
icon.png">
<link rel="icon" href="images/icon.svg" type="image/svg+xml">
<link rel="manifest" href="site.webmanifest">
```

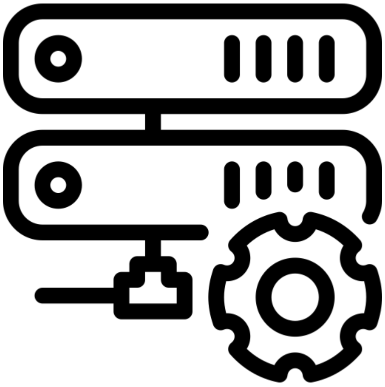
As with the meta example, you could place them together within the head of an HTML document and use that as a boilerplate to get yourself started (replacing the template text with your own). You can also use any of the meta or link tags provided in this chapter to help optimize your site and make it as productive as possible for visitors.

Reference Web Developer Josh Buchea has created a GitHub page that gives a quick reference to several of these properties: https://github.com/joshbuchea/HEAD

Chapter 2:

Optimizing the server for performance

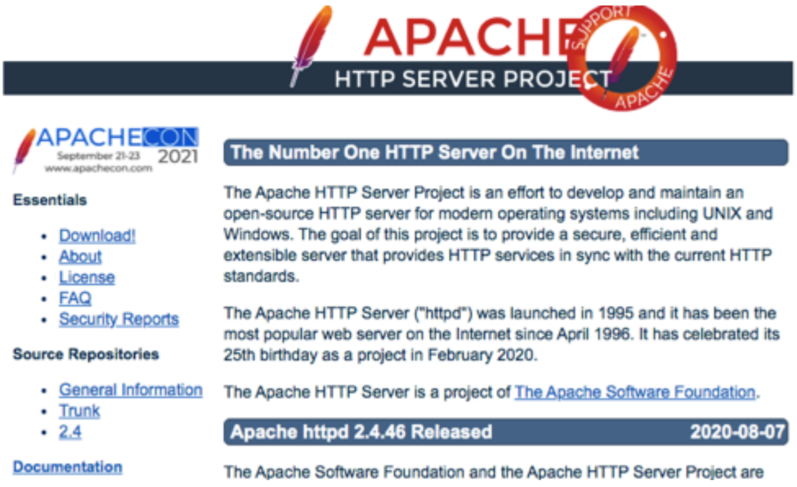
Build Apache server configuration files with all the trimmings you require for a fast, efficient server.



.htaccess

Welcome to the wonderful world of server configuration. I know this is a daunting world to find yourself in, but if you want your projects to be as performant as possible, it's a critical part of the equation. If you happen to use one of the static hosting services to deploy your site (JAMStack) such as Netlify, this chapter will be entirely optional, so feel free to skip ahead to another chapter; otherwise, it's time to examine the powerful httpd.conf and htaccess server configuration files.

As I primarily deal with LAMP servers (Linux, Apache, MySQL & PHP), I'll focus all of my attention on the configuration of Apache within this chapter, however if you prefer another server such as NginX, IIS, or macOS Server there are better, dedicated guides out there. In fact, the HTML boilerplate team has a great NginX toolkit that'll help you achieve fairly similar results. With all of this in mind, let's get started.



The Number One HTTP Server On The Internet

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

The Apache HTTP Server ("httpd") was launched in 1995 and it has been the most popular web server on the Internet since April 1996. It has celebrated its 25th birthday as a project in February 2020.

The Apache HTTP Server is a project of The Apache Software Foundation.

Apache httpd 2.4.46 Released 2020-08-07

The Apache Software Foundation and the Apache HTTP Server Project are

For the basis of this book, I'm basing all of these performance tweaks upon the wonderful cultivated snippets offered in the open source HTML5 Boilerplate. As experience has shown, these to be among the most extensive and tested in the industry, offering a solid foundation of tweaks. I've made some expansions upon the base set provided to improve upon the compatibility where possible, and I'll examine each section in turn to help you identify if you should use it on your site.

Reference The HTML5 Boilerplate Apache files can be found here https://github.com/h5bp/server-configs-apache and the NginX boilerplate files can be found here https://github.com/h5bp/server-configs-nginx

As noted on the boilerplate's website, If you have access to a server's main configuration file (usually httpd.conf), it's always preferential to make tweaks there (as unlike .htaccess, it won't slow Apache down as much). If you don't, you can always create a .htaccess file in your base directory and include all the server tweaks you require within the file. The critical thing with httpd.conf or .htaccess is that you keep it as lean as possible, as the more actions a server has to perform against each HTTP request, the slower it will run (coding is a balancing act).

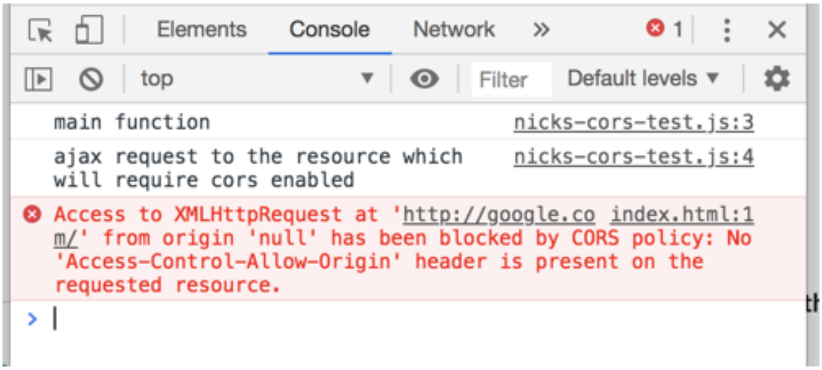
Reference If you need an introduction to working with .htaccess and httpd.conf files. Check the specification at https://httpd.apache.org/docs/current/howto/htaccess.html

Recommended

First, let's examine the stuff that you need within your htaccess:

Basic CORS Requests

To begin, let's examine Cross-origin resource sharing (otherwise known as CORS). Without getting into too much detail, it's something that prevents resources from another domain (or subdomain) from being loaded into your domain (for security reasons like hijacking).



```
main function                                  nicks-cors-test.js:3
ajax request to the resource which             nicks-cors-test.js:4
will require cors enabled
Access to XMLHttpRequest at 'http://google.co index.html:1
m/' from origin 'null' has been blocked by CORS policy: No
'Access-Control-Allow-Origin' header is present on the
requested resource.
```

While this is great for scripts, it can cause all sorts of headaches for things like images and typefaces. As such, I recommend including the following into your server config to send CORS headers for images and icons when browsers request it (and it also allows fonts to load).

```
<IfModule mod_setenvif.c>
<IfModule mod_headers.c>
    <FilesMatch "\.(avifs?|bmp|cur|gif|ico|jpe?g|a?png|svgz?|webp)
    $">
        SetEnvIf Origin ":" IS_CORS
        Header set Access-Control-Allow-Origin "*" env=IS_CORS
    </FilesMatch>
</IfModule>
</IfModule>
<IfModule mod_headers.c>
    <FilesMatch "\.(eot|otf|tt[cf]|woff2?)$">
        Header set Access-Control-Allow-Origin "*"
    </FilesMatch>
</IfModule>
```

Character Encoding

Ensuring that a site renders correctly is critical to the user experience. The below code is fairly straightforward as it firstly ensures that each file format is associated with the right MIME type, and then ensures that all text (by default) renders in unicode to ensure that it's formatted correctly when printed to the screen. There's quite a few potential file formats that can be used on the web, so the following code is quite big, you can remove any formats you've no intention of including in your project as that will make your htaccess file more performant.

```
AddDefaultCharset utf-8
<IfModule mod_mime.c>
    AddType application/atom+xml   atom
    AddType application/json   json map topojson
    AddType application/ld+json   jsonld
    AddType application/rss+xml   rss
    AddType application/geo+json   geojson
    AddType application/rdf+xml   rdf
    AddType application/xml   xml
    AddType application/manifest+json   webmanifest
    AddType application/x-web-app-manifest+json   webapp
    AddType application/wasm    wasm
    AddType application/octet-stream   safariextz
    AddType application/x-bb-appworld   bbaw
    AddType application/x-chrome-extension   crx
    AddType application/x-opera-extension   oex
    AddType application/x-xpinstall   xpi
    AddType text/javascript   js mjs
    AddType text/cache-manifest   appcache
    AddType text/calendar   ics
    AddType text/markdown   markdown md
    AddType text/vcard   vcard vcf
    AddType text/vnd.rim.location.xloc   xloc
    AddType text/vtt   vtt
    AddType text/x-component   htc
    AddType image/x-icon   cur ico
    AddType image/avif   avif
    AddType image/avif-sequence   avifs
    AddType image/bmp   bmp
    AddType image/svg+xml   svg svgz
    AddType image/webp   webp
    AddType font/woff   woff
    AddType font/woff2   woff2
    AddType application/vnd.ms-fontobject   eot
    AddType font/ttf   ttf
    AddType font/collection   ttc
    AddType font/otf   otf
    AddType audio/mp4   aac f4a f4b m4a
    AddType audio/mpeg   mp3
    AddType audio/ogg   oga ogg opus
    AddType audio/midi   mid midi kar
    AddType video/mp4   f4v f4p m4v mp4
    AddType video/ogg   ogv
    AddType video/webm   webm
    AddType video/x-flv   flv
</IfModule>
<IfModule mod_mime.c>
    AddCharset
    utf-8 .appcache .atom .bbaw .css .geojson .htc .ics .kml .js .json .
    jsonld .log .manifest .map .markdown .md .mjs .php .rdf .rss .top
    ojson .txt .vcard .vcf .vtt .webapp .webmanifest .xloc .xml
</IfModule>
```

URL Rewrites

Part of maintaining a website is redirecting pages from one location to another as URL’s change and your structure evolves. If your host doesn’t allow FollowSymlinks, then just remove it and SymLinksIfOwnerMatch should be enabled instead (though it will unfortunately impact your site's performance). Some cloud hosts may require RewriteBase and some servers may need RewriteOptions - though only enable these if and when they are required by your host.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    Options +FollowSymlinks
    # Options +SymLinksIfOwnerMatch
    # RewriteBase /
    # RewriteOptions <options>
    RewriteRule ^old/(.*)$ /new/$1 [R=301,NC,L]
</IfModule>
```

As demonstrated by the above, you replace both old and new with the names of the folders you wish to redirect. For redirecting to an individual file, the syntax is slightly different, but the effect is the same.

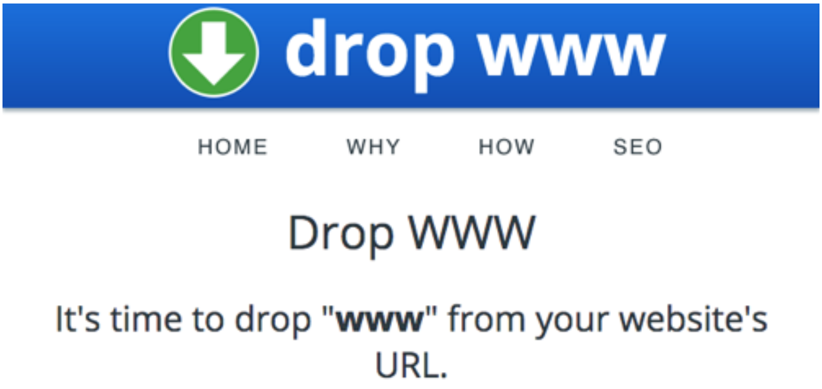
```
Redirect 301 /old/file.html https://example.com/new/file.html
```

Next, let’s rewrite our URL’s for security reasons, ensuring that when a visitor browses to a page on your website, instead of getting an HTTP version, they are redirected automatically to the HTTPS page. You should provide an HTTPS certificate to allow secure browsing, if you don't have one already, Lets Encrypt offers free ones, so there’s no excuse not to have a privacy first approach to your projects. Below is the code required to trigger the redirect from HTTP to HTTPS.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{HTTPS} !=on
    RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [R=301,L]
</IfModule>
```

Here’s another useful code snippet; removing the www from the website address. Why would you want to-do it? Well, the www doesn’t really offer visitors anything except extra letters to type when trying to find you, plus eliminating it reduces the chance of duplication (being indexed at both site.com and www.yoursite.com) which may hurt your SEO. Sounds good? Add the below into your htaccess file and it’ll redirect visitors.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{HTTPS} =on
    RewriteRule ^ - [E=PROTO:https]
    RewriteCond %{HTTPS} !=on
    RewriteRule ^ - [E=PROTO:http]
    RewriteCond %{HTTP_HOST} ^www\.(.+)$ [NC]
    RewriteRule ^ %{ENV:PROTO}://%1%{REQUEST_URI} [R=301,L]
</IfModule>
```



Drop WWW

It's time to drop "**www**" from your website's URL.

If you want to always have a www at the start of your URL, use the code below instead (you should only use one or the other, not both):

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{HTTPS} =on
    RewriteRule ^ - [E=PROTO:https]
    RewriteCond %{HTTPS} !=on
    RewriteRule ^ - [E=PROTO:http]
    RewriteCond %{HTTP_HOST} !^www\. [NC]
    RewriteCond %{SERVER_ADDR} !=127.0.0.1
    RewriteCond %{SERVER_ADDR} !=::1
    RewriteRule ^ %{ENV:PROTO}://www.%{HTTP_HOST}%
    {REQUEST_URI} [R=301,L]
</IfModule>
```

Error Prevention

Every website will encounter an error once in a while, the most common of which is the page not found error, most well known as 404. The way in which servers handle errors is by directing any user to a specified page upon identifying one of these exceptional events occurring. Ensuring that your server pages have this handled is one of the most common and critical elements of having a functional site (we talk about this in Chapter 9). Alongside setting error pages, we'll also disable multiviews which prevents Apache from returning an error as the result of a rewrite (unless required, it's better off disabled).

```
Options -MultiViews
ErrorDocument 403 https://example.com/error.html#403
ErrorDocument 404 https://example.com/error.html#404
ErrorDocument 500 https://example.com/error.html#500
ErrorDocument 503 https://example.com/error.html#503
```

MANTRA LABS

Oops! You ran out of oxygen.

The page you're looking for is now beyond our reach.
Let's get you.

Back Home in 00:00:00

HOME PAGE

404

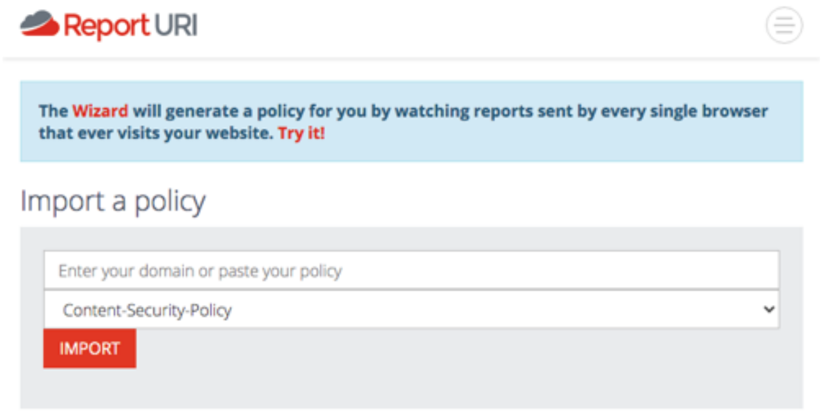
Security Settings

Keeping your website secure is one of the primary aims of the server settings, it's right up there with the performance settings as one of the key things we can tweak within the htaccess files to make real-world benefits to your app or website. With that in mind, let's look at a few of the things you can add that should directly benefit your visitors.

First, you'll want to reduce the risk of any cross-site scripting or content injection attacks using a content security policy. Consider that the below code (which makes a solid default) is just an example of what is possible, and there are plenty of header generators that will build and validate your policy for you for free.

```
<IfModule mod_headers.c>
    Header set Content-Security-Policy "default-src 'self';"
</IfModule>
```

Reference To generate a content security policy for your website, use the following free tool as it'll save you time https://report-uri.com/home/generate/


The Wizard will generate a policy for you by watching reports sent by every single browser that ever visits your website. Try it!

Import a policy

Another security setting that everyone should have in their settings is the indexes option, which prevents anyone from being able to surf through every directory of the website (as you can imagine, being able to browse a site's structure isn't a smart security move).

```
<IfModule mod_autoindex.c>
    Options -Indexes
</IfModule>
```

Another thing you'll want to-do is block access to all hidden files and directories except for visible content within the well-known directory (as that usually contains important stuff like the security.txt file, See Chapter 26). Again, this is a great best practice to ensure that nothing invisible that end-users shouldn't see - is accessible to visitors.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_URI} "!(^|/)\.well-known/([^./]+./?)+$"
    [NC]
    RewriteCond %{SCRIPT_FILENAME} -d [OR]
    RewriteCond %{SCRIPT_FILENAME} -f
    RewriteRule "(^|/)\." - [F]
</IfModule>
```

Additionally, you'll want to block access to any backup or source files that may have been left on the server by IDE's or text editors. The reason behind this is that these may accidentally expose security information to anyone who accesses them. Hopefully, you will do an occasional sweep of your server to tidy away (delete) these files - in addition to relying on these files being hidden by this security switch.

```
<IfModule mod_authz_core.c>
    <FilesMatch "(^#.*#|\.(bak|conf|dist|fla|in[ci]|log|orig|psd|sh|sql|
    sw[op])|~)$">
        Require all denied
    </FilesMatch>
</IfModule>
```

Now something to help prevent cross-origin data leaks and drive-by-download attacks. If you have any executable code or uploadable content, then this particular snippet will be mission-critical to your site. There's not much to explain, it's just something you'll want to include.

```
<IfModule mod_headers.c>
    Header always set X-Content-Type-Options "nosniff"
</IfModule>
```

If your site takes advantage of PHP, there's an attack vector known as fingerprinting, which allows the gaining of sensitive details about the servers' PHP version (probing). To prevent this, include the below.

```
ServerTokens Prod
<IfModule mod_rewrite.c>
    RewriteCond %{QUERY_STRING} PHP[a-z0-9]{8}-[a-z0-9]{4}-[a-
    z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{12} [NC,OR]
    RewriteCond %{REQUEST_URI} =PHP[a-z0-9]{8}-[a-z0-9]{4}-[a-
    z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{12} [NC]
    RewriteRule .* - [F,L]
</IfModule>
```

Reference If you'e producing a website that is built using the CMS WordPress, then there's a bunch of additional security settings you should implement to avoid issues https://www.buyhttp.com/27-most-useful-htaccess-file-tricks-for-your-wordpress-website/

Another available security setting can help you prevent Denial of Service attacks (DoS) relating to user-uploaded content by allowing you to set the maximum file size allowed for uploads. For this case, I've set it at 10mb, but you can choose whatever size you want.

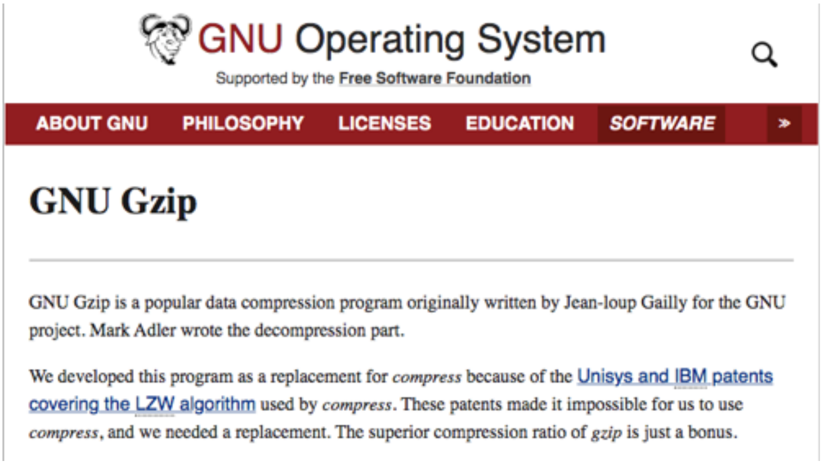
```
LimitRequestBody 10240000
```

Finally, the below code will disable some response headers provided by the server (or server software) offering key information that could be exploited by hackers. Details include if you run Apache, the server-side language (and version) you might be using, and more. This should be included within any htaccess file, as it's a best practice not to leak data which could be used by bad actors to exploit your code.

```
ServerSignature Off
<IfModule mod_headers.c>
    Header unset X-Powered-By
    Header always unset X-Powered-By
</IfModule>
```

Performance Tweaks

Here’s what most people come looking for when they start to dive into an htaccess file (aside from setting a custom 404 page). It’s the performance settings, those handy tips and tricks, that will accelerate their website or app from the laggy slug it began as to the whippet it can potentially become. There’s a few really useful things you can set from the server side, so let’s not waste any time and get tweaking.



GNU Gzip

GNU Gzip is a popular data compression program originally written by Jean-loup Gailly for the GNU project. Mark Adler wrote the decompression part.

We developed this program as a replacement for *compress* because of the Unisys and IBM patents covering the LZW algorithm used by *compress*. These patents made it impossible for us to use *compress*, and we needed a replacement. The superior compression ratio of *gzip* is just a bonus.

First things first, let’s get gzip up and running. What’s gzip? It’s a type of sever-side compression that gives you massive performance benefits in terms of data transfer speeds of text-based files. Your files are made as small as they can be, passed to the users' machine and inflated at their end. This does have a hit in terms of processing power (as the user's machine decompresses), but the benefits are almost always worth it. So what code do we need to get going?

```
<IfModule mod_deflate.c>
    <IfModule mod_setenvif.c>
        <IfModule mod_headers.c>
            SetEnvIfNoCase ^(Accept-EncodXng|X-cept-Encoding|
            X{15}|~{15}|-{15})$ ^((gzip|deflate)\s*,?\s*)+|[X~-]{4,13}$
            HAVE_Accept-Encoding
            RequestHeader append Accept-Encoding "gzip,deflate"
            env=HAVE_Accept-Encoding
        </IfModule>
    </IfModule>
    <IfModule mod_filter.c>
        AddOutputFilterByType DEFLATE text/text text/plain text/
        vcard text/calendar text/vtt text/cache-manifest text/x-
        cross-domain-policy text/vnd.rim.location.xloc text/x-
        component text/markdown text/html application/
        xhtml+xml text/xml application/xml application/rss+xml
        application/atom+xml application/rdf+xml application/
        vnd.google-earth.kml+xml text/css text/javascript text/
        ecmascript application/javascript application/x-javascript
        application/json application/ld+json application/
        manifest+json application/schema+json application/
        geo+json application/x-web-app-manifest+json
        application/wasm font/collection font/eot font/opentype
        font/otf font/ttf application/x-font-ttf application/vnd.ms-
        fontobject image/svg+xml image/vnd.microsoft.icon
        image/x-icon
    </IfModule>
    <IfModule mod_mime.c>
        AddEncoding gzip   svgz
    </IfModule>
</IfModule>
```

There’s not much to the above, it first has a bit of code to force the compression of accept-encoding headers, then it compresses all the MIME types provided in the list (all the file types used in this book are included in the list); and then finally it maps the gzip extension to the svgz encoding type to ensure they uncompress successfully.

Next up, let's give our files a nice set of expires headers for the web browsers cache. By doing so, the files which are updated regularly will get refreshed in the cache as often as needed, and those which rarely see new versions will stick around for longer, saving the visitor valuable bandwidth (and you loading time between pages).

```
<IfModule mod_expires.c>
    ExpiresActive On
    ExpiresDefault "access plus 1 month"
    <FilesMatch "\.(txt|log|html|php|markdown|md|json|geojson|
    topojson|jsonld|manifest|webapp|appcache|webmanifest)$">
        ExpiresDefault "access plus 0 seconds"
    </FilesMatch>
    <FilesMatch "\.(ic[os]|vcard|vcf|vtt|xml|atom|rss|rdf|kml)$">
        ExpiresDefault "access plus 1 hour"
    </FilesMatch>
    <FilesMatch "\.(ico|cur|swf|pdf|doc[x]|xls[x]|ppt[x]|rtf)$">
        ExpiresDefault "access plus 1 week"
    </FilesMatch>
    <FilesMatch "\.(css|js|m?js|otf|eot|tt[cf]|woff2?)$">
        ExpiresDefault "access plus 1 month"
    </FilesMatch>
    <FilesMatch "\.(avifs?|crx|xpi|safariextz|htc|oex|wasm|svgz?|
    bmp|gif|jpe?g|a?png|jxl|tif?f|web[mp]|opus|m4[av]|midi?|
    mp[34]|og[agv]|aac|f4[abpv]|flv|wav|mov|avi|mk[av])$">
        ExpiresDefault "access plus 1 year"
    </FilesMatch>
</IfModule>
```

Finally, we’ll remove ETags as we don’t need them due to the fact all of our files will be provided with expires headers (via caching) giving them a long date before they are due for a refresh. This is better, as controlling when files expire saves bandwidth. Below is the code:

```
<IfModule mod_headers.c>
    Header unset ETag
</IfModule>
FileETag None
```

Bonus Scripts

OK, so what else is there? A few optional extras you can enable.

More CORS Requests

First, let’s have a look at CORS (Cross-origin resource sharing). You know, that most annoying of errors that triggers developers across the world to scream "why won't my file load" when they try to utilize resources outside their main domain. You can set permissions for CORS within your htaccess configuration using the below:

```
<IfModule mod_headers.c>
    Header set Access-Control-Allow-Origin "sub.example.com"
</IfModule>
```

Or, if you really want, just allow all cross-origin requests. Though you’d need to beware the danger in giving other sites unfettered access to use your stuff without permission if you set the below:

```
<IfModule mod_headers.c>
    Header set Access-Control-Allow-Origin "*"
</IfModule>
```

You can also allow access to the timing information in CORS requests if you require it within htaccess. Again, only use if absolutely needed:

```
<IfModule mod_headers.c>
    Header set Timing-Allow-Origin: "*"
</IfModule>
```

Internet Explorer

Yes, Apache even has something to help tackle the old gremlin that has caused problems for developers since the early days of the web. If you still have visitors browsing on Internet Explorer 8-10, then you'll want to include the following in your htaccess file, as it'll trigger the standard's mode automatically. It's a quick and clean IE compatibility fix. If you have users on IE11, document modes were deprecated, so standard's mode is triggered by default (though hopefully, you won't have any users remaining on Internet Explorer to need this).

```
<IfModule mod_headers.c>
    Header always set X-UA-Compatible "IE=edge" "expr=%
    {CONTENT_TYPE} =~ m#text/html#i"
</IfModule>
```

Version: 9.0.8112.16421
Update Versions: RTM (KB982861)
Cipher Strength: 256-bit
Product ID: 03553-236-0200203-71008

Warning: This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

© 2011 Microsoft Corporation

More Security Settings

We’ve previously covered some of the security settings I recommend that you should add into an htaccess file to ensure that your server remains as hardened as possible. Below, I’m going to cover a few of the other things you could do if you wanted to further improve your security settings, though these additions may have consequences for how your project operates on the web (or is seen by others).

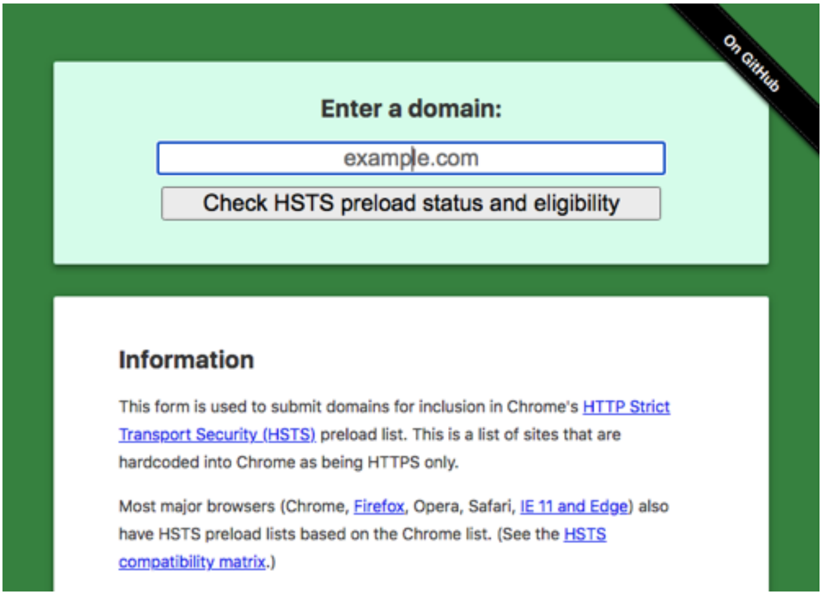
Firstly, we will cover the X-frame-options header, which works to prevent clickjacking on any page it’s utilized on. Sounds like a great idea, right? Well, the problem is that if you utilize it on every page of your site, it’ll break non-malicious framing of your pages (like Google Image search or social network links). It does have its uses though, it's worth using this on pages that lets visitors purchase anything or modify documents (where security must be heightened).

```
<IfModule mod_headers.c>
    Header always set X-Frame-Options "DENY" "expr=%
    {CONTENT_TYPE} =~ m#text/html#i"
</IfModule>
```

Even with HTTPS redirection enabled (which we covered earlier in the chapter), there is still a window of opportunity between the initial HTTP connection and the redirect that occurs, in which an attacker could downgrade or redirect a users' request. By using HSTS headers, a browser will only browse via HTTPS, regardless of what the user types. The downside is that if you experience any certificate issues, the user will error out and any HTTP attempt will fail. This is why you must be sure you want to trigger this, as it’s non-revokable.

```
<IfModule mod_headers.c>
    Header always set Strict-Transport-Security "max-
    age=16070400; includeSubDomains" "expr=%{HTTPS} == 'on'"
    Header always set Strict-Transport-Security "max-
    age=31536000; includeSubDomains; preload" "expr=%{HTTPS}
    == 'on'"
</IfModule>
```

Once you're all setup, submit your site at https://hstspreload.org/



Information

This form is used to submit domains for inclusion in Chrome's HTTP Strict Transport Security (HSTS) preload list. This is a list of sites that are hardcoded into Chrome as being HTTPS only.

Most major browsers (Chrome, Firefox, Opera, Safari, IE 11 and Edge) also have HSTS preload lists based on the Chrome list. (See the HSTS compatibility matrix.)

The below code is offered in the HTML5 boilerplate as an optional extra to help prevent cross-site scripting attacks. However, as noted in the documentation, it's not foolproof and shouldn't be relied upon fully. I've included it as it could be useful for scripting beginners to deflect some basic attacks, but it shouldn't be relied upon as sanitizing code and form inputs and validating syntax is the best practice. Don't let yourself become a target and use the below alongside other tools.

```
<IfModule mod_headers.c>
    Header always set X-XSS-Protection "1; mode=block" "expr=%
    {CONTENT_TYPE} =~ m#text/html#i"
</IfModule>
```

Another optional security snippet is setting a referral policy, which can help mitigate information leakage. Using the referrer-policy header, it targets resources that can request other resources. This sounds great, but you could use a more targeted approach to prevent leakage by using the no-referrer header (though this may impact analytics apps).

```
<IfModule mod_headers.c>
    Header always set Referrer-Policy "strict-origin-when-cross-
    origin" "expr=%{CONTENT_TYPE} =~ m#text\/(css|html|
    javascript)|application\/pdf|xml#i"
</IfModule>
```

Modern browsers have (built into them) the ability to prevent users' credentials being stolen via JavaScript; as they disable the Trace method by default. However, other methods of sending Trace commands can be exploited by attackers. While these situations are becoming rare due to Java, Flash (and other web plugins) being terminated (and disabled) at the browser point, you may wish to enable this if you utilize such a plugin via your app or site.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_METHOD} ^TRACE [NC]
    RewriteRule .* - [R=405,L]
</IfModule>
```

Finally, there is a permission's policy that exists within the header syntax that mitigates browser access to certain hardware features. While it's up to individual browsers to block such requests, it's useful for the privacy conscious individual that doesn't want Geodata or camera access being triggered accidentally (as an example).

```
<IfModule mod_headers.c>
    Header always set Permissions-Policy
    "accelerometer=(),autoplay=(),camera=(),display-
    capture=(),document-domain=(),encrypted-
    media=(),fullscreen=(),geolocation=(),gyroscope=(),magnetom
    eter=(),microphone=(),midi=(),payment=(),picture-in-
    picture=(),publickey-credentials-get=(),screen-wake-
    lock=(),sync-xhr=(self),usb=(),web-share=(),xr-spatial-
    tracking=()" "expr=%{CONTENT_TYPE} =~ m#text\/(html|
    javascript)|application\/pdf|xml#i"
</IfModule>
```

Extra Performance

Along with the settings that were mentioned earlier, there are a few additional things you can do to improve your app or website's overall performance. Just be aware that what's listed below are optional extras, and they will either have an impact on how you deploy your site or you will find there are additional things you need to-do for them to be useful, which is why I’ve listed them here as extras (due to their technical nature). Whether they are worth it is entirely up to you.

If you're intent on using the more performant Brotli over gzip to compress content, you can use the below htaccess code snippet to serve it via your server. The one thing you'll need to-do is ensure that you've generated Brotli encoded files to serve them using this method. This means compressing before deploying.

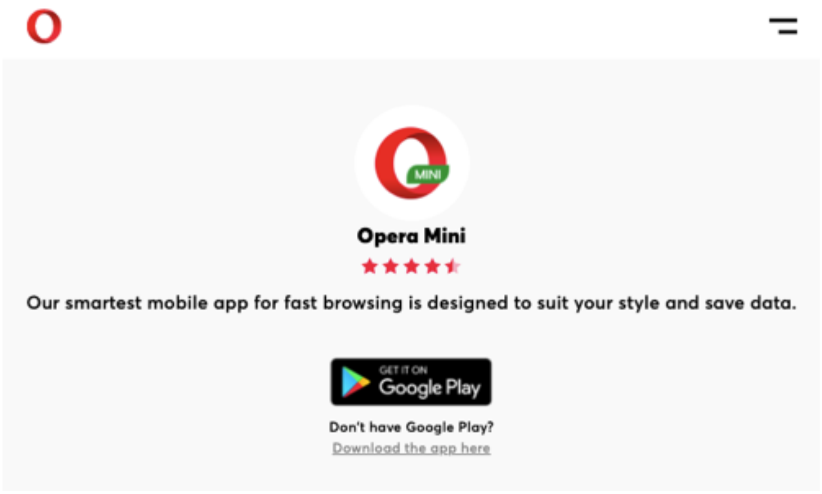
```
<IfModule mod_headers.c>
    RewriteCond %{HTTP:Accept-Encoding} br
    RewriteCond %{REQUEST_FILENAME}\.br -f
    RewriteRule \.(css|ics|js|json|html|svg)$ %{REQUEST_URI}.br [L]
    RewriteRule \.br$ - [E=no-gzip:1]
    <FilesMatch "\.br$">
        <IfModule mod_mime.c>
            RemoveLanguage .br
            AddType text/css   css.br
            AddType text/calendar   ics.br
            AddType text/vcard   vcf.br
            AddType text/javascript   js.br
            AddType application/json   json.br
            AddType text/html   html.br
            AddType image/svg+xml   svg.br
            AddCharset utf-8 .css.br .ics.br .js.br .json.br
        </IfModule>
        Header append Vary Accept-Encoding
    </FilesMatch>
    AddEncoding br .br
</IfModule>
```

If you are into pre-compressing content, and you're not into using Brotli, you could instead serve gzip files (if you've encoded them).

```
<IfModule mod_headers.c>
    RewriteCond %{HTTP:Accept-Encoding} gzip
    RewriteCond %{REQUEST_FILENAME}\.gz -f
    RewriteRule \.(css|ics|js|json|html|svg)$ %{REQUEST_URI}.gz [L]
    RewriteRule \.gz$ - [E=no-gzip:1]
    <FilesMatch "\.gz$">
        <IfModule mod_mime.c>
            RemoveType gz
            AddType text/css   css.gz
            AddType text/calendar   ics.gz
            AddType text/vcard   vcf.gz
            AddType text/javascript   js.gz
            AddType application/json   json.gz
            AddType text/html   html.gz
            AddType image/svg+xml   svg.gz
            AddCharset utf-8 .css.gz .ics.gz .js.gz .json.gz
        </IfModule>
    </FilesMatch>
    AddEncoding gzip .gz
</IfModule>
```

This next Apache feature is only something you should enable if you have no alternative. If you find that browser data saving tools (such as those on mobile browsers), proxy tools, or other user "page altering" agents are damaging your website's ability to render (and that's not the aim of those tools to-do so), you can utilize the below to turn off content transformation (the ability to re-write HTML). Note that this could cost your visitors some much-needed performance!

```
<IfModule mod_headers.c>
    Header merge Cache-Control "no-transform"
</IfModule>
```


Opera Mini

Our smartest mobile app for fast browsing is designed to suit your style and save data.

Don't have Google Play?
Download the app here

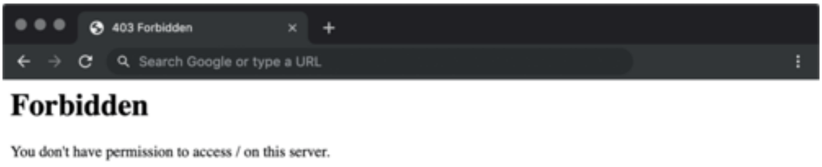
Finally, if you're not using any kind of build system (you should be using something like Git to manage projects), or if you don't have some kind of version control system in place (again, this is something you should have), there is a basic Apache code snippet which can help cache-bust issues with new releases. Though it's better to have a proper versioning system in place, this makes a handy alternative.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^(.+)\.(\w+)\.(avifs?|bmp|css|cur|gif|ico|jpe?g|m?js|
    a?png|svgz?|webp|webmanifest)$ $1.$3 [L]
</IfModule>
```

Reference If you want even more tips and tricks than this chapter provides, one great resource to check is AskApache. https://www.askapache.com/htaccess/

Other Information

Before we leave this chapter, there are a few other things in Apache we should probably cover in terms of use cases which might come in handy when you're running your site or app. Hopefully these snippets will come in handy and help you manage your site more effectively.



Forbidden

You don't have permission to access / on this server.

First, let's talk about bad actors. There may come a time when you need to block an individual (or numerous people from visiting your site) as they're abusing your service. The below snippet should work well, blocking one IP address at a time, or block an entire range of addresses in one go. To find an IP, check the server logs for details.

```
order allow,deny
# Block 1 IP
Deny from 11.22.33.44
# Block IP Range
Deny from 11.22.33.44 99.88.77.66
allow from all
```

Moreover, if you want to prevent image hotlinking on your project, there's a handy script you can use; though in these social networking days you might lose a few fans if visitors cannot share their love of your craft - it's just a bit less annoying than blocking right click.

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{HTTP_REFERER} !^$
    RewriteCond %{HTTP_REFERER} !^http://
    (www\.)example.com/.*$ [NC]
    RewriteRule \.(avifs?|a?png|gif|jpe?g|svgz?|webp)$ https://
    example.com/nolink.gif [R,L]
</IfModule>
```

Reference

If you would like to automate creating an htaccess file, you can at https://assetconfig.com/ and for NginX files Digital Ocean has a tool available at https://www.digitalocean.com/community/tools/nginx

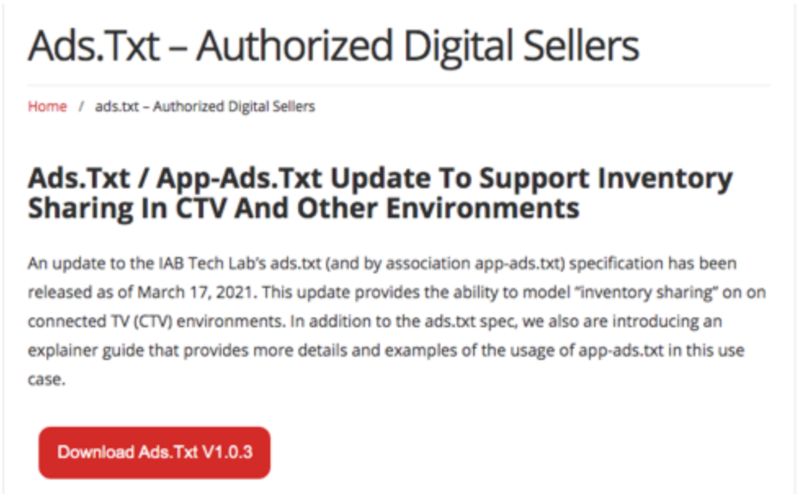
Chapter 3:

Ensuring your sales routes are secure

Ensure that advertisement fraud is reduced using a simple SEO friendly text file to verify sales channels.

ads.txt

The Authorized Digital Sellers file (or Ads for short) is one of the more recent assets to be adopted in web development, but don't assume that because it's a late bloomer, it's not an important part of the web. Recent statistics show that within the top 1,000 most visited websites, over 50% utilize this file already (and it's growing rapidly). Considering this has only been around since 2017, it sounds widespread enough to be taken seriously, so let's find out what it does and if you need it.



Ads.Txt – Authorized Digital Sellers

Home / ads.txt – Authorized Digital Sellers

Ads.Txt / App-Ads.Txt Update To Support Inventory Sharing In CTV And Other Environments

An update to the IAB Tech Lab's ads.txt (and by association app-ads.txt) specification has been released as of March 17, 2021. This update provides the ability to model "inventory sharing" on on connected TV (CTV) environments. In addition to the ads.txt spec, we also are introducing an explainer guide that provides more details and examples of the usage of app-ads.txt in this use case.

Well, as you might imagine with a name like ads.txt it's all about online advertising, specifically it's about helping to prevent counterfeit ads from getting through the net. See, by having a list of authorized sellers within this file, it reduces the chance of unauthorized reselling via domain spoofing (ad fraud) occurring. One of the great things about the ad's file is that Google has been quick to adopt it, so it already has major ad backing and recognition within the advertising industry.

This document won't protect you from invalid traffic or repeat click fraud (which advertisers have their own ways of detecting), but the good thing is it's perfectly compatible with any existing systems you have in place to tackle other forms of ad fraud. So if you have adverts, it's a given you'll want to implement it on your site. Sounds like a plan? Get started by creating an ads.txt file in your root directory (you can also include it in subdomains additionally if you serve ads there).

Reference	For a comprehensive guide to the ads.txt format, you can read the full specification and explainer guide at https://iabtechlab.com/ads-txt/

Comments

As with many text files, you can include comments in your ads file by preceding your text with a hash (#) character. This can be useful for keeping track of which advertisers belong to which groups you sell too. The below is a pretty basic example of what you could include.

```
# This is a comment
```

Records

Each record will consist of four variables, as denoted below. To help you fill in the four variables, a table is also provided on the next page, listing what you need to put within each section. For each advertiser you trust, you will need to list them individually using the four variables provided below. You can provide as many as you require.

```
<V1>, <V2>, <V3>, <V4>
```

Each variable will be comma separated. An example is shown below to give you an idea of what a complete advertiser record looks like:

```
example.com, 12345, DIRECT, f794e0a46588c21f
```

Variable	Example	Required	Description
<V1>	example.co m	Yes	Domain name of the system that advertisers connect too.
<V2>	12345	Yes	The publishers account ID used in ad transactions.
<V3>	DIRECT	Yes	Either use DIRECT or RESELLER, who controls the account.
<V4>	f794e0a4658 8c21f	No	An ID for an advert system within a certificate authority.

Extensions

The ads.txt file supports a few extensions which might be useful to developers, which have to be mentioned too. The first of which is if you are using subdomains. If you redirect your main domain to a subdomain and want advertisers to know that the subdomain is more important than the root domain, you should add the below into your ads.txt in your root domain, and it will bounce advert requests to the ads.txt file on your subdomain (yes, you need two files) without issue.

```
SUBDOMAIN=sub.example.com
```

Next, there is a property for ads.txt, which is intended for when you are serving advertisements from a partner. By using this property, you can avoid having to list every domain (relationship) you have with that advertising partner, as you'll need to declare the domain that your website is connected to (in terms of authorizing monetization of ads).

```
INVENTORYPARTNERDOMAIN=example.com
```

Next, there is a property which simply allows you to specify what business website owns the website or application. If there is more than one owner, then the first (or primary) owner should be listed. It's highly recommended that this property be included.

```
OWNERDOMAIN=example.com
```

Next, there is a property which allows you to specify the monetization partner of the domain listed. This is an optional property that should exclusively be used for a seller who isn't the publisher of the work.

```
MANAGERDOMAIN=example.com
```

Finally, while it's preferable to use the humans.txt file for contact details (and we'll talk about this useful format later in the book), the ads.txt file has a property that can be used to list advertiser contact information. Please bear in mind it could potentially be scraped and targeted by spammers as it will be stored in plaintext, so whether you do want to provide an email address in this variable is up to you.

```
CONTACT=email@example.com
```

And that's just about all there is to it. A text file with comments, four variables and a couple of extensions. It's a simple specification that has widespread adoption and reduces the amount of advertising fraud in the web industry. A very useful file if you rely on this form of income on your website or app. All you'll need to-do in the long run is ensure that you maintain it (weeding out redundant ads) over time.

Reference For details relating to how Google handles ads.txt files you can check this guide https://support.google.com/adsense/answer/7532444?hl=en-GB

Chapter 4:

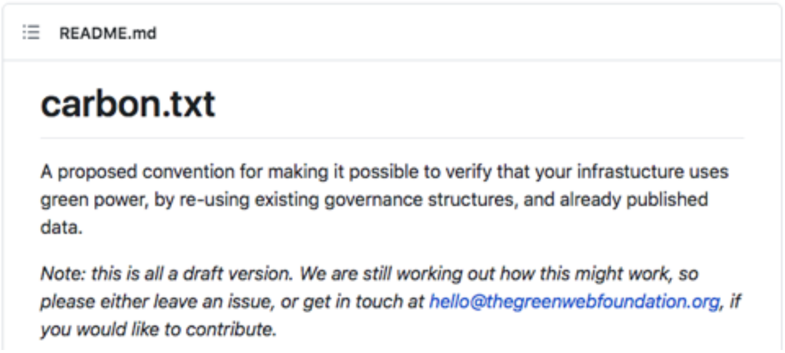
Getting your web project eco-friendly

Use this emerging standard to both track and validate your businesses digital green credentials.

carbon.txt

Having a performant website these days is important for ensuring that your site loads well on a variety of devices. But being performant has additional benefits, such as being able to make your website or app eco-friendly - and by that, I mean that it's energy efficient and ethical in how it both gets and uses the electrical resources it has access too.

Perhaps it's something you're interested in, or perhaps it's not top of your agenda, but it's a fact that the web burns through a heck of a lot of electricity. Whether server-side (processing scripts or loading data) or client-side (via hardware use), being able to reduce the carbon impact of your site can help save the earth, users battery life, and power bills. With this in mind, it's time to introduce you to carbon.txt.



README.md

carbon.txt

A proposed convention for making it possible to verify that your infrastucture uses green power, by re-using existing governance structures, and already published data.

Note: this is all a draft version. We are still working out how this might work, so please either leave an issue, or get in touch at hello@thegreenwebfoundation.org, if you would like to contribute.

Using this file, visitors will be able to track the path "upstream" from provider to provider, identifying the route generated energy travels; (starting from your host) to show a sustainability trail. As it stands, this file isn't utilized by any third parties like search engines, however, visitors who are interested in your sites' transparency may go looking for this file (if they are aware of it) so it may be useful.

While it's still a draft specification and hasn't seen much activity lately, by including this file in your base directory (with the necessary code below) you can be transparent about how your servers are powered, and how green your site really is. The formatting follows other text files such as a robots file or, as in the previous chapter, the ads.txt file.

Reference For a comprehensive guide to how a carbon.txt file should be formatted, check the specification at https://github.com/thegreenwebfoundation/carbon.txt

Comments

As with many text files, you can include comments in your carbon file by preceding your text with a hash (#) character. You can also use comments to provide additional details about the provider, such as whether they offset energy use via carbon credits, or power their sites using renewable energy such as wind, solar or water. You could even provide any percentages involved (in mixed cases) for visitors.

```
# This is a comment
```

Upstream

Every record will need to provide two space separated variables, which are denoted below. To assist you filling in the two variables, a table has been provided, listing what you need to put in each field.

You'll have to provide one set of values for each host you wish to declare any eco credentials for. If you would rather not rely on others having their own carbon file, you could do your research and log the full path of your energy journey including any CDN's, third parties, or payment portals in the file (be sure to document using comments).

```
<V1> <V2>
```

Variable	Example	Required	Description
<V1>	example.co m	Yes	URL of your hosting provider.
<V2>	london	No	Datacenter that is eco-friendly.

Finally, I've included a code sample below to give you an idea of what to expect in your file. We must enclose the word upstream in squared brackets on a new line. Next, include the space separated variables to show how visit comes from a green pathway. This is arguably one of the smallest files you'll ever produce. Though, it'll take time to-do the necessary research if you want to be inclusive.

```
[upstream]
example.com london
```

And that concludes this chapter. Hopefully, this file will gain adoption on a wider level in the future, and with the world focusing heavily on issues like climate change, I suspect this will be the case. Until then, it's an optional file that can help you become more transparent about your project's sustainability. And as a basic file only takes a minute to build, it's just worth adding into your next project as it doesn't have any downsides in terms of performance and needs little maintenance.

Reference	To find out how green any third party tools or scripts a site uses are, use: https://aremythirdpartiesgreen.com/

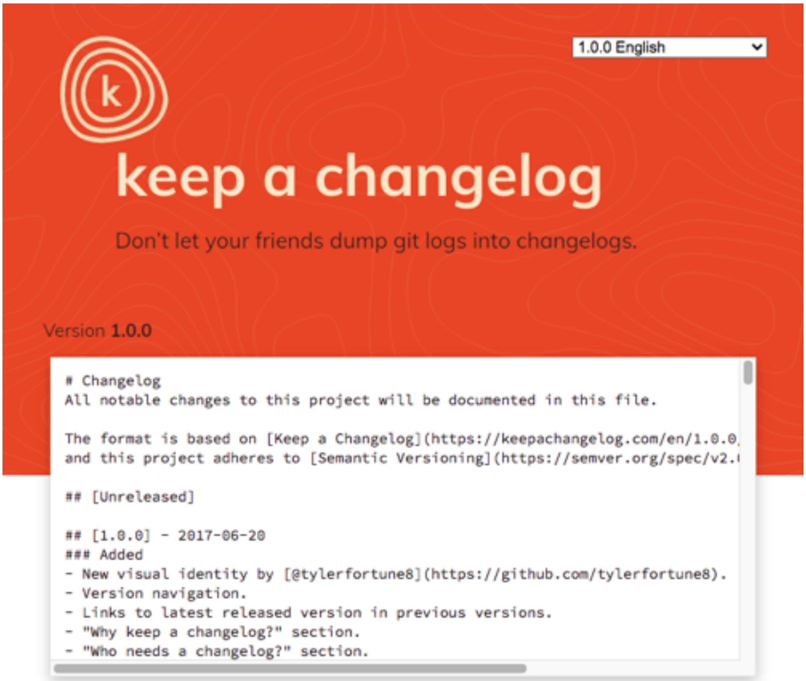
Chapter 5:

Show your development progress with ease

Build a changelog that identifies additions, changes, deprecations, removals, fixes, and security patches.

change.log

When you create an app or website with app-like functionality (say a SAAS), it's important to let users know what developments are being made to the product. In an offline app, this is usually done through a list provided in a "Check for updates" feature, or through notifications in the app store, or even a list of updates made on the app's site.



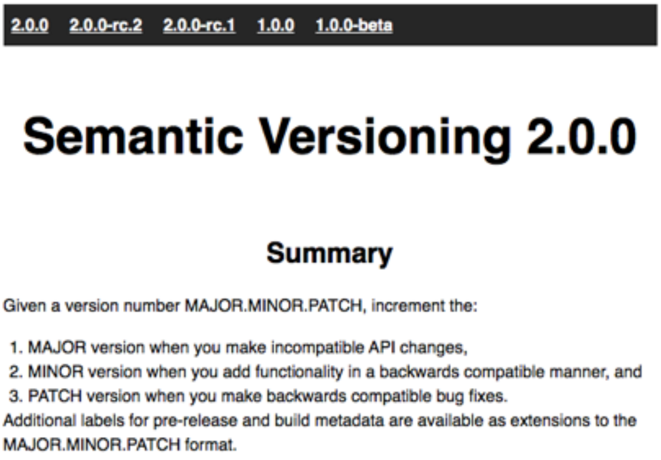
```
# Changelog
All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](https://keepachangelog.com/en/1.0.0
and this project adheres to [Semantic Versioning](https://semver.org/spec/v2.

## [Unreleased]

## [1.0.0] - 2017-06-20
### Added
- New visual identity by [@tylerfortune8](https://github.com/tylerfortune8).
- Version navigation.
- Links to latest released version in previous versions.
- "Why keep a changelog?" section.
- "Who needs a changelog?" section.
```

Reference The best overview of changelogs which I consider the gold standard is here https://keepachangelog.com/

For a website or app that primarily lives online and changes regularly, tracking changes can be trickier, yet is just as essential for many visitors. As such, we need a centralized method of giving our visitors simple, easy to find and read details of what's new or updated and where to locate it. Enter the change.log - a plaintext file that sits in your base directory and acts as a clean, maintained, human-readable guide to the entire history of your digital product or service.

Versioning

First things first, a changelog isn't intended to be where you dump git logs, this text file should be for people, not machines. It's curated and cleaned up, so users can read it quickly (with links to read more if they choose to). There should be an entry for each version of a product, with the release date (YYYY-MM-DD) displayed alongside. Moreover, ensure the latest version always comes first in the list of changes.



Semantic Versioning 2.0.0

Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

Furthermore, you should try to work with semantic versioning, which works off the principle of vA.B.C with each being replaced with one of the following variables (the table below shows its meaning):

Version	Type	Description
A	Major	You make incompatible API changes.
B	Minor	You add backward compatible functionality.
C	Patch	You fix bugs that don't break existing functionality.

Changes

Additionally, when you're entering the details of what's changed to your product, similar types of changes should be grouped together, I've provided a list of the various variables in the table below:

Change	Description
Added	New features added into the product.
Changed	Changes to existing functionality.
Deprecated	Soon-to-be removed product features.
Removed	Product features that have been removed.
Fixed	Bug fixes or patches to the product.
Security	Vulnerabilities which have been resolved.
Unreleased	Features that will appear in future versions.

Summary

And that's about it. Aside from ensuring you label your document as a changelog at the top of the file to ensure visitors know what they're going to be reading, there's not much else to know. Below I'm going to provide you with a template change.log which contains various useful snippets for a first-time launch (based on the files in this book).

```
# Changelog - Website Name Here

## [Unreleased]
- N/A

## 1.0.0 - YYYY-MM-DD
### Added
- change.log File with transparent version tracking.
- Design System compiled to match product expectations.
- .htaccess File with Headers, Rewrites, Security, & Performance modules.
- index.html & error.html File with metadata & content.
- Licensed SVG images along with any generated external JPEG & PNG.
- style.css with Base, RWD, Animate, Modes, Print, A11y & Quirks sheets.
- <Font> Typeface for Personal brand (x1 license per site).
- <Font> Typeface from Google Fonts (self hosted).
- LICENSE ADDED - Check: https://website.com/license.html
- humans.txt File with Website creation credits.
- security.txt File with contact information.
- robots.txt File with integrated sitemap link.
- favicon.ico File with PNG's for iOS, Android, etc.
- site.webmanifest File & browserconfig.xml for PWA's.
- event.ics File with Calendar events for the website.
- vcard.vcf File with Contact information for clients.
- rss.xml, atom.xml, feed.json & feed.opml for syndication.
- dublin.rdf, foaf.rdf, geo.kml & geo.rdf for DCMI metadata.
- sitemap.xml File with page listings and IA for site.
```

- opensearch.xml File with basic Google search integration.
- subtitles.vtt File with accessible videos across site.
- carbon.txt File for verifying eco-friendly credentials.
- p3p.xml, dnt-policy.txt, crossdomain.xml & clientaccesspolicy.xml for security.
- PICS.rdf & powder.xml for child-friendly platforms.
- script.js File with Service Worker & linted, minified JavaScript.
- Performance metrics indexed and optimizations provided.

## [Guide]
- Added: New features.
- Changed: Altered functionality.
- Deprecated: Disappearing features.
- Removed: Removed features.
- Fixed: Bug fixes.
- Security: Any vulnerabilities.

Chapter 6:

Keeping Microsoft Silverlight functional

Now obsolete due to Microsoft Silverlight being redundant, you can build the streaming policy file here.

clientaccesspolicy.xml

When deciding to write this book, chapters such as this were a matter of debate whether I should even include them at all. On one hand, this file has played a critical part in many peoples sites for the best part of the last decade because of the technology behind them.

Yet, within the next year (October 12, 2021 to be exact), the framework this supports - and with it the file, will become deprecated. Should this book cover obsolete files? Well, they're a part of the web's history, and some people dealing with old systems may come across (and have to support) them. So I say yes - though do so tentatively.

With this in mind, allow me to introduce you to clientaccesspolicy.xml, which is quite a mouthful of a filename. It's another one for the root of your app or websites structure, though instead of being a text file it uses the more complex XML syntax (think a cross between HTML and JSON, but with stricter rules as if you make a typo it won't work).

Prepare for Silverlight 5 end of support after October 2021. Learn more >

Get Silverlight 5

Silverlight is a powerful development tool for creating engaging, interactive user experiences for Web and mobile applications. Silverlight is a free plug-in, powered by the .NET framework and compatible with multiple browsers, devices and operating systems, bringing a new level of interactivity wherever the Web works.

Get the Silverlight 5: A powerful tool for creating interactive web and mobile applications.

Reference If you would like to learn more about Silverlight, read the Wikipedia article at https://en.wikipedia.org/wiki/Microsoft_Silverlight and then visit it's developer portal at https://www.microsoft.com/silverlight/

This file has one sole purpose, which is to allow Microsoft Silverlight apps - an alternative to the popular Flash format, to send limited cross-site requests. Why would you want to-do this? Well out-of-the-box, Silverlight only allows same-site requests except for images and media. This means that apps can't connect to CDN's or subdomains. It's done for security to prevent cross-site forgery exploits.

Boilerplate

Do you use Microsoft Silverlight, if not, skip this chapter. Do you have visitors on IE11 or lower? If not, skip this chapter. Do you only need images or media? If so, you don't need this file. If you do meet the criteria for this niche format, just put the below code into your file "as-is". There aren't any variables that you'll need to edit to get it working.

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
    <cross-domain-access>
        <policy>
            <allow-from http-request-headers="SOAPAction">
                <domain uri="*"/>
            </allow-from>
            <grant-to>
                <resource path="/" include-subpaths="true"/>
            </grant-to>
        </policy>
    </cross-domain-access>
</access-policy>
```

Reference

To learn more about configuring a crossdomain.xml file, read the following https://docs.microsoft.com/en-us/previous-versions/windows/silverlight/dotnet-windows-silverlight/cc197955(v=vs.95)

Chapter 7:

Allow Adobe Flash to run despite retirement

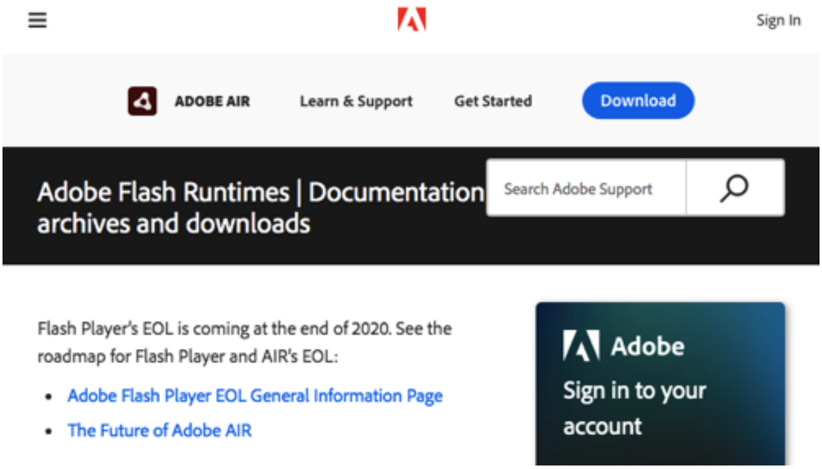
Now obsolete due to Adobe Flash's retirement, you can still build the streaming allowance policy file here.

crossdomain.xml

Unlike the previous chapter's content which followed a file that is due to be deprecated later this year, this chapter examines a file that has just been deprecated at the start of the year (on 12 Jan 2021). The crossdomain.xml document, which is another XML file that gets put the in the base directory of your website, serves the same purpose as clientaccesspolicy.xml (preventing cross-site forgery). Though rather than working explicitly for Microsoft Silverlight, its primary purpose is to work for the now deprecated (end-of-life) Adobe Flash format.

Reference To understand the difference between Silverlight and Flash XML files, visit https://social.msdn.microsoft.com/Forums/SqlServer/en-US/e7241343-3408-478f-aac7-f381b9ba21c8/difference-between-clientaccesspolicyxml-and-crossdomainxml

Adobe Flash, had a much wider user-base than Silverlight, so there may be more user-cases for the crossdomain.xml file (if your SWF needed to access any files beyond the primary domain). But as Adobe has ceased support for the format, browser vendors are now disabling support and the file's use has quickly become limited.



Sign In

ADOBE AIR   Learn & Support   Get Started   Download

Adobe Flash Runtimes | Documentation archives and downloads

Search Adobe Support

Flash Player's EOL is coming at the end of 2020. See the roadmap for Flash Player and AIR's EOL:

- Adobe Flash Player EOL General Information Page
- The Future of Adobe AIR

Adobe
Sign in to your account

If you find a use for this file on a system that still needs to utilize Flash, perhaps say an old app that can't afford to transition, or a game built in the format that needs preserving; being able to use this file will be useful. While the specification does allow a fair amount of tweaking, to keep things simple I'm going to provide a few templates that Adobe themselves offer as boilerplates, after all, it's a rare occasion when you'll need to use this file in a real-world situation any more.

Reference	For a comprehensive guide to crossdomain.xml files, check the specification at https://www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/CrossDomain_PolicyFile_Specification.pdf

Moderate

The below example allows Flash to access data from the root domain, including any subdomains or Adobe Flash servers, via SOAP:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/
xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="master-only"/>
    <allow-access-from domain="*.example.com"/>
    <allow-access-from domain="www.example.com"/>
    <allow-http-request-headers-from domain="*.adobe.com"
    headers="SOAPAction"/>
</cross-domain-policy>
```

Safest

The below example is the most restrictive policy, as it'll block your Flash files from requesting data from any domain or server:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/
xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="none"/>
</cross-domain-policy>
```

Dangerous

The below example is the least restrictive policy, as it'll allow your Flash files to request data from any domain or server they choose:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/
xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="all"/>
    <allow-access-from domain="*" secure="false"/>
    <allow-http-request-headers-from domain="*" headers="*"
    secure="false"/>
</cross-domain-policy>
```

Using one of these three pieces of code, you should cover any use-case of Flash wanting to access external files on a website. Aside from that, there's nothing else that needs to be said. It's a simple basic file that serves one basic function, so you can move to the next chapter.

Reference For a quick start guide to the specification and how to implement crossdomain.xml, check https://www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/xdomain.html

Chapter 8:

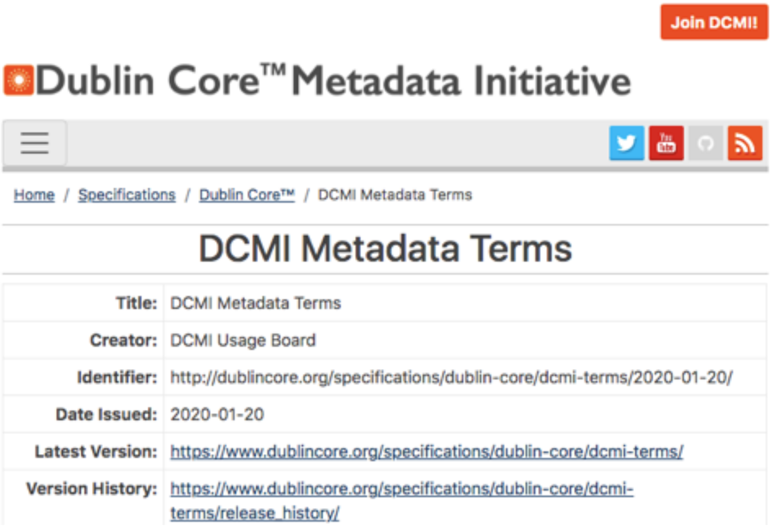
Optimizing semantic metadata for RDF

If you rely on RDF metadata, use this generator to quickly build a profile index of your site for SEO purposes.

dublin.rdf

Getting noticed is hard, especially when your site is one among the sea of millions in search engine results out there. One of many ways to get yourself visible is through search engine optimization, which is a method of using techniques to help those looking for you find you easier. Doing this ethically is tricky, but using rich metadata remains one of the most popular ways of organically signposting yourself.

Recently, the most popular method of using metadata is via microdata (such as schema) or the older pattern of microformats, which works equally well. Yet, a more complex but solid choice that is recognized by search engines exists that we can use to build an index file for your app or website that provides a profile all in one location.


Join DCMI!
Dublin Core™ Metadata Initiative
Home / Specifications / Dublin Core™ / DCMI Metadata Terms

DCMI Metadata Terms

Title:	DCMI Metadata Terms
Creator:	DCMI Usage Board
Identifier:	http://dublincore.org/specifications/dublin-core/dcmi-terms/2020-01-20/
Date Issued:	2020-01-20
Latest Version:	https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
Version History:	https://www.dublincore.org/specifications/dublin-core/dcmi-terms/release_history/

RDF is a format that is a lot like XML, very strict in its nature and prone to glitching on any syntax error - which is probably why it didn't have a very high adoption rate. Yet, for simple files that do one simple task, it's not going to be too troublesome to utilize its benefits. For this file, we're going to use the Dublin Core metadata initiative, which we examined in the first chapter's meta section (as it's a solid standard).

Reference For more information about DCMI, check the spec at https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ or read this handy usage guide at https://www.dublincore.org/specifications/dublin-core/usageguide/2000-07-16/simple-html/

Reference

So what do you need to-do? Well, first you'll want to create a file called dublin.rdf and place it somewhere on your site. Unlike previous files, no browser seeks the file automatically, so there's no need for it to be in the base directory unless you want it to be there. Though, for this reason, you will need a reference in the head of your HTML files so search engines can find and take advantage of the metadata when indexing your data. The exact code you'll need is provided below.

```
<link rel="author" type="application/rdf+xml" href="dublin.rdf">
```

Example

Next, you'll want to produce the code for the file itself. As I mentioned before, RDF can be a tricky file format, so I've produced a boilerplate below that you can work from. In addition, you'll find a table which showcases all the possible variables you can tweak in your profile.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms="http://purl.org/dc/terms/">
    <rdf:Description rdf:about="https://example.com/">
        <dc.title>Title</dc.title>
        <dc:creator>Name</dc:creator>
        <dc.contributor>Name</dc.contributor>
        <dc:date>YYYY-MM-DD</dc:date>
        <dc:description>Description</dc:description>
        <dc.type>Category</dc.type>
        <dc.subject>Subject, Subject</dc.subject>
        <dc:language>en</dc:language>
        <dc.coverage>Location</dc.coverage>
        <dc:publisher>Publisher</dc:publisher>
        <dc:source>https://example.com/</dc:source>
        <dc.relation>https://example.com/</dc.relation>
        <dc.rights>https://example.com/</dc.rights>
        <dc.identifier>000-0-00-000000-0</dc.identifier>
    </rdf:Description>
</rdf:RDF>
```

Variable	Description
dc.contributor	If you have multiple contributors to a resource, you can include them here.
dc.coverage	Include place names or coordinates to show the scope of the document.
dc.creator	The person or business responsible for the app or site will be added here.
dc.date	Using YYYY-MM-DD you can provide a precise timestamp for content.
dc.description	Offer a brief explanation or a table of contents of what the content is about.
dc.format	If the content isn't in HTML (default), use this to offer the file MIME type.
dc.identifier	Provide a link to a URL, DOI, ISSN or ISB which references the content.
dc.language	Provide details of the locale in which th content is offered to visitors.
dc.publisher	Provide details of the organization or individuals responsible for content.
dc.relation	If there are any links which connect to the content, provide the reference.
dc.rights	License information, copyright details o links to agreements are used here.
dc.source	If the content cites any sources, you ca credit the references with this tag.
dc.subject	Offer details of the content's subject using semicolon separated strings.

Variable	Description
dc.title	Give the article title without branding sc it can be read by aggregators.
dc.type	Shows content categories, functions, genres, or aggregation levels.

Summary

Having this file as part of your organic marketing strategy could make a difference, though, as a word of caution. Ensure your content is relevant, and you don't keyword stuff phrases which don't appear on your pages; otherwise you may suffer penalties from the search engines (so only provide the above variables if they apply to you). The key benefit of the RDF approach over countless meta tags in your page headers is this will be more performant, cache better, and you can update your metadata easier, having it apply across your project.

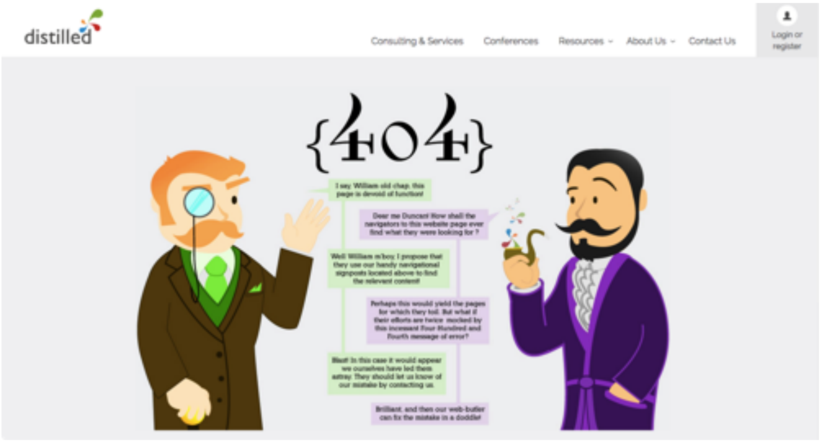
Chapter 9:

Tackle common server errors together

Mistakes happen, so use this error page (and it's anchor points) to cover your problematic endpoints.

error.html

If there's one thing we learn about code (and life), it’s that mistakes will happen. When maintaining a website or app; a broken link, a quirk of a server, or a wrongly entered password could be all that will stand between your visitors and an error page. Therefore, it’s important when users stumble upon such problems that we intercept them and try to help them find their way to either the information they wanted to locate on your website, or something equally useful or valuable.



I say William old chap, this page is devoid of function!

Dear me Duncan! How shall the navigators to this website page ever find what they were looking for ?

Well William m'boy, I propose that they use our handy navigational signposts located above to find the relevant content!

Perhaps this would yield the pages for which they toil. But what if their efforts are twice mocked by this incessant Four Hundred and Fourth message of error?

Blast! In this case it would appear we ourselves have led them astray. They should let us know of our mistake by contacting us.

Brilliant, and then our web-butler can fix the mistake in a doddle!

Many websites have separate error pages for each type of quirk that can occur, from the well known 404 (page not found) to the 500 (internal server error). My personal preference, and one I think is well worth adopting, uses a single page with fragment links for each type of error (reducing HTTP overheads and maintaining errors easier). This is the type of page we'll be building below, and it uses a combination of pure HTML and CSS. There is no requirement for JavaScript.

Configuration

Because we'll be controlling error messages within this file, we first need to ensure that the server redirects to the page (and the correct fragment error identifier) when an issue occurs. If you use Apache, edit your htaccess file (or http.conf file) with the below if you haven't already done so, otherwise any errors will use the server default:

```
Options -MultiViews
ErrorDocument 400 https://example.com/error.html#400
ErrorDocument 401 https://example.com/error.html#401
ErrorDocument 403 https://example.com/error.html#403
ErrorDocument 404 https://example.com/error.html#404
ErrorDocument 408 https://example.com/error.html#408
ErrorDocument 408 https://example.com/error.html#410
ErrorDocument 429 https://example.com/error.html#429
ErrorDocument 500 https://example.com/error.html#500
ErrorDocument 501 https://example.com/error.html#501
ErrorDocument 502 https://example.com/error.html#502
ErrorDocument 503 https://example.com/error.html#503
ErrorDocument 504 https://example.com/error.html#504
ErrorDocument 505 https://example.com/error.html#505
ErrorDocument 508 https://example.com/error.html#508
```

If you use another web server such as NginX or IIS, they'll use another method of delivering custom error pages. You should always refer to the manual for that server software to get it working. You should also try to only include the error codes you feel your app or website is going to need (to avoid unnecessary overhead in your error.html file).

Reference

For errors, use https://github.com/h5bp/server-configs-apache/blob/master/dist/.htaccess (Apache), https://github.com/h5bp/server-configs-nginx/blob/master/h5bp/errors/custom_errors.conf (NginX), and https://www.bruceclay.com/blog/microsoft-iis-custom-404-error-page-configuration/ (IIS).

Webpage

Create an error.html file in the base folder of your app or website so that visitor errors are redirected to the top directory. Unlike the rest of your product, this file should be self-contained to avoid suffering any errors if say any external images, CSS or other resources disappear.

Finally, let's get building some HTML and CSS. We'll start by adding the necessary HTML, leaving a gap between the style tags for the CSS and another gap for each of the error content between the main tags:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
        initial-scale=1">
        <title>Error - Product</title>
        <meta name="description" content="Description.">
        <style>

        </style>
    </head>
    <body>
        <header>
            <h1>Product</h1>
        </header>
        <main>

        </main>
        <footer>
            <p>Copyright, 2021.</p>
            <p>Built by Your Name.</p>
        </footer>
    </body>
</html>
```

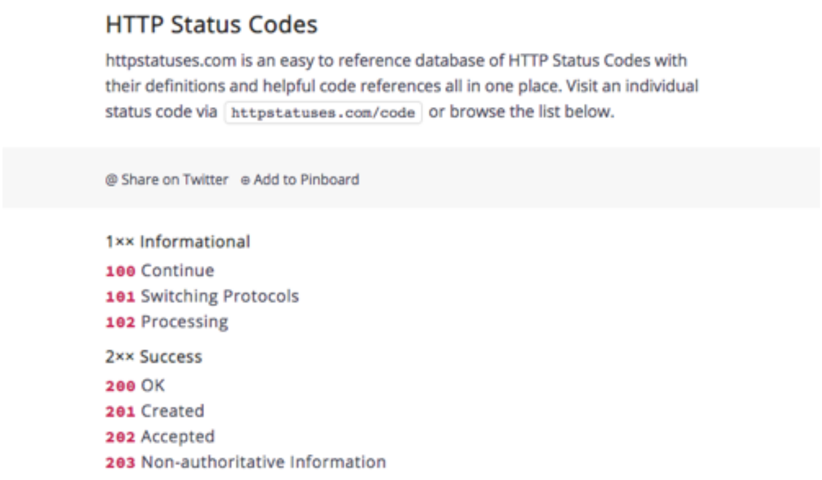
Error Sections

Next we'll add the individual error messages which appear within the main tags. As you can see, I've decided to keep things simple by giving each their number, a title, and a description for users of what's happened. You could go much further for visitors by adding useful links, contact details, downtime detectors, useful reading and more.

```
<article class="error" id="400">
    <h2>400</h2>
    <h3>Bad Request</h3>
    <p>Your browser sent a request that this server could not
    understand. Try Again.</p>
</article>
<article class="error" id="401">
    <h2>401</h2>
    <h3>Unauthorized</h3>
    <p>You cannot access this page until you have logged in and
    authenticated.</p>
</article>
<article class="error" id="403">
    <h2>403</h2>
    <h3>Access Forbidden</h3>
    <p>Sorry, you'll need to login to visit this area.</p>
</article>
<article class="error" id="404">
    <h2>404</h2>
    <h3>Resource Not Found</h3>
    <p>You made a wrong turn. Lets try a different link.</p>
</article>
<article class="error" id="408">
    <h2>408</h2>
    <h3>Request Timeout</h3>
    <p>The website took too long to respond. Try again later.</p>
</article>
<article class="error" id="410">
    <h2>410</h2>
```

```
        <h3>Gone</h3>
        <p>This resource is no longer available. Lets try a different
        link.</p>
    </article>
    <article class="error" id="429">
        <h2>429</h2>
        <h3>Too Many Requests</h3>
        <p>The website has too many visitors. Please try again in a few
        minutes.</p>
    </article>
    <article class="error" id="500">
        <h2>500</h2>
        <h3>Internal Server Error</h3>
        <p>We broke something. Sorry! Refresh, or try again later.</p>
    </article>
    <article class="error" id="501">
        <h2>501</h2>
        <h3>Not Implemented</h3>
        <p>We’re missing something. Let us know to fix this bug.</p>
    </article>
    <article class="error" id="502">
        <h2>502</h2>
        <h3>Bad Gateway</h3>
        <p>We got an error from a service we use. Please try again.</
        p>
    </article>
    <article class="error" id="503">
        <h2>503</h2>
        <h3>Gateway Timeout</h3>
        <p>A service we use didn’t respond to our request. Try again
        later.</p>
    </article>
    <article class="error" id="504">
        <h2>504</h2>
        <h3>Internal Server Error</h3>
        <p>We broke something. Sorry! Refresh, or try again later.</p>
```

```
</article>
<article class="error" id="508">
<h2>508</h2>
<h3>Loop Detected</h3>
<p>We redirected you incorrectly. Lets try a different link.</p>
</article>
```



HTTP Status Codes

httpstatuses.com is an easy to reference database of HTTP Status Codes with their definitions and helpful code references all in one place. Visit an individual status code via `httpstatuses.com/code` or browse the list below.

1×× Informational
100 Continue
101 Switching Protocols
102 Processing

2×× Success
200 OK
201 Created
202 Accepted
203 Non-authoritative Information

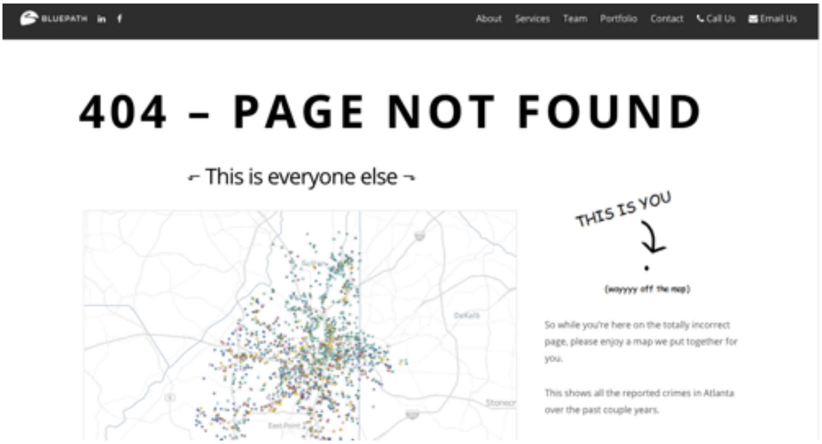
Reference You'll probably want to include the above errors, but there are many other codes you could also support. This site lists each HTTP status https://httpstatuses.com/

CSS Styles

Finally, we need to account for the CSS within the two style tags in the head of the file. This is pretty simple as all you need to-do is hide the content that is not required (errors not triggered) and show the error that has been triggered, and the content to go along with that issue.

```
.error:not(:target) { height: 1px; left: -1000px; overflow: hidden;
position: absolute; top: -1px; width: 1px; }
.error:target { height: auto; left: auto; overflow: auto; position:
relative; top: auto; width: auto; z-index: 1; }
```

Regarding how this error loading method works, each article has been assigned an ID that, upon being targeted using the CSS :target pseudo, it will be made visible (whilst all non targets will be hidden off-screen). Though it's important to note that it won't be hidden from screen readers and accessibility aids as inclusive design is important.



404 – PAGE NOT FOUND

← This is everyone else →

THIS IS YOU

(wayyyy off the map)

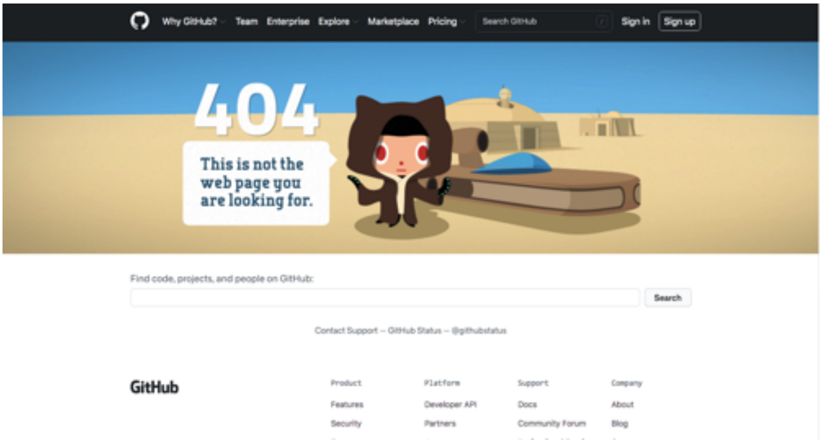
So while you're here on the totally incorrect page, please enjoy a map we put together for you.

This shows all the reported crimes in Atlanta over the past couple years.

Reference An error page is an chance to have fun with the design, if you need inspiration check out https://search.muz.li/inspiration/404-page-not-found-design-inspiration/

Summary

With all of this in place, your all-in-one error page is ready to go. One thing to remember is that if visitors accidentally browse to the error page and no error hash is triggered, nothing will be shown (to avoid confusing the user), You could choose to provide a default article and have that hidden using a CSS sibling selector if you so wished. This page can also be tricky for analytics apps, as all errors get sucked into one location (though server logs should register the individual errors).



If you really need the analytics, you could go back to separate files, but for maintainability and for keeping all possible errors self-contained, I prefer this style of one-pot solution. A single error.html file that can serve your visitors either a simple response to a glitch, or if you put the work in, plenty of useful data to guide them through your product dynamically. It's the best of (most) worlds.

Chapter 10:

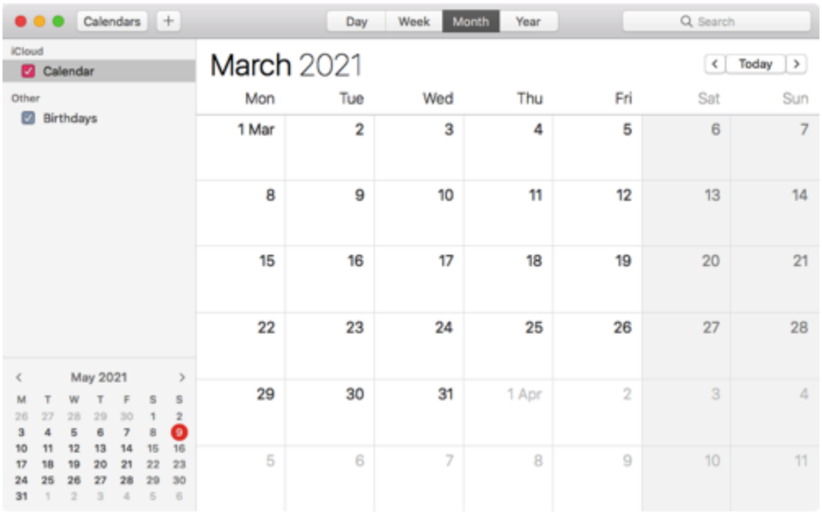
Never miss a future calendar event

Craft downloadable event files that are supported by both desktop and mobile calendar applications.

event.ics

Running an interactive website can be fun and time-consuming, but one thing that can really keep visitors coming back for regular or specific occasions are events. If your site holds timed events like sales, new releases, or if you are planning live sessions (such as streaming media), or want to offer reminders for things like podcasts; having a reminder or a listing in the user's calendar is precisely what you need to avoid lead losses online. This is entirely possible using one tiny file.

The event.ics file can be placed anywhere you like on a server, and you can create as many of them as you require within a project, unlike the previous tiny files we've detailed previously. It uses an unusual plaintext syntax made up of properties and values which are only similar to another file type we'll cover later in this book, the vCard.vcf format (Chapter 32). Once downloaded, event files can be imported into phone, desktop or email, calendar apps quickly and easily.



Regarding properties, as only a few are compatible with popular apps (such as iOS, Mac, Google Calendar & Outlook), we will place most of our focus on elements that have widespread implementation. For other outstanding properties and values, they will be noted but may not offer much value (if one is unsupported, it will be ignored). For device specific-properties, they will be listed additionally.

Reference For a comprehensive guide to using the iCalendar syntax, check out the specification at https://tools.ietf.org/html/rfc5545

Event Creation

To begin, every calendar event must have some opening and closing tags, just like HTML. We will be placing our properties in the space between them. The only thing of interest here is PRODID, which acts like a title tag does. All you need to fill in is your company and event (between the // slashes), and if you want to change the language, use the two letter abbreviation to denote your locale of choice.

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Company//Event//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT

END:VEVENT
END:VCALENDAR
```

Event Naming

First and most importantly, we need to give the name of your event, as this will be what the item is known as within the visitors' calendar (and what will pop up when they receive notifications and reminders if scheduled). This is a required element (as is the start and end date). I'd also advise not making the name too long to avoid the title getting partially cut off or truncated in certain calendar apps.

```
SUMMARY:Event
```

Time Stamps

Next, we need to provide the start and end date of the event. These are important as it identifies if your event spans multiple days or if the event lasts only a single day. If it spans a single day, have it start on one day and end the next, as this indicates a midnight to midnight scenario. It's also worth noting that events are labelled in the format YYYY-MM-DD / HH-MM-SS (as shown below). You can also add an optional time stamp here of when the event.ics was created.

```
DTSTAMP:20210330T071500Z
DTSTART;VALUE=DATE:20211105
DTEND;VALUE=DATE:20211106
```

Frequency

An additional rule for the event is the frequency rule, as you might wish your event to reoccur. Options include daily, weekly, monthly, and yearly. You can tell the rule to occur a set number of times (the second example), and a specific day in a month (the third example).

```
RRULE:FREQ=YEARLY
RRULE:FREQ=MONTHLY;COUNT=6
RRULE:FREQ=YEARLY;BYMONTH=3;BYDAY=FR
```

Location

Another useful property that many calendar apps can take advantage of is location and geo. Aside from using `\n` to indicate a new line it's pretty straightforward, though adding Apple Maps support requires some geo co-ordinates which you should be able to get via Google Maps if you want to add some useful integration for physical events.

```
LOCATION:Lewes\nEngland
GEO:50.873132,0.010429
X-APPLE-STRUCTURED-LOCATION;VALUE=URI;X-TITLE=Lewes\
\nEngland:geo:50.873132,0.010429
```

Description

Something optional but useful for people who check their calendars regularly is the organizer and description. They are self-explanatory as the organizer identifies who is responsible for the event and the description is a great place to provide notes about what will occur.

```
ORGANIZER:Company
DESCRIPTION:Details
```

Website

Next, there is the optional URL which allows you to provide a link to your event's website, which can be really useful to quickly let visitors who booked an event or are awaiting something to happen to jump straight from the notification they get in a calendar to your event's website (or perhaps a promotion page related to the event).

```
URL;VALUE=URI:https://example.com
```

Alarm

Furthermore, there is an alarm or notification. If you have an event in a user's calendar, they will likely want to be reminded about it. You can set up a specific reminder of the event set to display on their screen at a timed interval of either minus x minutes (-PT0M) or days (-PT0D). This is naturally dependent on a user's device settings.

```
BEGIN:VALARM
TRIGGER:-PT60M
ACTION:DISPLAY
DESCRIPTION:Reminder
END:VALARM
```

Checksum

Lastly, we need to give this file a UID (unique ID). You could make up a random string of capital letters and numbers if you wanted, or use a password manager to do it for you; however, my method which works well for me and ensures a random string every time is to use an MD5 checksum for the event.ics file. Then add this into your final file.

```
UID:Code
```

If you don't know how to make an MD5 checksum? On Mac, you can open a terminal window at the file's location, type `md5 filename.ics`, and it will create one. On Windows, open a command prompt and type `CertUtil -hashfile <path to file> MD5`, and it will create one. And on Linux, open a terminal prompt at the files location and just type `md5sum <filename>.ics` to create the MD5 checksum you require.

Reference	To get an MD5 hash, see the following articles. https://onthefencedevelopment.com/2017/08/15/windows-10-builtin-md5-checksum-calculator/ (Windows), https://www.mjdtech.net/how-to-check-md5-checksum-in-os-x-terminal/ (Mac), https://www.geeksforgeeks.org/md5sum-linux-command/ (Linux)

Summary

And that’s all there is to it. Hosting event files allows a visitor to click, download, and import useful event information straight into their desktop / email client or smartphone. It’s a quick and easy way to assist visitors who have memory issues (so it’s great for accessibility) and it helps timed events retain users who otherwise fail to return.

Reference For more information about the iCalendar format and additional syntax details, check out the Wikipedia article at https://en.wikipedia.org/wiki/ICalendar

Chapter 11:

Make your project stand out with an icon

Generate the various images required for multi-platform support for browsers, banners, and icons.

favicon.ico

Is there anything more underestimated than a favicon? A tiny image that sits in the browser tab list, bookmarks menu or reading list that helps visitors identify your website or app more easily among the many windows or tabs they may have opened or have saved. It's great for enhancing the user-experience, but they can be difficult to put together thanks to the ever-changing standards that define how to create them over the years. We shall examine these more closely.

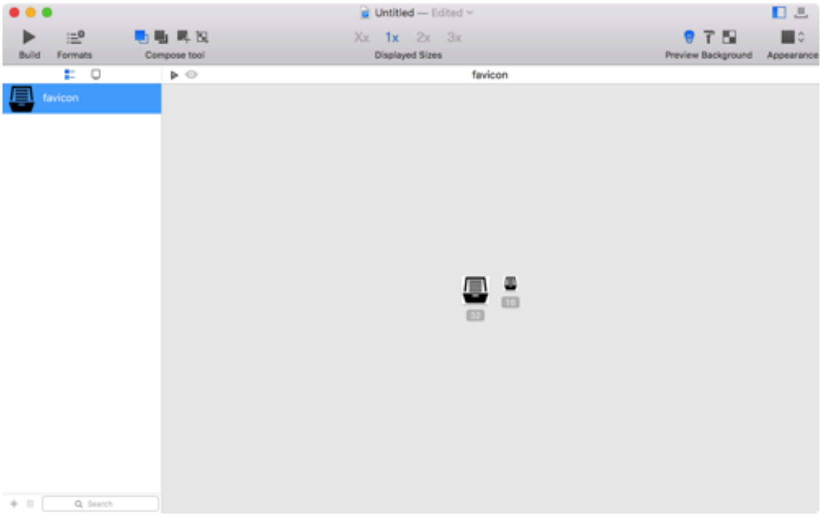
Reference For additional information about Favicons, Wikipedia has a general overview of the history and legacy of the format at https://en.wikipedia.org/wiki/Favicon

Before we cover the first format you'll need to include, let's discuss the ones you no longer need to add to your website. For example, you don't need a 32×32 or 16×16 png file anymore as one of your larger images or your SVG favicon will (down)scale well. You also don't need a Safari monochrome pinned SVG icon, as it's deprecated. Opera speed dial images aren't very widely adopted either, so no point having those. Now it's time to get to the stuff you will need.

First up is the favicon.ico image, and it's a bit of a controversial one, as it's a proprietary format created by Microsoft and was supported by Internet Explorer (but also works in all other desktop browsers). Truth be told, now IE is just about dead; better formats exist, and the only way to create an ICO file is using a special icon editor. However, favicon.ico has the unique quirk that browsers and search engines might seek the file automatically, so it's worth having.

Previously, with ICO files you had to support up to 256×256 images due to Window's deep integration with Internet Explorer, however, with the browser nearly dead and the later versions very much separated from the OS, when you create a favicon.ico file using an icon editor, just include the 16×16 and 32×32 image formats to make your icon more performant as it'll look good enough for most users.

If you need an editor to create favicon.ico files, there are plenty of editors out there. On Windows, there's IcoFX Portable (it's an old version of their paid product, but it's free and works very well on its own), on Mac and Linux there's no free alternative, but there's the web app RealFaviconGenerator though if you really want an app for your workflow for Mac, Icon Slate or Image2icon both work well.



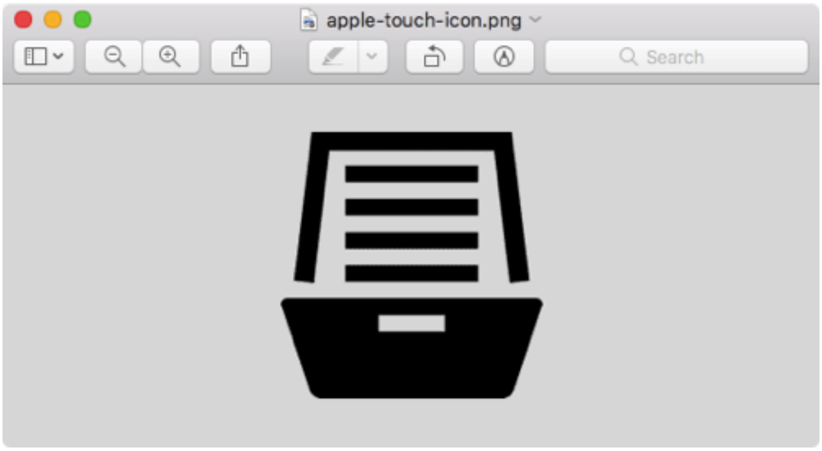
With this file, you should also include a reference in HTML files in the head section.

```
<link rel="icon" href="/favicon.ico" sizes="32x32">
```

Reference You can get the apps here: https://portableapps.com/apps/graphics_pictures/icofx_portable (IconFX), https://realfavicongenerator.net/ (RFG), https://www.kodlian.com/apps/icon-slate (Icon Slate), and https://img2icnsapp.com/ Image2Icon).

apple-touch-icon.png

Next we're going to examine the apple-touch-icon.png file, which was the second favicon file to be introduced to web browsers, and the first to be built for smartphones. This one, rather than serving your visitors in their tab bar and bookmarks, however, was built by Apple for iOS devices for progressive web applications (and any websites) that are added to the devices home-screen. It's a very useful icon to have for visitors wanting a more permanent shortcut to your site.



As with all the images in this chapter, you can build them using any editor you prefer (there's an abundance of choice on all platforms). Previously it was recommended to create various sizes of touch icons to account for retina and non-retina devices; but as Apple has mostly ended support for the older devices (as they are over 6 years old) you should just make one 180×180 file and let it downscale.

Another interesting point about touch icons is that, alike favicons, it's worth using them because Apple devices have come to seek them automatically, so make sure to include yours in the base directory. Furthermore, make sure it's called apple-touch-icon.png; otherwise server logs might display errors (as with favicon.ico) in devices that fail to find it. With this file, you should also include a reference in HTML files in the head section, the code for this is shown on the next page.

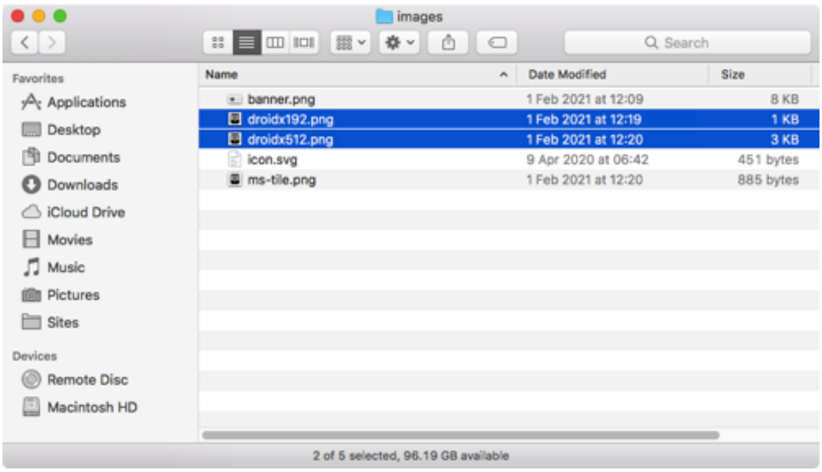
```
<link rel="apple-touch-icon" sizes="180x180" href="apple-touch-icon.png">
```

Reference For more details about the Apple Touch Image format, read https://developer.apple.com/library/archive/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html

x512.png and x192.png

Just as iOS has a favicon built for progressive web applications, so does Android (or more specifically Chrome) - actually two of them. You'll house them within a tiny file which we'll encounter later in the book called site.webmanifest (Chapter 27) and they will provide two scaled sizes of icon to the device's home-screen; the device will pick which best suits the situation and ignore the other image used.

While you can provide more than two images, there's only two that the Chrome team recommend, and as images are scaled as required, we may as well stick with that. You'll need to create both a 512×512 and 192×192 PNG image which can be located anywhere you prefer (as it'll be pointed too within the manifest file we create later on).



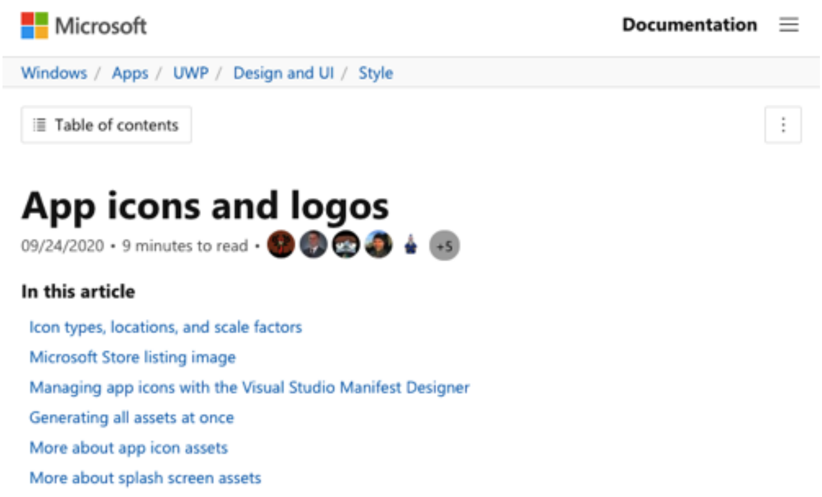
Name	Date Modified	Size
banner.png	1 Feb 2021 at 12:09	8 KB
droidx192.png	1 Feb 2021 at 12:19	1 KB
droidx512.png	1 Feb 2021 at 12:20	3 KB
icon.svg	9 Apr 2020 at 06:42	451 bytes
ms-tile.png	1 Feb 2021 at 12:20	885 bytes

2 of 5 selected, 96.19 GB available

Reference If you want to know more about the sizes you need to offer to Chrome, visit https://web.dev/add-manifest/

small.png, medium.png, wide.png and large.png

Just like with the ICO favicon that was built originally for Microsoft Internet Explorer, this favicon is built for the same browser. If you have any visitors that still use Internet Explorer version 8 to 11 then you will definitely want to include this image as you'll reference it within another tiny file which we'll encounter later on in the book (Chapter 27). However, if all your users are refined individuals using better, more up-to-date browsers, you can safely ignore this image file.



App icons and logos

09/24/2020 • 9 minutes to read • +5

In this article

Icon types, locations, and scale factors
Microsoft Store listing image
Managing app icons with the Visual Studio Manifest Designer
Generating all assets at once
More about app icon assets
More about splash screen assets

If you choose to create the ms-tile.png file, you'll need to make four separate images to cover the various tile sizes that both Windows and Internet Explorer supports. small.png should be 70×70, medium.png should be 150×150, wide.png should be 310×150, and large.png should be 310×310. You can put the image where you want on your server, as it'll be pointed to from within the BrowserConfig.xml file. As with all the images in this chapter, make sure you optimize (compress) them as best you can to improve the performance of the website or app.

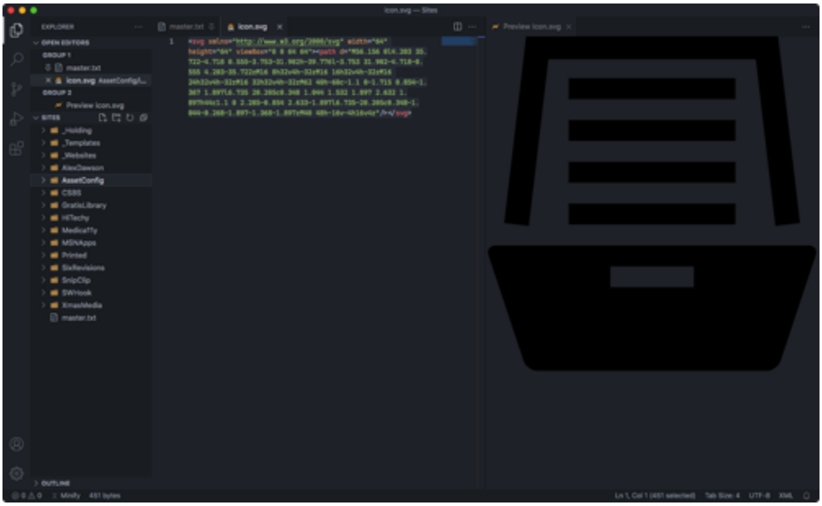
Reference For information about the tile sizes and splash screens, visit this detailed guide https://docs.microsoft.com/en-us/windows/uwp/design/style/app-icons-and-logos

icon.svg

Next is the newest in the bunch and the most unique as it uses a text-based format (SVG). It's easily the most performant of the favicons as you can compress it easily using Apache's GZIP or Brotli and with support in all modern web browsers, it's the icon that visitors will most likely see when browsing the web. It's the natural successor to the favicon.ico format, and you only need the one file size, as SVG has the wonderful ability to scale itself without losing any image quality.

Your SVG favicon can be placed anywhere on your web server, and among its special qualities are that you can use CSS (within the file) to give it support for dark mode, and using embedded CSS or JavaScript you can animate your SVG icon to provide little effects which could enhance the UX such as alerts, counters, and more. You can see why it's the best of all worlds for icon design, but as with the Apple touch favicon, you'll need to add a link reference in your HTML head.

```
<link rel="icon" type="image/svg+xml" href="images/icon.svg">
```

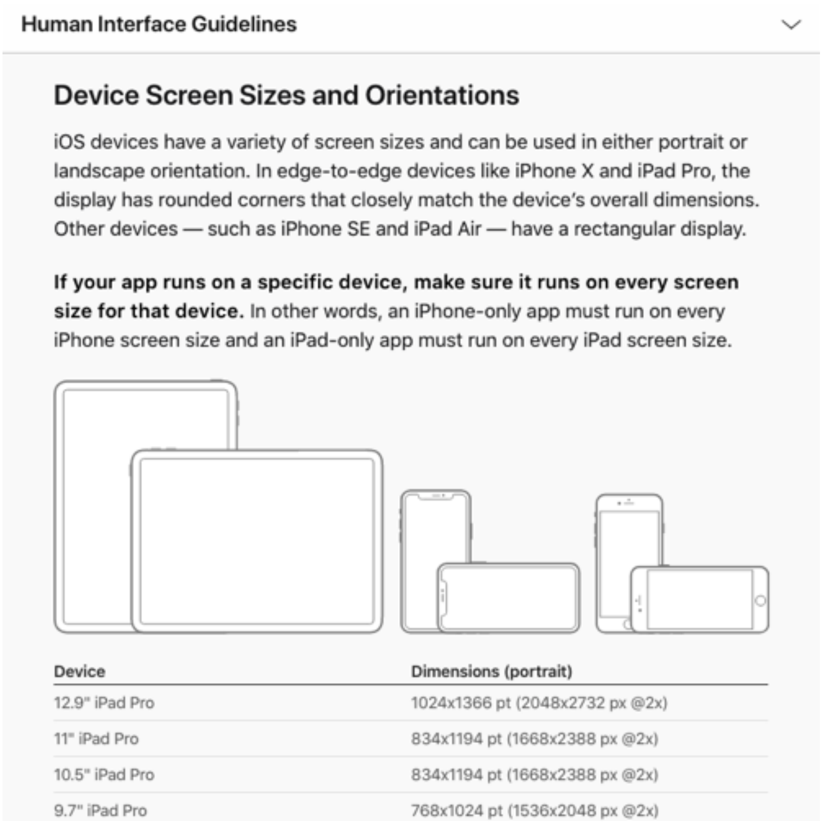


```
<svg xmlns="http://www.w3.org/2000/svg" width="64"
height="64" viewBox="0 0 64 64"><path d="M56.136 8l4.283 35.
722-6.718 0.555-3.753-31.982h-39.776l-3.753 31.982-6.718-0.
555 4.283-35.722zM16 8h32v4h-32zM16 16h32v4h-32zM16
24h32v4h-32zM16 32h32v4h-32zM62 40h-60c-1.1 0-1.715 0.894-1.
367 1.897l6.735 20.205c0.348 1.044 1.532 1.897 2.632 1.
897h46c1.1 0 2.283-0.854 2.633-1.897l6.735-20.205c0.348-1.
044-0.268-1.897-1.368-1.897zM40 48h-16v-4h16v4z"/></svg>
```

Reference For a comprehensive guide to all the cool things you can do with SVG icons, visit https://css-tricks.com/svg-favicons-and-all-the-fun-things-we-can-do-with-them/

apple-splash.png

Now we are officially done with the favicons, we can look at some of the other images that may prove useful for your web apps and sites. First up is the splash screen (startup-image) that iOS progressive web applications can add to introduce your web app to your visitors. It's a nice touch to make your product feel a bit more native even though it's hosted online. Think of it as a mini loading dialog box for your app.



Human Interface Guidelines

Device Screen Sizes and Orientations

iOS devices have a variety of screen sizes and can be used in either portrait or landscape orientation. In edge-to-edge devices like iPhone X and iPad Pro, the display has rounded corners that closely match the device's overall dimensions. Other devices — such as iPhone SE and iPad Air — have a rectangular display.

If your app runs on a specific device, make sure it runs on every screen size for that device. In other words, an iPhone-only app must run on every iPhone screen size and an iPad-only app must run on every iPad screen size.

Device	Dimensions (portrait)
12.9" iPad Pro	1024x1366 pt (2048x2732 px @2x)
11" iPad Pro	834x1194 pt (1668x2388 px @2x)
10.5" iPad Pro	834x1194 pt (1668x2388 px @2x)
9.7" iPad Pro	768x1024 pt (1536x2048 px @2x)

One of the main issues you'll come across when implementing this feature is the sheer number of images required to cover each iOS device. This is because unlike favicons, splash screen images don't scale, and you'll need one for each of the 31 supported devices and a link reference to it in your head to go with it, plus you'll also need another 31 to support rotation as portrait and landscape look visually different. This means 62 PNG files and link tags with media queries!

If you're up for this massive task (for what amounts to a small, useful branding exercise), I've included the five meta tags you include to ensure your app triggers the iOS progressive web application status. I've also included an example link tag with a media query that will target a specific iOS device (based on width and height) and orientation. Is the performance hit of including all the added code to pick the image worth the benefit? You are the ultimate judge on that.

```
<meta name="mobile-web-app-capable" content="yes">
<meta name="apple-touch-fullscreen" content="yes">
<meta name="apple-mobile-web-app-title" content="Name">
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style"
content="default">
<link rel="apple-touch-startup-image" media="screen and (device-
width: 320px) and (device-height: 568px) and (-webkit-device-
pixel-ratio: 2) and (orientation: landscape)" href="images/splash/
1136x640.png" >
```

Reference To help you work out what image sizes you require for a splash screen, use the Apple Interface Guidelines at https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/

banner.png

Last, we have an image that acts a splash screen, but doesn't require nearly as much effort as the Apple iOS image; however, it could be argued that it has a much wider reach and appeal. The banner.png is an image that exists to serve the OpenGraph protocol, which in turn will give visitors who share your site (and its pages) or app on social networks an image which represents your brand alongside the URL.

A minimum of 1080 pixels wide, Facebook recommends that you have an image of 1200×630; and as it will scale well on Twitter and other social networks, it's the resolution that I'd generally recommend going with to ensure widespread compatibility. To integrate the image into your site, you can place it anywhere on your server, but you need to link to it with a meta link (and alt tag) in your HTML head.

```
<meta property="og:image" content="images/banner.png">
<meta property="og:image:alt" content="Logo">
```

Reference If you would like a quick start workflow to help you generate favicons, this article may be of use https://evilmartians.com/chronicles/how-to-favicon-in-2021-six-files-that-fit-most-needs

Chapter 12:

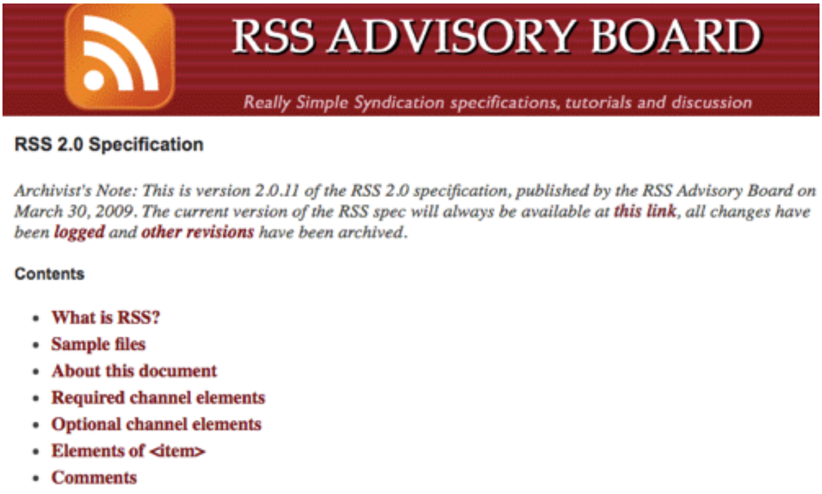
Offer syndicated content for non-browsers

Generate an RSS, Atom or JSON feed that's compatible with syndication readers or podcast clients like iTunes.

rss.xml

Welcome to the wonderful world of feed syndication. In a world of data hungry websites, advert infested blogs, magazines that spread your attention all over, clickbait news sites, and with social networks spiraling down the evil wormhole; the neutral force of this age-old medium has made it increasingly popular. Get the latest stories you're interested in without algorithms getting in the way. Offering this technology to your visitors should be high on your ethical agenda.

Syndication feeds provide the content and only the content. You won't find all the bandwidth hungry stuff that normally clutters up the average website. New content is pushed to you as separate articles you can read at your convenience (like emails in a client). You can even get notifications within feed readers for your favorite websites.


RSS ADVISORY BOARD
Really Simple Syndication specifications, tutorials and discussion

RSS 2.0 Specification

Archivist's Note: This is version 2.0.11 of the RSS 2.0 specification, published by the RSS Advisory Board on March 30, 2009. The current version of the RSS spec will always be available at **this link**, *all changes have been* **logged** *and* **other revisions** *have been archived.*

Contents

- What is RSS?
- Sample files
- About this document
- Required channel elements
- Optional channel elements
- Elements of <item>
- Comments

Creating a feed is pretty simple as all feeds (bar one) use the XML language, and it's down to personal preference as to which flavor you use to generate your code. We're going to start with RSS (Really Simple Syndication) which remains the most popular. You'll need to ensure that your code conforms to the XML specification (which is very strict) in order for feeds to work correctly, so code carefully).

RSS Creation

So let's get building our rss.xml file. First, we'll need an XML doctype with RSS and a channel opening and closing tags to encase our content feed. The below is a sample of the code you can use:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
    <channel>


    </channel>
</rss>
```

Next we'll start placing some website information in the channel tags, beginning with the required title, link, and description elements. I've included a table below the code to show all the potential elements you can include to describe your website or app in the channel element. I've also added an extra element used in the atom specification which is optional, but I find useful as it acts as a self-referential link for the feed (you'll see it in the code example).

```
<title>Title</title>
<link>https://example.com/</link>
<description>Description.</description>
<language>en-us</language>
<pubDate>Sun, 19 May 2021 15:21:36 GMT</pubDate>
<copyright>© Copyright Company 2021</copyright>
<atom:link rel="self" href="https://example.com/rss.xml"
type="application/rss+xml" />
```

Property	Required	Description
category	No	One or more categories.
<cloud domain="url.com" port="80" path="RPC2" registerProcedure="pingMe" protocol="soap"/>	No	Notify cloud service of updates to the channel or website.
copyright	No	Channel copyright notice.
description	Yes	A description of your channel.
docs	No	Docs about the format this is.
generator	No	App used to create this feed.
image	No	Contains url tag to image, title tag, and link tag to website.
language	No	Locale the feed is written in.
lastBuildDate	No	When the feed was updated.
link	Yes	The URL to your website.
managingEditor	No	Email of the feed editor.
pubDate	No	Date and time of publication.
skipDays	No	Which days can be skipped. Add day tag with day within.
skipHours	No	What hours can be skipped. Add hour tag with 0-23 within.
textInput	No	Must contain title, description, name, and link elements.
title	Yes	The name of your channel.

Property	Required	Description
ttl	No	Cache time before refresh.
webMaster	No	Email for technical support.

A Podcaster's Guide to RSS

A Podcaster's Guide to RSS

An RSS feed contains all the metadata of a podcast. This information governs what listeners will see about your podcast on Apple's services: from the the show art, to whether a podcast shows up in relevant searches, to episode titles and descriptions.

In order for a podcast to be listed on Apple's podcast directory and show up within the Apple Podcasts app, a corresponding RSS feed must be created, submitted on Podcasts Connect, and it must pass Apple's validation (see Podcaster Support for more information).

All metadata in an RSS feed is stored within element tags (here is a RSS Feed Sample for reference). Each tag is applicable to either show data (contained within the <channel> tag) or episode data (contained within an <item> tag). This guide will explain what data to include and how it should be structured.

Note: This page does not list all supported RSS tags, just the most important ones. If Apple

Reference	For a comprehensive guide to RSS, the specification is available at https://www.rssboard.org/rss-specification

Content

Next we need to add some news or content to our feed, and no surprise, it uses a pair of item properties within your channel tags. You'll need to add the item tags with some sub-properties for each news story (or item) you want to add. I've provided another table showing the elements you can use within item tags, and below that an example of a template you can use containing some tags.

Property	Required	Description
author	No	Email of the item's author.
category	No	One or more categories.
comments	No	URL to a comments page.
description	Yes	A description of your channel.
enclosure	No	Link to a podcast or videocast.
guid	No	A unique identifier string.
link	Yes	The URL to your website.
pubDate	No	Date and time of publication.
source	No	Channel the item came from.
title	Yes	The name of your channel.

```
<item>
    <title>Title</title>
    <author>Author</author>
    <pubDate>Sun, 19 May 2021 15:21:36 GMT</pubDate>
    <guid>https://example.com/article.html</guid>
    <link>https://example.com/article.html</link>
    <enclosure url="https://example.com/episode1.mp3"
    length="1069871" type="audio/mpeg"/>
    <description>Description.</description>
</item>
```

Integration

Finally, you’ll need to add a link to your completed RSS feed in the head of your HTML documents so that syndication clients, feed readers, and podcast clients can identify and use this useful format.

```
<link rel="alternate" type="application/rss+xml" href="feed.xml">
```

itunes.xml

Next up we're going to look at a subset of the RSS format which is built for the iTunes client - which is particularly popular with Mac and iOS users for podcast listening and well worth considering. Because it's a subset of RSS, I won't go as in-depth as we've covered the RSS language in detail above. What I will cover are the various differences and the extensions that iTunes provides that are worth looking at as they improve the experience for Apple podcast subscribers.

Reference iTunes uses an extension of the RSS language. For a list of the elements it supports check Apples specification at https://help.apple.com/itc/podcasts_connect/#/itcb54353390 and its element requirements at https://podcasters.apple.com/support/podcast-requirements

Feed Creation

To begin with, let's add the usual XML declaration, RSS element and channel tags. One thing to note is that the iTunes.xml file contains two more attributes in the RSS element. One for RSS modules and one for the iTunes extensions. It doesn't affect you, it's just worth mentioning.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
xmlns:content="http://purl.org/rss/1.0/modules/content/">
    <channel>

    </channel>
</rss>
```

Next we're going to add the usual website information in the header. This is where things get interesting because iTunes adds some extra tags that you can (and should) include to improve how your feed appears in the iTunes store for podcast users who use Apple devices on Mac or iOS. I've included an example, plus a table of supported elements in iTunes feeds. Some are from the RSS specification.

```
<title>Title</title>
<link>https://example.com/</link>
<description>Description.</description>
<language>en-us</language>
<pubDate>Wed, 17 Mar 2021 19:30:00 GMT</pubDate>
<copyright>© Copyright Company 2021</copyright>
<itunes:author>Author</itunes:author>
<itunes:explicit>false</itunes:explicit>
<itunes:image href="https://example.com/artwork.png"/>
<itunes:owner>
    <itunes:name>Owner</itunes:name>
    <itunes:email>email@example.com</itunes:email>
</itunes:owner>
<itunes:block>no</itunes:block>
<itunes:category text="Category"/>
```

Property	Required	Description
copyright	No	Channel copyright notice.
description	Yes	A description of your channel.
itunes:author	No	Who created the show.
itunes:block	No	Yes, to hide the podcast.
itunes:category	Yes	Which group does the show fit.
itunes:complete	No	Yes, value for no more updates.
itunes:email	No	Email the channel owner.
itunes:explicit	Yes	True or False (Adult content).
itunes:image	Yes	The artwork for your show.
itunes:name	No	Who owns the channel.
<itunes:new-feed-url>https://example.com/example.rss</itunes:new-feed-url>	No	Change the feed URL.
itunes:title	No	Concise iTunes podcast title.
itunes:type	No	Episodic or Serial (numbered).
language	Yes	Language the feed is written in.
link	No	The URL to your website.
pubDate	No	Date and time of publication.
title	Yes	The name of your channel.

Content

Next we’re going to add an episode to our podcast feed, as with an RSS feed, we need an item link for every episode in the season (or series), and there are a few elements we can enclose within it. Below I’ve provided an example of some code you could use, plus I’ve provided a table of the elements iTunes accepts in podcasts.

```
<item>
    <title>Title</title>
    <pubDate>Wed, 17 Mar 2021 19:30:00 GMT</pubDate>
    <guid>https://example.com/article.html</guid>
    <link>https://example.com/article.html</link>
    <description>Description.</description>
    <enclosure url="https://example.com/episode1.mp3"
    length="13475901" type="audio/mp3"/>
    <itunes:title>Title</itunes:title>
    <itunes:episode>1</itunes:episode>
    <itunes:author>Author</itunes:author>
    <itunes:explicit>false</itunes:explicit>
    <itunes:duration>1671</itunes:duration>
    <itunes:image href="https://example.com/artwork.png"/>
</item>
```

Property	Required	Description
description	No	A description of your channel.
enclosure	Yes	Link to a podcast or videocast.
guid	No	A unique identifier string.
itunes:block	No	Yes, to hide the episode.
itunes:duration	No	The audio duration in seconds.
itunes:episode	No	Podcast episode number.
itunes:episodeType	No	Full, Trailer, or Bonus media.
itunes:explicit	No	True or False (Adult content).
itunes:image	No	1400×1400 PNG or JPG image.
itunes:season	No	Podcast season number.
itunes:title	No	Concise iTunes podcast title.
link	No	The URL to your website.
pubDate	No	Date and time of publication.
title	Yes	The name of your channel.

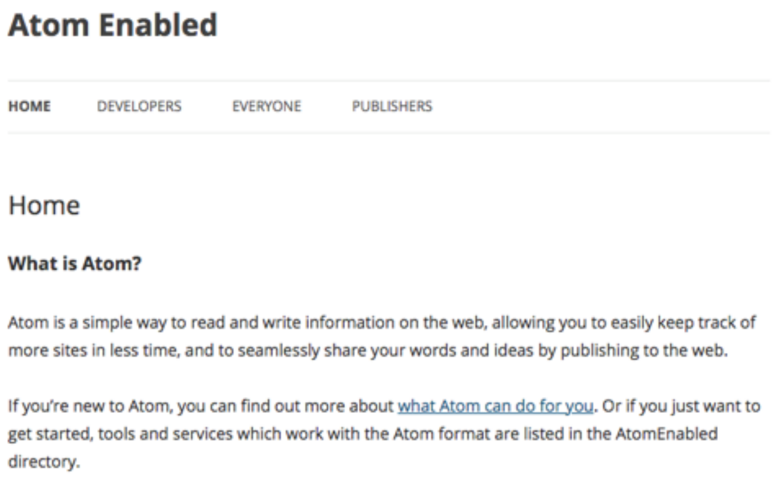
Integration

And no RSS feed could be complete without the link in the head of your HTML files, which allows iTunes to find a podcast based on the URL you paste into your client. Below is the source code you require.

```
<link rel="alternate" type="application/rss+xml" href="feed.xml">
```

atom.xml

Created as an alternative to the popular RSS format, this competing way of syndicating content never really gained the same level of popularity. But as it's still in use on the web, and some people may prefer to use it over the RSS format, I'm going to cover it here.



Home

What is Atom?

Atom is a simple way to read and write information on the web, allowing you to easily keep track of more sites in less time, and to seamlessly share your words and ideas by publishing to the web.

If you're new to Atom, you can find out more about what Atom can do for you. Or if you just want to get started, tools and services which work with the Atom format are listed in the AtomEnabled directory.

Reference For a comprehensive guide to Atom, I recommend checking out the syntax specification at http://www.atomenabled.org/developers/syndication/ and https://tools.ietf.org/html/rfc4287

To create an atom.xml document as with the RSS format, you begin by building an XML file (that can be stored anywhere you choose), and you add an XML doctype reference with a pair of feed tags which will contain our content. A code sample is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

</feed>
```

Between the feed tags, you add some information about the site or podcast you want to syndicate. Below are a list of the tags that Atom supports (many are the same or variations of RSS) in a table. You'll also see a code example that you can adapt as you want for your file.

```
<title>Title</title>
<id>https://example.com/</id>
<subtitle>Description.</subtitle>
<updated>2021-06-13T18:30:02Z</updated>
<rights>© Copyright Company 2021</rights>
<link rel="self" href="https://example.com/atom.xml"
type="application/atom+xml" />
```

Property	Required	Description
author	No	Contains sub elements: name, email & URI of the feed author.
category	No	One or more categories.
contributor	No	One or more name elements.
<generator uri="blog.php" version="1.0">Example Toolkit</generator>	No	App used to build the feed.
icon	No	Square image of the feed.
id	Yes	The URL to your website.
logo	No	The artwork for your feed.
rights	No	Channel copyright notice.
subtitle	No	Description of the channel.
title	Yes	The name of your channel.
updated	Yes	When the feed was updated.

Content

Next we need to add some content to the feed in the form of entry tags, as per RSS it’s one entry tag per item. You can add numerous elements within this tag as well, below is a boilerplate plus a table of tags for you. You'll need to include an id, title and updated element.

```
<entry>
    <title>Title</title>
    <author>
        <name>Name<name>
        <email>email@example.com</email>
    </author>
    <updated>2021-06-13T18:30:02Z</updated>
    <id>https://example.com/article.html</id>
    <link rel="alternate" href="https://example.com/article.html" />
    <content type="xhtml" xml:lang="en"><div xmlns="http://
    www.w3.org/1999/xhtml"><p>Description.</p></div></
    content>
</entry>
```

Property	Required	Description
author	No	Contains sub elements: name, email & URI of the feed author.
category	No	One or more categories.
content	No	See the above HTML example.
contributor	No	1400×1400 PNG or JPG image.
id	Yes	The URL to your website.
link	No	A related link to the channel.
published	No	Date and time of publication.
rights	No	Channel copyright notice.
source	No	Contains elements id, title, updated and rights tags.
summary	No	Summary of the content.
title	Yes	The name of your channel.
updated	Yes	When the feed was updated.

Integration

Finally, we need to add a link into the head of our HTML document to finish, as is required of any syndication format to let it be recognized.

```
<link rel="alternate" type="application/atom+xml" href="feed.xml">
```

feed.json

This is the last of the syndication formats you may come across, and as it's the newest one. It's the one with the most compatibility issues, as most feed readers focus their support on XML rather than this format, which is the cleaner and easier to maintain JSON format. It is gaining adoption rapidly, so it's one to keep your eye on, and maybe offer as an alternative. With this in mind, lets build a feed.json file.

Reference Creating a JSON file may be straight forward, but if you have any questions not covered in this book, check the specification at https://jsonfeed.org/version/1.1

JSON Creation

Unlike XML files, no declarations are required, JSON just uses opening and closing brackets. Within these, you'll want to include some basic information about your feed, I've included a code sample below (leaving a space for the items). As usual, I've included a table for the properties supported. And there are some required elements.

```
{
    "version": "https://jsonfeed.org/version/1.1",
    "title": "Title",
    "home_page_url": "https://example.com/",
    "feed_url": "https://example.com/feed.json",
    "items": [

    ]
}
```

Property	Required	Description
authors	No	Contains name, url and avatar. Array syntax shown like items.
description	No	A description of your channel.
expired	No	Yes, or No if feed is finished.
favicon	No	Min, 64×64 square image.
feed_url	Yes	The URL of this JSON feed.
home_page_url	Yes	The URL to your website.
hubs	No	Contains type and URL array to subscribe for notifications.
icon	No	512×512 image of your channel.
link	No	A related link to the channel.
next_url	No	The next feed in sequence.
rights	No	Channel copyright notice.
title	Yes	The name of your channel.
user_comment	No	The purpose of the feed.
version	Yes	URL of the JSON specification.

Content

As you may notice, the JSON feed supports fewer properties, which may or may not be a positive for your productivity. Either way, it's time to add some items, remembering that each one will be enclosed within open and close brackets, and I've provided a boilerplate with a few example tags. Below that is a table of required and optional tags.

```
{
    "title": "Title",
    "id": "1",
    "url": "https://example.com/article",
    "content_html": "<p>Description.</p>"
}
```

{"url": "jsonfeed.org"}

JSON Feed Version 1.1

by Brent Simmons and Manton Reece

The JSON Feed format is a pragmatic syndication format, like RSS and Atom, but with one big difference: it's JSON instead of XML.

Property	Required	Description
attachments	No	Array containing URL, title, and mime_type. Used for podcasts.
authors	No	Contains name, url and avatar. Array syntax shown like items.
content_html	Yes	If not text, use this for content.
content_text	Yes	If not html, use this for content.
date_modified	No	When the item was modified.
date_published	No	When the item was published.
external_url	No	Additional URL for the item.
id	Yes	Unique identifier for the item.
image	No	Image to accompany the item.
language	No	The language of the content.
summary	No	Brief description of the content.
tags	No	Array of tags describing data.
title	No	The name of your item.
url	Yes	URL for the article or podcast.

Integration

Finally, to end this lengthy chapter, you need to add a reference to the head of your HTML element in order for feed readers to identify it.

```
<link rel="alternate" type="application/feed+json" href="feed.json">
```

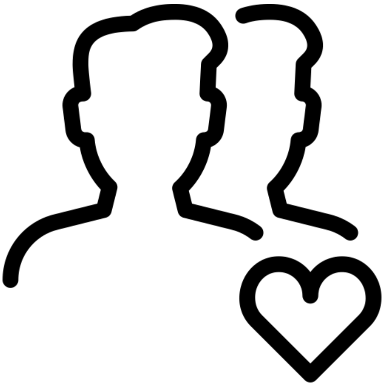
Just as a final rounding off, whichever format you choose (whether RSS, iTunes, Atom, or JSON), run your code through a feed validator to ensure that your syntax is well-formed. This is because XML is prone to breaking easily if a single mistake is made. Additionally, be sure to try your feed in more than one feed reader (desktop, mobile, iTunes, and web) to check it looks (visually) and behaves as expected. Compatibility testing doesn't just stop at your HTML, CSS, and JS.

Reference The W3C has a feed validator that will work on any XML document at https://validator.w3.org/feed/

Chapter 13:

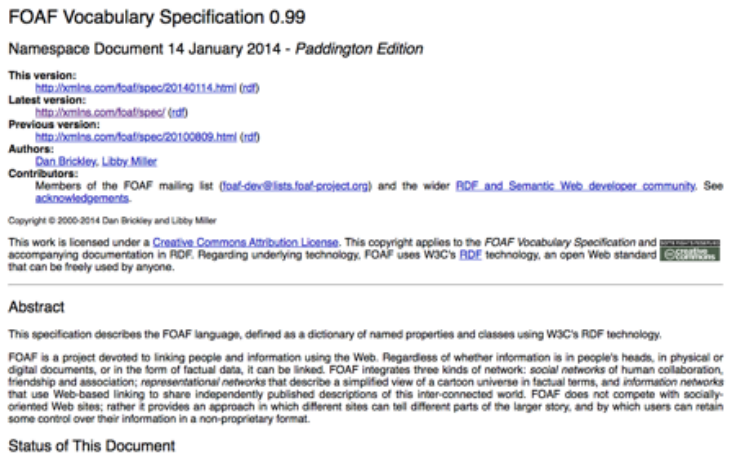
If you're friendly, network with other sites

If you're a member of the FOAF scheme, use this tool to build a profile of your identity for SEO purposes.



foaf.rdf

Described using a more complex version of the XML specification (the RDF schema), it's no wonder that this tiny file didn't gain the levels of adoption that could have matched it's potential. It's a shame as in this age of social networking and the semantic web (where all we post has a meaning), is this way of telling users who you are so, they can in turn link to you using a webring, literally as a friend-of-a-friend.



FOAF Vocabulary Specification 0.99

Namespace Document 14 January 2014 - *Paddington Edition*

This version:
http://xmlns.com/foaf/spec/20140114.html (rdf)
Latest version:
http://xmlns.com/foaf/spec/ (rdf)
Previous version:
http://xmlns.com/foaf/spec/20100809.html (rdf)
Authors:
Dan Brickley, Libby Miller
Contributors:
Members of the FOAF mailing list (foaf-dev@lists.foaf-project.org) and the wider RDF and Semantic Web developer community. See acknowledgements.

Copyright © 2000-2014 Dan Brickley and Libby Miller

This work is licensed under a Creative Commons Attribution License. This copyright applies to the *FOAF Vocabulary Specification* and accompanying documentation in RDF. Regarding underlying technology, FOAF uses W3C's RDF technology, an open Web standard that can be freely used by anyone.

Abstract

This specification describes the FOAF language, defined as a dictionary of named properties and classes using W3C's RDF technology.

FOAF is a project devoted to linking people and information using the Web. Regardless of whether information is in people's heads, in physical or digital documents, or in the form of factual data, it can be linked. FOAF integrates three kinds of network: *social networks* of human collaboration, friendship and association; *representational networks* that describe a simplified view of a cartoon universe in factual terms, and *information networks* that use Web-based linking to share independently published descriptions of this inter-connected world. FOAF does not compete with socially-oriented Web sites; rather it provides an approach in which different sites can tell different parts of the larger story, and by which users can retain some control over their information in a non-proprietary format.

Status of This Document

While the specification hasn't seen much action recently, it hasn't been superseded by anything better. As the semantic web is still important for SEO and for accessibility and for social visibility, it's still worth adding this micro-file to your website. The great thing is it doesn't really take much effort, you just need to wrap your head around the different style of syntax, which we shall describe below.

Reference For a more comprehensive guide to Friend of a Friend, check the specification at http://xmlns.com/foaf/spec/

Networking

First, you'll need to create your foaf.rdf file and put it somewhere on your server. Within this file, you'll have to include the below replacing the word Here with a value of Agent, Person, Organization, Group or Project, depending on whether your site is for you, your business or your application. You'll also want to replace #Name with the name of your website or app (remembering to include the # character).

```
<foaf:Here rdf:about="#Name" xmlns:foaf="http://xmlns.com/foaf/
0.1/">

</foaf:Here>
```

Next, you'll want to include the friend-of-a-friend property and value pairs. There are a number of them, so I've included a table of the various ones you can use below. Note that I've removed a few from the specification that are deprecated due to the services they work with no longer being available. You should include enough properties to sum up you or your business's connection to the website, don't include every tag in this file. Furthermore, remember to precede each tag with a foaf: reference in the tag, for examples, see the boilerplate.

Property	Description
account	A service run by you or your business.
accountName	The name associated with the service.
accountServiceHomepage	The URL associated with the service.
age	The age of you or your business / app.
based_near	The location you are based closest too.

Property	Description
currentProject	What you're working on (See made).
depiction	Relationship between you and an item.
familyName	Your surname (or use name instead).
givenName	Your first name (or use name instead).
homepage	You or your businesses' homepage.
img	An image that represents the profile.
interest	A webpage about your hobbies.
jabberID	A Jabber ID for messenger users.
knows	Someone known to your business.
logo	The logo for you or your business.
made	Project made by you or your business.
mbox	The email address used for contact.
mbox_sha1sum	SHA1 checksum for contact email.
member	A group that you are a member of.
name	Your name or your product's name.
nick	A nickname or handle you use.
openid	The openID address for authentication.
pastProject	A project you worked on (See made).
primaryTopic	The main category or topic (See topic).
publications	Link to a list of publications by you.
schoolHomepage	URL to the school you have attended.

Property	Description
sha1	A SHA1 checksum for a purpose.
thumbnail	A profile thumbnail image (See img).
tipjar	How to donate or pay on your site.
title	Mr, Mrs, Miss, Ms, Dr, Prof, Etc.
topic	The category or topic of this website.
topic_interest	Main category of interest (See topic).
weblog	Link to the blog for your website.
workInfoHomepage	Link to your portfolio (See homepage).
workplaceHomepage	A link to your businesses' website.

Next, you'll want to include a link to the file in the head of your HTML file so that social media and search engines that support the semantic web and the friend-of-a-friend scheme can pick up and utilize the file.

```
<link rel="author" type="application/rdf+xml" href="foaf.rdf">
```

Boilerplate

Finally, I’ve included an example of a complete foaf.rdf document below, which you could adapt if you wanted to help you in your projects. URL's and image references have no closing tag, but have an rdf:resource attribute. Much alike the dublin.rdf file we discussed in an earlier chapter, it’s a great tool to promote your connection to your work. While social networks and search engines will be the main method of pushing your brand, this tiny file might be a low-impact method of helping users connect to you through your digital work.

```
<foaf:Organization rdf:about="#name" xmlns:foaf="http://
xmlns.com/foaf/0.1/">
<foaf:name>Name</foaf:name>
<foaf:homepage rdf:resource="https://example.com/">
<foaf:img rdf:resource="apple-touch-icon.png">
</foaf:Organization>
```

Chapter 14:

Show your physical location for retail stores

If you rely on RDF metadata, create an indexable link to your location, with full Google Earth KML integration.

geo.rdf

Existing on the web is central to our everyday lives, yet people often still underestimate the value that having a physical location can bring. It gives you a greater sense of authenticity, having roots in the real world, it offers another means of individuals getting in touch, and it provides location awareness when people reach out for technical support. Knowing your timezone means they may not bother you if they know you're asleep is a prime benefit of location awareness.

As this information is useful to offer for both individuals and businesses, let's next introduce you to the tiny but useful geo.rdf file. It's not too well known as like dublin.rdf and foaf.rdf it uses the RDF schema which is very strict (using XML) and more complex than formats like JSON and thus less popular. As Geo uses a hybrid of both DCMI and FOAF to provide its markup, it's become deprecated in favor of Microdata (Schema) which can be embedded in a page.

Reference If you need a reminder of either specifications that Geo uses, you can do so at https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ (DCMI) and http://xmlns.com/foaf/spec/ (FOAF)

RDF Creation

With all of this being said, Google can take advantage of RDF files and the data is still useful, especially as using RDF, you can avoid lots of repetitive markup cluttering your HTML (making maintenance easier if the location data should be on all your pages). To take advantage of this file, create a geo.rdf file somewhere on your server and include an XML doctype, RDF tags, and a rdf:Description element which should contain an rdf:about attribute with your base URL as the value.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:geo="http://
www.w3.org/2003/01/geo/wgs84_pos#" xmlns:rdf="http://
www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="https://example.com">

    </rdf:Description>
</rdf:RDF>
```

Between the rdf:Description elements, we need to put a couple of tags which describe where the website is located. You'll want to describe the region in which your office or physical address is, and a description of how you would label your premises for people who choose to visit. With these, I often reflect my HTML head values.

```
<dc:title>Location</dc:title>
<dc:description>Description</dc:description>
```

Location Data

Next we should include your location. These are the latitude and longitude co-ordinates that are used by tools like Google Maps to identify where you are physically located when they pick up this file.

```
<foaf:topic rdf:parseType="Resource">
    <geo:lat>00.000000</geo:lat>
    <geo:long>-0.00000</geo:long>
</foaf:topic>
```

Finally, we need to include a reference to your file in the head of your HTML document so that search engines can make use of the RDF file.

```
<link rel="author" type="application/rdf+xml" href="geo.rdf">
```

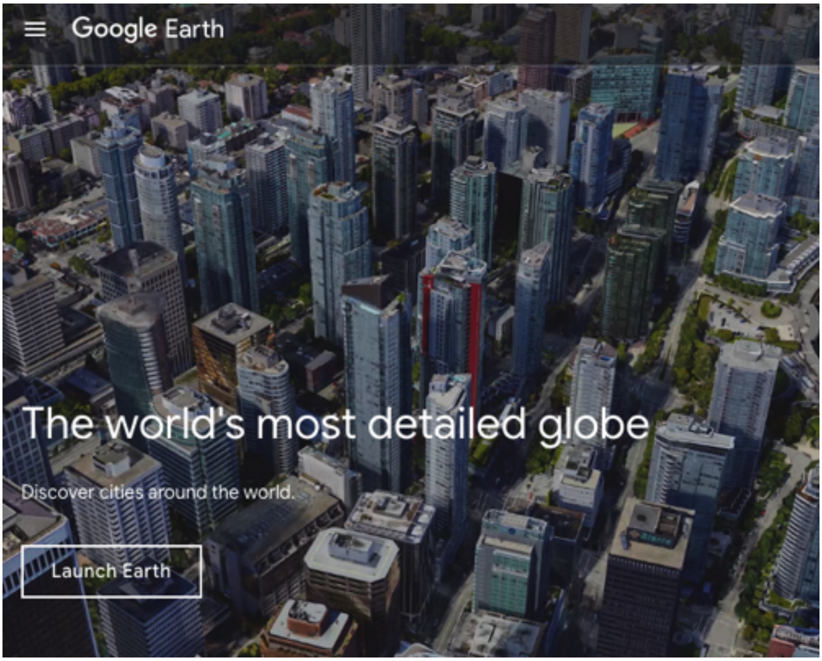
Reference To find your geodata co-ordinates, use the tag creator at https://www.geo-tag.de/generator/en.html

geo.kml

This isn't the end of our physical location journey because another format exists which can provide (with the same co-ordinates) a more augmented reality experience. If you've ever used the Google Earth application, you'll know it's a pretty amazing tool, as it allows you to use a range of satellite photography to zoom down onto a location with some accuracy. Well, we can add support for this directly within our website to help visitors find your premises literally on the planet.

Google Earth supports a custom subset of the XML language called KML (Keyhole Markup Language). While there's a fair amount to the specification, for this tutorial I'm just going to give you enough to build a file that Google can use to allow your users to both import and Zoom straight to your physical location. It might not be the most practical tool, but it's a fun bonus feature for your site or app.

Reference For a more comprehensive guide to the KML language, read this introduction which links to a syntax reference https://developers.google.com/kml/documentation/


Google Earth
The world's most detailed globe
Discover cities around the world.
Launch Earth

KML Creation

To begin, create a geo.kml file and place it alongside your geo.rdf file (after all, you may as well support both if you're going to handle the Semantic Web). Within the KML file you'll want an XML declaration alongside opening and closing elements for KML and Document.

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
    <Document>

    </Document>
</kml>
```

Within the Document element (note this is case-sensitive) you'll want to include some elements to describe the location (as we did in the geo.rdf file). One named region, and a description of the location.

```
<name>Location</name>
<description>Description</description>
```

Location Data

Next we need to-do a couple of things. First, we require an element to stick a pin in your location so when Google Earth zooms in, the visitor can find it easily. Secondly, we need to add the co-ordinates. Note that KML requires them to be entered in two places, the LookAt element and via the Point element, for Google Earth to successfully navigate to the location. The other values you can safely ignore.

```
<Style id="pin"><IconStyle><Icon><href>http://maps.google.com/
mapfiles/kml/pushpin/ylw-pushpin.png</href></Icon></
IconStyle></Style>
<Placemark>
    <LookAt>
        <longitude>-0.00000</longitude>
        <latitude>00.000000</latitude>
        <altitude>0</altitude>
        <tilt>0</tilt>
        <range>5500</range>
    </LookAt>
    <styleUrl>#pin</styleUrl>
    <Point><coordinates>-0.00000,00.000000,0</coordinates></
    Point>
</Placemark>
```

Integration

Finally, now that our file has been put together you can try it out in the Google Earth app to make sure it works, and if it does, you can put the below line of HTML in the head of your pages to let it be used. You might also want to offer a direct download link to the file as it's a non-standard format that is only utilized by the Google Earth app.

```
<link rel="author" type="application/vnd.google-earth.kml+xml"
href="geo.kml">
```

Reference You can try your KML file in the Google Earth app by downloading it at https://www.google.com/earth/

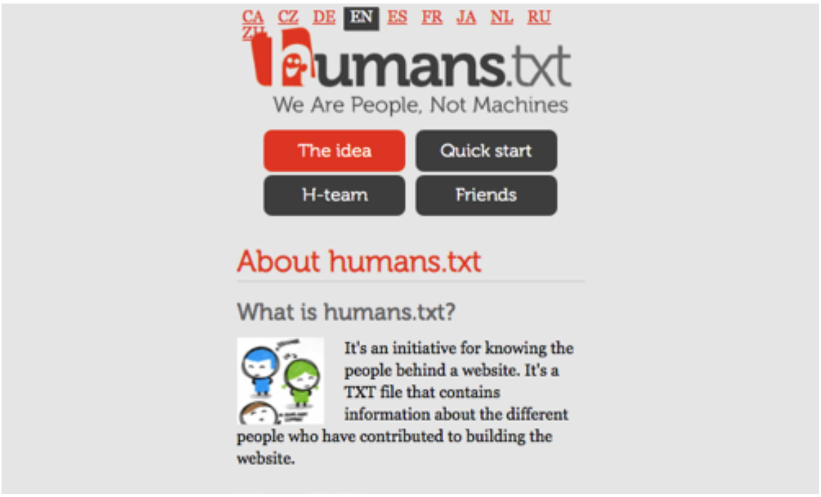
Chapter 15:

Credit those who made the project possible

Give credit where it's due with the self-contained, non-impactful accreditation text file format.

humans.txt

Every website or app deserves credit, and often multiple people are involved in the creative process. From the developers and designers who work behind the scenes to the images and typefaces you buy that require attribution. One of the biggest questions involved in attribution is how do you give fair credit to people or businesses on the web without it becoming overly spammy or cluttering up critical pages of your content? This is where the humans.txt file appears.



About humans.txt

What is humans.txt?

It's an initiative for knowing the people behind a website. It's a TXT file that contains information about the different people who have contributed to building the website.

Using a similar form of syntax to the robots.txt file (which we'll cover later in the book in Chapter 25), humans.txt is a plaintext method of providing thanks, credit, and your technology stack details in a centralized, human-readable format without having numerous anchor links seeping SEO karma where you don't want it too. By having all of your credits within this file, you can ensure that you offer a technical peek under the hood of your site, but in a very low-impact manner.

Overview

Sounds good? It’s well-supported as many sites adopt this standard, and it allows you to give credits to images, libraries and such in one place without scattering credit links all over your sites. To start, you’ll want to create a humans.txt file and put it in the base directory of your website (as this is where it's assumed it’ll be by visitors). Within the file, you will want to include some sections (using comments).

```
/* ---------------------- */
/* Product Website Credits */
/* ---------------------- */

/* TEAM */

/* THANKS */

/* SITE */
```

Team

You can include custom sections and properties as humans.txt doesn’t have a strict set of standards, but the collection I’ve built up has been adopted by a variety of sites, so they are probably considered to be best practices. With this in mind, you should replace "Product" with the name of your site or app, and in the TEAM section, you’ll want to include the following properties for each individual working on the website. They are all optional, but recommended, and you could choose to create additional custom property / value pairs.

```
Title: Name
Position: Job Title
Site: https://example.com
Twitter: @TwitterID
Location: Region, Country
Contact: email@example.com
```

Thanks

Next, you'll want to fill in the THANKS section by providing credit to those who helped in the creation of your website or app. Whether this was for images or typefaces you used, or just the people who tested your product, they all deserve to be given a mention. Feel free to include as many individuals (or groups) as you feel is necessary.

```
Title: Name
Twitter: @TwitterID
Site: https://example.com
License: https://example.com/license.txt

/* You for visiting this website! */
```

Website

Finally, we need to fill in the SITE section, which contains an array of useful under-the-hood website details for anyone who's interested. Why should you include this? It's useful for other web developers; plus if a visitor encounters an issue, they might be able to use the information you offer to identify the cause of the error and provide a solution when they contact you for support (say, if a rogue script is at work). I've provided a list of the properties that you can include.

```
Language: English (US)
Standards: TXT, HTML, CSS, JS
Components: None
Software: None
Services: Font (Typeface), Name (Icons), Hosting, Domains
Testing: Chrome (Mac/PC), Firefox (Mac/PC), Safari (Mac/iOS), IE
(11+)
Updated: 2021/06/12 10:12
```

If you support multiple languages, list them. If you utilize more than the included language standards (say SVG), add them. If you use Components (code libraries like jQuery, Vue, or Angular), list them. If you built the site using apps (Word processors, image editors, IDEs, and such), include them. Also provide information on any typefaces, icon packs, hosting, and domains you use. Next, include the date the file was last updated. And any devices you test the site upon.

Integration

Finally, you'll need to include a link to the file in the head of your HTML file. It's not so much for visitors as they'll automatically look at the root of your site, but it's for any search tools that index human.txt files.

```
<link rel="author" type="text/plain" href="humans.txt">
```

Reference For a comprehensive guide to the humans.txt file, you can read the specification at http://humanstxt.org/

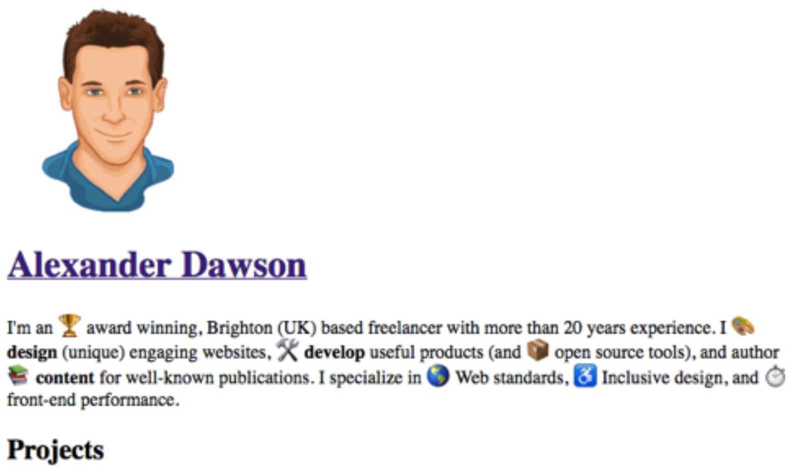
Chapter 16:

Get a solid backbone for your web design

Every project needs a starting point, this is mine. Generate an HTML5 boilerplate that'll get your site started.

index.html

When beginning a web project, you'll usually start with an HTML file. As every project is different and will have different requirements, it's often difficult to determine exactly what each project will require as a starting point to be most productive. Some people like to work from scratch in a blank IDE coding their HTML manually, while others like to work with a large complete set of pre-built boilerplates such as the open source Bootstrap or the H5BP (though the choice is endless).



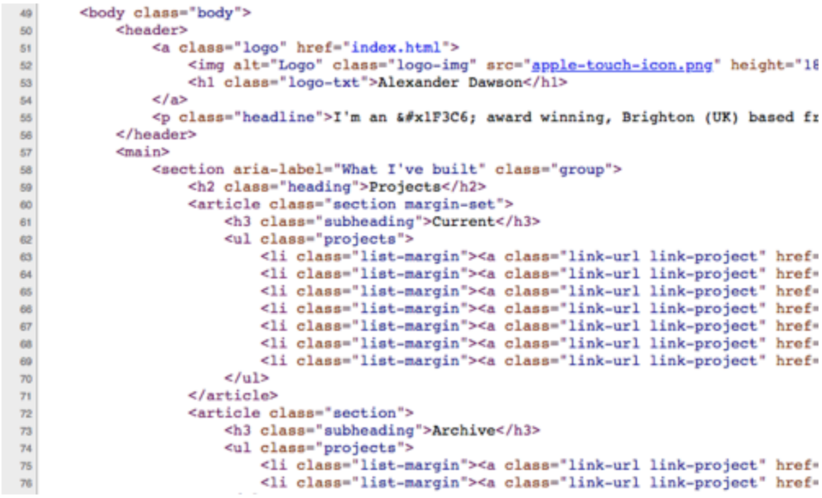
Alexander Dawson

I'm an 🏆 award winning, Brighton (UK) based freelancer with more than 20 years experience. I 🎨 **design** (unique) engaging websites, 🛠 **develop** useful products (and 📦 open source tools), and author 📚 **content** for well-known publications. I specialize in 🌎 Web standards, ♿ Inclusive design, and ⏱ front-end performance.

Projects

Reference If you do want to use a pre-existing boilerplate, your best options to use are http://www.initializr.com/ (Initializr), https://html5boilerplate.com/ (H5BP), or https://getbootstrap.com/ (Bootstrap).

The problem with using any method aside from starting from scratch, however, is they typically add a lot of unnecessary bulk and may not be up-to-date. While in an earlier chapter we covered all the potential elements you could add in the head of your HTML (and there's quite a few), this chapter is focused on building a lean starter document with the minimal required markup (plus a few of the latest best practice recommendations for HTML) so you can customize it as you require.



```
    <body class="body">
        <header>
            <a class="logo" href="index.html">
                <img alt="Logo" class="logo-img" src="apple-touch-icon.png" height="18
                <h1 class="logo-txt">Alexander Dawson</h1>
            </a>
            <p class="headline">I'm an &#x1F3C6; award winning, Brighton (UK) based fr
        </header>
        <main>
            <section aria-label="What I've built" class="group">
                <h2 class="heading">Projects</h2>
                <article class="section margin-set">
                    <h3 class="subheading">Current</h3>
                    <ul class="projects">
                        <li class="list-margin"><a class="link-url link-project" href=
                        <li class="list-margin"><a class="link-url link-project" href=
                        <li class="list-margin"><a class="link-url link-project" href=
                        <li class="list-margin"><a class="link-url link-project" href=
                        <li class="list-margin"><a class="link-url link-project" href=
                        <li class="list-margin"><a class="link-url link-project" href=
                        <li class="list-margin"><a class="link-url link-project" href=
                    </ul>
                </article>
                <article class="section">
                    <h3 class="subheading">Archive</h3>
                    <ul class="projects">
                        <li class="list-margin"><a class="link-url link-project" href=
                        <li class="list-margin"><a class="link-url link-project" href=
```

Reference HTML5 is a straight forward but changing language. If you want to learn more about it check the specification at https://html.spec.whatwg.org/multipage/ or browse https://developer.mozilla.org/en-US/docs/Web/HTML

Base Code

So let’s begin by creating an index.html file in your base directory. You’ll need several things within it, including the HTML5 doctype, an HTML5 element (with a lang attribute indicating the locale you’ll be using for the document), plus a head and body tag for the "thinking code" and "content code" of your document. Next is an example.

```
<!DOCTYPE html>
<html lang="en">
    <head>

    </head>
    <body>

    </body>
</html>
```

Head Code

Within the head element, we need to put some important tags. We need to declare character encoding using the charset meta tag. We require a viewport meta tag to set and allow responsive scaling on mobile devices. You’ll also have to add a title for your document and a description (up to 156 characters) of your app or site. Finally, you’ll want to add links to your external CSS and JavaScript files (for when you create them later on). They should appear in this priority order.

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-
scale=1">
<title>Title - Subtitle</title>
<meta name="description" content="Description.">
<link rel="stylesheet" href="assets/styles.css">
<script async src="assets/script.js"></script>
```

You can include many other tags, including meta and link elements, which we covered in the first chapter. If you need a reminder, take a look as you might want to add social media, asset links, progressive web application features, etc (depending on your projects needs).

Body Code

Within the body elements, we're going to split the content into three defined sections using HTML elements called header, main, and footer. In the header, you'll put your logo and navigation. I've included an example of what could be used within a portfolio website.

```
<header>
    <h1>Logo</h1>
    <nav>
        <ul>
            <li>Articles</li>
            <li>Work</li>
            <li>Info</li>
        </ul>
    </nav>
</header>
```

Next on the list is the main body of content, hence the main element. You'll want to split your article or app into clearly defined regions via the article element, which will help (for accessibility and legibility). Think of them like chapters in a book or screens on your app which you can hide or display as required, and you'll understand their hierarchy purpose. You can only have one main element, but you should provide as many article elements as you require for regions.

```
<main>
    <article>
        <h2>Section</h2>
        <p>Content</p>
    </article>
    <article>
        <h2>Section</h2>
        <p>Content</p>
    </article>
</main>
```

Finally, we have the footer, which is where you store useful links that aren't suitable for the navigation and, more importantly, copyright information on your site. An example you can use is shown below.

```
<footer>
    <p>Copyright, 2021</p>
    <p>Built by <a href="#">Your Name</a>.</p>
</footer>
```

Summary

And that's just about it. I would rather not place anything else in the file, as it would assume too much of your project. If you're building an app or one-page website, you won't want bundles of links, so you can alter the navigation links as required. If you use a JavaScript tooling system that generates HTML on-the-fly then your output may vary in terms of quality; that being said, semantic, clean HTML is still just as important as ever before, with accessibility being critical to UX.

Reference	Don't forget to validate your code. Errors in your HTML syntax can cause accessibility issues so it's critical that your code is well formed. https://validator.w3.org/

Chapter 17:

Have all your documents in order for visitors

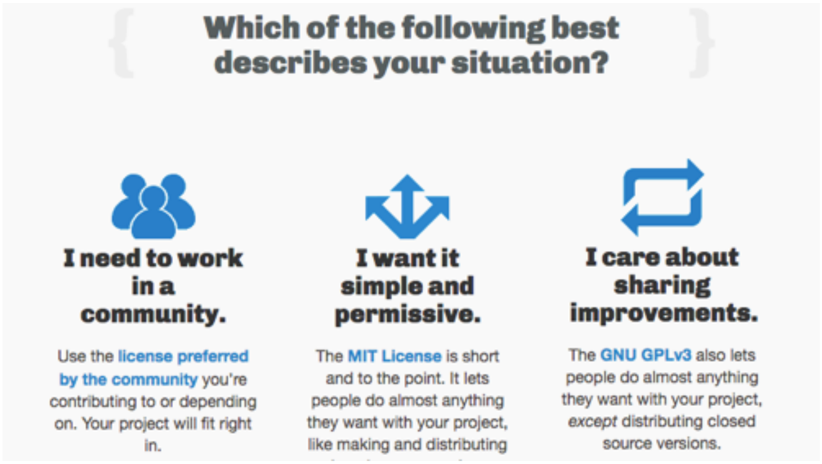
Build license, terms of service, impressum, accessibility statements, and privacy policy docs for your site.

license.txt

When creating projects, or using existing works created by others, you'll want to license your product to ensure that it's only used in the manner you wish. Back in the old days of app and site development, you'd have to create your own license agreement and the legality of these files was largely untested. These days, boilerplate agreements, written by legal experts, exist to cover uses you're likely to need.

Occasionally, you'll need to use a specific agreement if you are dependent on a particular licensed system (WordPress, for example, requires GPLv3). Other times you might use multiple licenses as your work will utilize code from various licensed libraries. Before we pick which license you'll use, you will want to create a license.txt file and link to it within your HTML (using the rel="license" attribute).

So which license should you pick? There's far too many to choose between (without it getting confusing), however a few stand out as community favorites: Apache, GPL v3, MIT and CC. Plus, there's also Blue Oak and Hippocratic license that rightly deserve some attention. I've provided a table below of the licenses, what they cover, what they don't and, you can decide for yourself if it's the one for you.



Which of the following best describes your situation?

I need to work in a community.

Use the license preferred by the community you're contributing to or depending on. Your project will fit right in.

I want it simple and permissive.

The MIT License is short and to the point. It lets people do almost anything they want with your project, like making and distributing

I care about sharing improvements.

The GNU GPLv3 also lets people do almost anything they want with your project, *except* distributing closed source versions.

Reference The below table is based upon the choose a license project. If you are having trouble picking one for your work, check out https://choosealicense.com/

Name	Permissions	Conditions	Limitations
	Commercial Use		
	Distribution	Include License	Limited Liability
Apache 2.0	Modification	Include Copyright	Trademark Use
	Patent Use	List All Changes	No Warranty
	Private Use		
License: https://www.apache.org/licenses/LICENSE-2.0			
	Commercial Use	Include License	
	Distribution	Include Copyright	
GNU GPL v3	Modification	List All Changes	Limited Liability
	Patent Use	Disclose Source	No Warranty
	Private Use	Same License For All Modifications	
License: https://www.gnu.org/licenses/gpl-3.0.en.html			
	Commercial Use		
	Distribution	Include License	Limited Liability
MIT	Modification	Include Copyright	No Warranty
	Private Use		
License: https://opensource.org/licenses/MIT			

Name	Permissions	Conditions	Limitations
Creative Commons	Commercial Use Distribution Modification Private Use	Include License Include Copyright List All Changes Same License For All Modifications	Limited Liability Patent Use Trademark Use No Warranty
License: https://creativecommons.org/choose/			
Blue Oak	Commercial Use Distribution Modification Patent Use Private Use	Include License Include Copyright List All Changes Disclose Source Same License For All Modifications	Limited Liability Patent Use Trademark Use No Warranty
License: https://blueoakcouncil.org/license/1.0.0			
Hippocratic	Commercial Use Distribution Modification Private Use	Include License Include Copyright Adhere To Human Rights Principles Adhere To Human Rights Laws	Limited Liability No Warranty
License: https://firstdonoharm.dev/version/2/1/license/			

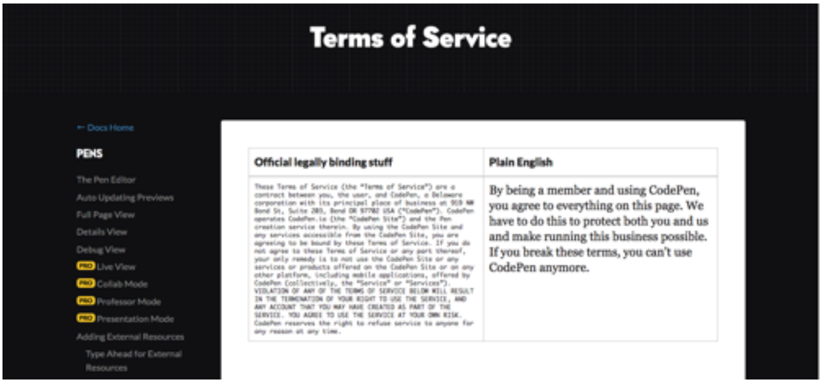
If you care about sharing improvements but need some (or all) of the product not to be open source whilst distributed with the license, Apache may be for you. If you want to just share improvements of your open source, GNU GPL v3 will work well. If you want something simple and permissive, go with MIT. If your license isn't for code or software, try Creative Commons. If you require patent and trademark liability avoided, try Blue Oak. Or if you don't want your work to be used for unethical purposes, the Hippocratic license will work well.

Reference

Lawyer Kyle E. Mitchell makes a good argument for the potential disadvantages for using MIT and BSD licenses, to read his opinion visit https://writing.kemitchell.com/2019/03/09/Deprecation-Notice.html

terms.txt

It's now worth touching on a file that many websites and web apps also include, which can be equally full of legal jargon but is just as important for projects to provide. The terms.txt file (or HTML file) will contain the terms of service agreement that users refer to when signing up for your service, using your products, or browsing your website. Creating one of these files requires time and a fair bit of skill.



Terms of Service

Official legally binding stuff	Plain English
These Terms of Service (the "Terms of Service") are a contract between you, the user, and CodePen, a Delaware corporation with its principal place of business at 919 NW Bond St, Suite 203, Bend OR 97702 USA ("CodePen"). CodePen operates CodePen.io (the "CodePen Site") and the Pen creation service therein. By using the CodePen Site and any services accessible from the CodePen Site, you are agreeing to be bound by these Terms of Service. If you do not agree to these Terms of Service or any part thereof, your only remedy is to not use the CodePen Site or any services or products offered on the CodePen Site or on any other platform, including mobile applications, offered by CodePen (collectively, the "Service" or "Services"). VIOLATION OF ANY OF THE TERMS OF SERVICE BELOW WILL RESULT IN THE TERMINATION OF YOUR RIGHT TO USE THE SERVICE, AND ANY ACCOUNT THAT YOU MAY HAVE CREATED AS PART OF THE SERVICE. YOU AGREE TO USE THE SERVICE AT YOUR OWN RISK. CodePen reserves the right to refuse service to anyone for any reason at any time.	By being a member and using CodePen, you agree to everything on this page. We have to do this to protect both you and us and make running this business possible. If you break these terms, you can't use CodePen anymore.

There's plenty of statistics about how few people actually read the terms and conditions, so one thing you'll need to ensure is, regardless of what you put within the document, be sure it's human-readable. You will be setting the rules for who can use your product or service, so it's critical you are clear, concise and don't re-write war and peace!

Sections you should provide include the following:

Section	Description
Overview	By using the services, you agree to the terms. They exist to protect both you and us. If you break the terms and conditions, you can't use the services anymore. Simple and direct.
Your Account	Your account is your responsibility. If your account is compromised, we'll try to help, but can't guarantee we can fix any damage.
License To You	You can use our stuff, just don't copy it unless authorized to under the license.

Section	Description
License To Us	You can upload stuff to us, automatically it’s licensed under a specific license. Also, don’t infringe copyrights or do bad stuff.
Responsibilities	You must be over 13. If you're under 18, have a legal guardian's consent. Don’t lie on the signup form. Obey the law. If an account is hacked, please let us know. You’re responsible for your equipment and Internet access. We don’t endorse what’s posted, you're responsible for what you post.
Violations	Don’t upload anything bad. Don’t pretend to be (or misrepresent / associate with) someone. Don’t steal login data, personal data or payment information. Don’t breach intellectual property rights. Don’t spam or scam on the service. Don’t send malware. Don’t break, hack or disrupt the service, exploit code, or steal any functionality without permission. If we need to stop a users' account, we will. If we legally have to provide information to a government request, we will do, but not otherwise.
Feedback	If you send us feedback, we may use it, in which case it’ll be covered by these rules.
Payment	All sales are final. If we bill monthly, we will automatically re-bill every 30 days, if you’re on an annual plan, we automatically re-bill every 365 days. You may need to pay taxes based on where you live. If you downgrade, you will lose access to the features in the higher tier.

Section	Description
Violations	We will delete anything that violates our terms, including accounts. We are entitled to take legal action against violators.
Cancellations	If you don't want an account anymore, it’s up to you to delete it. If you delete an account, it’s gone forever. If you’re a paid user, you won’t be re-billed once your account is deleted.
Modifications	We would rather not shut down a service, but if we have to, we will do. We don’t need to give notice. If we change prices, you will receive 30 days notice. If you don’t like changes to the service, you can choose to leave. We will let you know of changes to our terms of service.
Copyright	The things you upload to the service are yours unless licensed otherwise.
Infringement	If you need to report a copyright, patent or trademark issue, contact us.
Disclaimer	There might be bugs, or downtime, or features that don’t meet your needs or requirements. You are responsible for your own actions. We respond to help requests but give priority to our paying users. You may have other rights, and this agreement won’t affect those.
Liability	We aren’t liable for misuse of the service or for it not meeting your needs, like any downtime costing you business, etc. We also aren’t liable for acts of nature or anything beyond our reasonable control occurring.

Section	Description
Indemnity	If you get in trouble with the law (or a third party) for something you did using our service, we'll pass the message onto you, but we're not responsible for your actions.
General	If their terms conflict with local laws, the rest still apply. Questions about the terms should be directed to us for answering.

As you can see, there are plenty of different sections which you can include in your document. Drafting one is usually the job of the legal department of a business; but for individuals working without legal support, adapting one you find online or using a template can often work quite well. Just be certain you keep your terms as short and jargon free as possible, and be as transparent with visitors as you can.

Reference	For a good example of a terms of service: https://blog.codepen.io/documentation/terms-of-service/, for a template you can use: https://www.pandadoc.com/website-standard-terms-and-conditions-template/

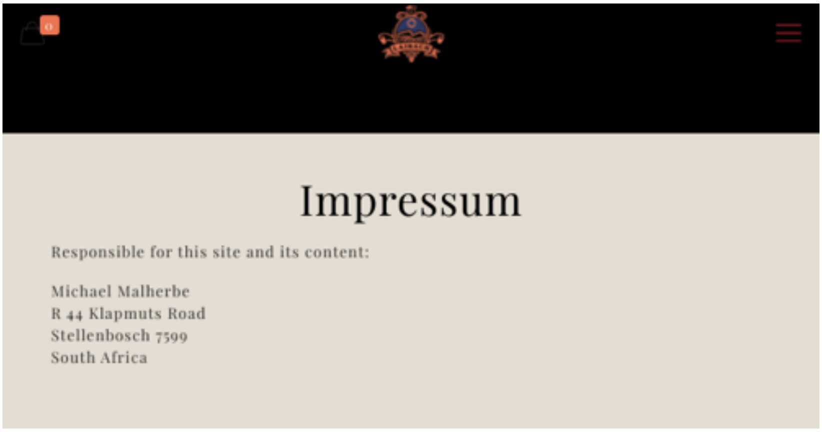
impressum.txt

Similar to a legal disclosure document, an Impressum is a legally required document that all Germanic websites must include to declare ownership and authorship of a document. Whilst this won't have much of a baring if you're building a local site for a non-German country, international sites may want to include this for german customers (and German websites have to have it by law).

The below describes what you'll need to include in impressum.txt:

Section	Description
Name	Business Name
Address	Full Business Address
Contact	Email, Phone, Fax, Etc
Owner	Name & Position
License	Commercial Registration Numbers & VAT Number
URL	Address Of Impressum Document

You can format this as either a text or an HTML document (just like any of these). It's just my personal preference to use plaintext, as it has no software requirements to display correctly for end-users. You can store it in the same location as your other legal web documents.



Impressum

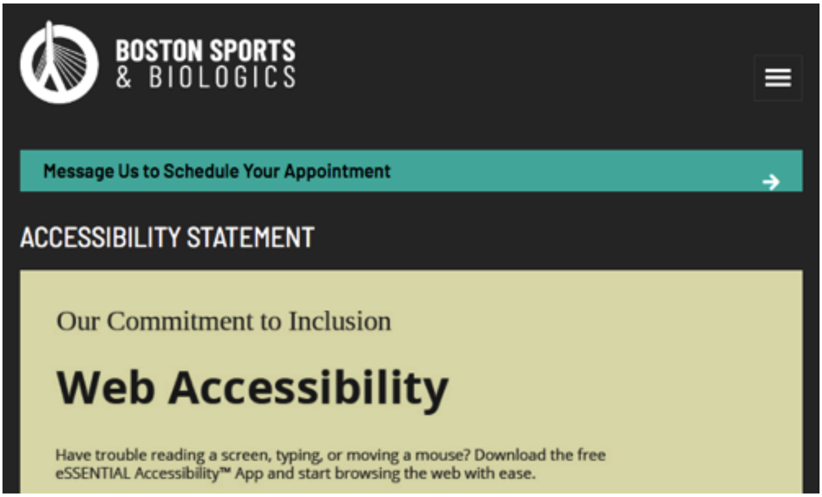
Responsible for this site and its content:

Michael Malherbe
R 44 Klapmuts Road
Stellenbosch 7599
South Africa

Reference For a detailed guide to Impressum files and what to include, visit https://www.linkedin.com/pulse/what-impressum-why-does-facebook-want-one-chris-bangs

accessibility.txt

Everyone agrees about its importance, but it's still an underestimated tool in the web developers toolkit. Accessibility and inclusive design remains something that can bring massive benefits to a wide range of visitors and can open up an entirely new audience to your services. Making your site accessible from the offset is the ideal, but having a statement that declares your accessibility credentials will do a lot to instill trust, as your transparency over your practices will be shown.


BOSTON SPORTS
& BIOLOGICS
Message Us to Schedule Your Appointment
ACCESSIBILITY STATEMENT
Our Commitment to Inclusion
Web Accessibility

Have trouble reading a screen, typing, or moving a mouse? Download the free eSSENTIAL Accessibility™ App and start browsing the web with ease.

So let's create an accessibility statement. After all, it will provide users with information about the inclusiveness of your content and show your social responsibility and that you care about them. You will first want to create your accessibility.txt (or HTML) and create a series of sections as shown in the below table. Remember, as always, to keep things human-readable. This is especially relevant as individuals may struggle to read lengthy blurbs of content if they have a disability.

Section	Description
Overview	A commitment to inclusive design, not restricted for people with disabilities, or different backgrounds.
Conformance	Declaration identifying what level of compliance the service has achieved against the Web Content Accessibility Guidelines. Also, if your code validates, you comply with accessibility laws, and if you meet any other inclusive design accreditation that exists.
Compatibility	Which browsers, devices and environments have been tested against to ensure compatibility. Also, what accessibility aids have been tested against to ensure the greatest possible usability. Note the syntax languages required for visitors to browse.
Features	Measures you have taken to improve inclusiveness such as training staff, including users with disabilities in the testing phase, performance, any code-based features (alt tags, skip links, etc), and more.
Issues	Any known issues with the website or app (such as compatibility), which might impact the end-user. List solutions that can be used until these are fixed.
Contact	The point of contact for accessibility related issues.

This is only a general overview, and you should expand upon it if you find additional things that your audience would find useful to be aware of. Accessibility is an ever-expanding subject that needs constant vigilance, and you shouldn't take the visitor for granted. By stating your intentions and (hopefully) backing it up by action in the way you code and offering compliant, accessible markup, your products, and services will be all the better for your visitors.

If this wasn't enough of an incentive, recently, accessibility legal battles have been escalating and lawsuits against websites have been running into the thousands with hundreds of millions in damages being awarded collectively. The risk of being sued shouldn't be your only motivation for change, but it should naturally raise awareness that disabled individuals have equal rights to able-bodied individuals.

Reference For an overview of an accessibility statement, with included examples that you can freely adapt, visit https://www.w3.org/WAI/planning/statements/

privacy.txt

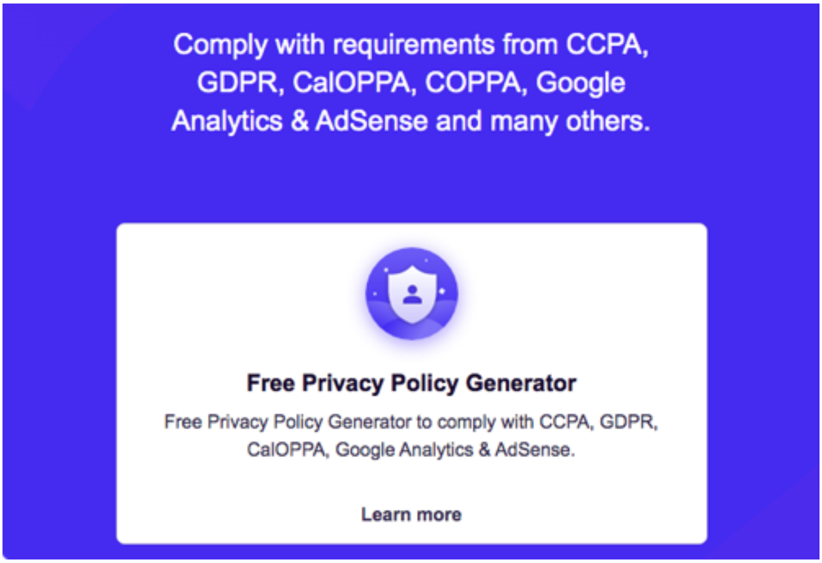
In the Lord of the Rings, Gandalf wisely told Frodo (regarding the ring of power) to "keep it secret, keep it safe". On the Internet, the ability to keep things secret and safe is central to providing usable apps and services to users. The right to privacy isn't just a bonus feature, it's part of the law and something we need to all abide by as providers.

Reference If you need a privacy policy that has been pre-drafted, this service will generate one for you which compiles with the law https://www.freeprivacypolicy.com/

We'll create our privacy.txt (or HTML) file and keep it with the terms of service agreement, as the two files go hand in hand when you run a digital service. Again, if you're creating one of these files you'll usually hand this off to the legal department, but if you don't have one, as many of us don't, you could adapt one you find online (or better yet, use a pre-made template built by lawyers). You'll also need it to be GDPR friendly to ensure that you comply with international laws!

GDPR is a data protection law that the EU brought in that not only governs data collection by EU companies, but any business that does business with an individual who lives in the EU. So, unless you're just serving to your own nation, you need to be GDPR-compliant to avoid potential lawsuits. So what content do we need in this policy? On the next page, I've included a table of the sections you'll need in yours.

Reference	Smashing Magazine has provided an excellent series on GDRP if you want a detailed guide on the subject from it's launch https://www.smashingmagazine.com/2018/02/gdpr-for-web-developers/ to new updates to the legislation https://www.smashingmagazine.com/2021/02/state-gdpr-2021-key-updates/


Comply with requirements from CCPA, GDPR, CalOPPA, COPPA, Google Analytics & AdSense and many others.

Free Privacy Policy Generator

Free Privacy Policy Generator to comply with CCPA, GDPR, CalOPPA, Google Analytics & AdSense.

Section	Description
About	Who we are. This policy lays out how we collect, use and disclose personal information you provide. If you don’t agree, don’t use the service. We'll also keep you up-to-date of any changes relating to the policy.
Collection	Types of data collected (Names, Email, Cookies, Logs, Analytics), Purpose of collection (Why it’s needed, what it’s used for, and for how long it's kept on the servers).
Processing	Third-party data collection and sharing that occurs. How information is used generally on the website.
Protection	Measures in place to protect information. Choices to limit the use of personal data (tracking, close account).
Rights	Your right to access your personal data, correct and delete your information. Plus complaint's procedure, GDPR requests and contact us regarding the policy.

Aside from that, you should have all the documents you’ll need. With a license, terms of service, impressum, accessibility statement, and privacy policy you’ll be covered from multiple angles when launching a project and visitors will know where they stand with your service. As I noted earlier, it’s worth getting legal advice when drafting some of these files if you can afford it, but otherwise, pre-drafted documents on the web, though-generic may generically suit your project.

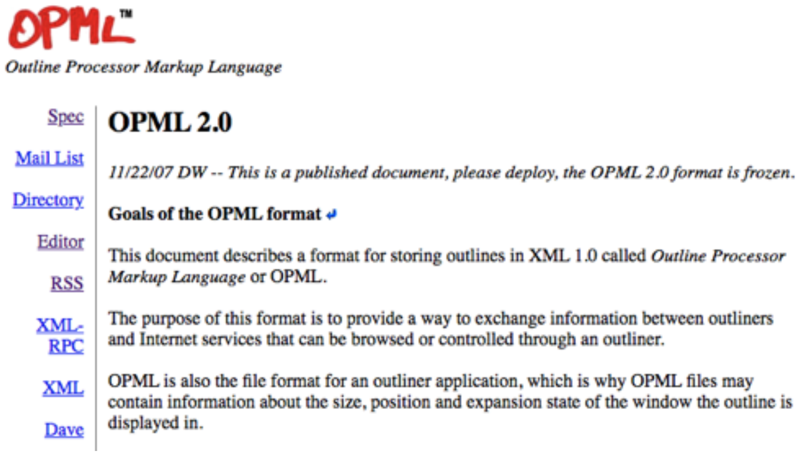
Chapter 18:

Gathering similar curated content to share

Create a curated list of items, whether links or media, to show your visitors what they really should subscribe to.

feed.opml

Back in Chapter 12, we examined various syndication formats that may be used to push news and podcasts to subscribers to your websites. In this chapter, we're going to look at a related format called OPML, which is formatted using the XML language and is often used to list multiples of items (such as a collection of links or media files). And it's this very useful feature that could be used to bring value to your site.



OPML™ Outline Processor Markup Language

Spec
Mail List
Directory
Editor
RSS
XML-RPC
XML
Dave

OPML 2.0

11/22/07 DW -- This is a published document, please deploy, the OPML 2.0 format is frozen.

Goals of the OPML format

This document describes a format for storing outlines in XML 1.0 called *Outline Processor Markup Language* or OPML.

The purpose of this format is to provide a way to exchange information between outliners and Internet services that can be browsed or controlled through an outliner.

OPML is also the file format for an outliner application, which is why OPML files may contain information about the size, position and expansion state of the window the outline is displayed in.

Sometimes OPML is used in writing apps to produce table of contents or document outlines, typically it's used in feed readers to subscribe users to multiple RSS feeds at once, but in our case (for websites and apps), we are going to use it for one of two purposes. We shall use it for sharing a playlist of music files which you could share with your visitors, and we shall use it for a list of hyperlinks for use in a library.

Reference For a comprehensive guide to OPML syntax, read the specification at http://dev.opml.org/spec2.html

Overview

Every OPML file begins the same, you'll create your feed.opml file adding the XML doctype along with the OPML tags, plus within this (just like in HTML) you'll want to add a head and body element.

```
<?xml version="1.0" encoding="utf-8"?>
<opml version="1.0">
    <head>

    </head>
    <body>

    </body>
</opml>
```

Within the head element, there are several elements we can use. Of them all, only the title is required. Once you've added your title tags, you can begin filling your body element with either a playlist of music files or a list of hyperlinks using the elements from the table below.

Element	Description
title	The title of the document.
ownerName	The individual who created the document.
ownerEmail	The email of the person who created the file.
ownerId	The website where the creator can be contacted.
dateCreated	The date and time of the document's creation.
dateModified	The date and time of the file's modification.
docs	A website link to the OPML specification.
expansionState	Comma separated list of line numbers to expand.

Element	Description
vertScrollState	The line to be displayed at the top of the window.
windowTop	Pixel location of the top edge of the window.
windowLeft	Pixel location of the left edge of the window.
windowBottom	Pixel location of the bottom edge of the window.
windowRight	Pixel location of the right edge of the window.

Categories

So let's start by grouping our playlist into categories. Within the body tag, you'll want to add two outline elements with text attributes that contain a text description of what best matches the style of music that'll play within them, or the types of hyperlinks that are contained within that section. Grouping is helpful for users to quickly navigate a potentially long list of items, as they can then find relevant results.

```
<outline text="Classics">

</outline>
<outline text="Fun">

</outline>
```

Content

Next, we need to begin customizing the OPML file for our content types. First up is the playlist of music files. You might automatically think of this as being useful for something like podcasts - and arguably it could be used for this purpose; though a syndication format like RSS is better suited as it allows images and descriptions and other useful descriptions. OPML music playlists would be ideal instead for a custom Spotify playlist (which is free of JavaScript!).

Then there is the list of hyperlinks. This is where I think OPML is at its most useful, as you could use it for sharing a list of literally anything. A list of links to your friends websites, resources you've bookmarked, Amazon links to books worth buying, collated educational material on a subject, a birthday, or wedding list, or pretty much anything else that you could group together. It's entirely up to your imagination.

No matter your choice, within each of the categories you will want to include some songs or hyperlinks. Below is an example of what your code will look like. All outline elements contain a type attribute that describes whether it's a song or link. While the attributes used differ slightly, they should be easy enough to manage in one file. You can mix and match both links and media within a single file if you wish.

Code for Songs:

```
<outline text="Song.mp3" type="song" f="Artist - Song.mp3"/>
```

Code for Hyperlinks:

```
<outline text="Useful Website" type="link" url="https://
example.com"/>
```

Finally, as with many of these unusual file types you'll want to include a reference to the file in the head of your HTML document (as certain feed readers can take advantage of these files). Plus you could also link to the file directly to allow visitors to view the links natively in the web browser (if supported) as the content (and its links) should be accessible even if the visitor doesn't have access to a feed reader.

```
<link rel="alternate" type="application/text+xml" href="opml.xml">
```

This concludes the chapter on OPML. While RSS is great for pushing fresh content to visitors, and websites are great for allowing people to read content in a stylish manner, OPML allows you to compile lists of links you think will be useful and offer them to visitors. It's a small, useful file that you might yet find some interesting use-cases for in your workflow. Who knows, with a little JavaScript you could even build a media player that can use the OPML playlists you've built.

Chapter 19:

Getting your layout optimized for preferences

Generate a default dark mode, prefers reduced motion, monochrome & prefers reduced data stylesheet.

modes.css

In a previous chapter we've examined the wonderful world of HTML, next it's time to look at some CSS that can be used to benefit your visitors. Or to be more specific, some media queries you can use to target certain browsing environments that a visitor might potentially be encountering. We won't be targeting screens or printers in this chapter (those are technical enough to require their own dedicated chapters), but these cover more niche cases of customization.

Reference	CSS is an evolving language with more standards being produced. To learn more about the work being done that affects media queries, read the specifications at https://www.w3.org/Style/CSS/specs.en.html and https://www.w3.org/TR/css3-mediaqueries/

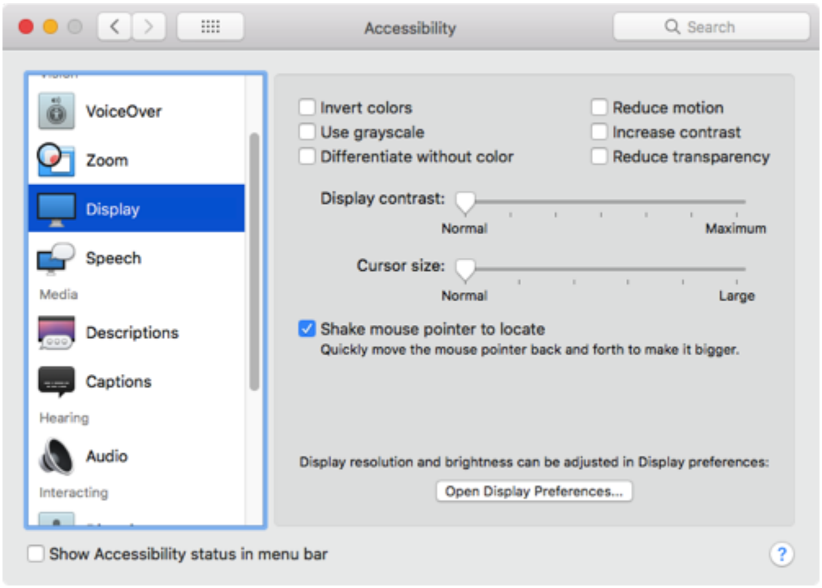
For this book, I've chosen to separate any media query that changes the website or apps layout based on the user's environment into a separate file called modes.css (away from the base styles.css and print.css files), however if it's your personal preference to reduce HTTP requests and manage your CSS under one roof, you can keep it in the one CSS document. Ready to begin? Let's create this modes.css document and start styling our HTML with some preferential CSS.

Reference	If you want to test the effects of the below, this article should help you get a true-to-life simulation https://www.a11yproject.com/posts/2020-01-23-operating-system-and-browser-accessibility-display-modes/

Inverted Colors

MacOS (and more specifically Safari) supports an unusual media query called inverted-colors, which detects if the OS has the accessibility mode set to invert all the colors on the screen. This may seem like a variation on the concept of light and dark mode, but it's useful and designed by Apple for people with low or impaired vision to help them navigate the screen. If you change colors, make sure you provide enough contrast for text visibility when colors flip.

```
@media (inverted-colors: none) {

}
@media (inverted-colors: inverted) {

}
```



- [ ] Invert colors
- [ ] Use grayscale
- [ ] Differentiate without color
- [ ] Reduce motion
- [ ] Increase contrast
- [ ] Reduce transparency

Display contrast: Normal … Maximum

Cursor size: Normal … Large

- [✓] Shake mouse pointer to locate

Quickly move the mouse pointer back and forth to make it bigger.

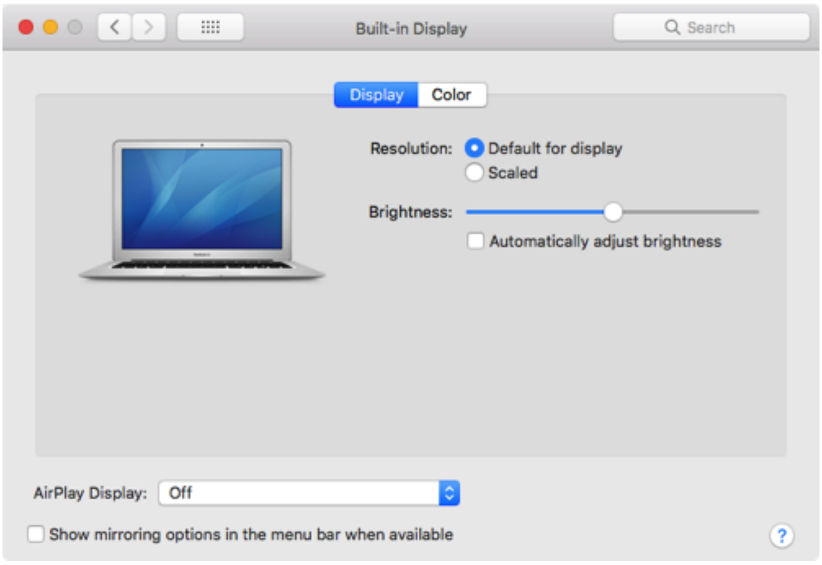
Display resolution and brightness can be adjusted in Display preferences:

- [ ] Show Accessibility status in menu bar

Light Level

Next, we have a useful tool which is useful especially on mobile devices and tablets called the light-level media query. If you use a smartphone regularly, you'll know the ambient light sensor alters the brightness of your screen based on how light or dark it is where you are. Using light level, you can change your visuals (for example the color scheme) based on if the visitor is reading in the dark (dim), in the sun (washed), or in a fairly lit region (normal). Pretty cool stuff.

```
@media (light-level: washed) {

}
@media (light-level: normal) {

}
@media (light-level: dim) {

}
```



Resolution: [✓ Default for display] Scaled

Brightness:

Automatically adjust brightness

AirPlay Display: Off

Show mirroring options in the menu bar when available

Pixel Density

If you want your images to look great on a variety of different devices, you'll know that sometimes, visitors will occasionally have hardware with different resolutions and pixel densities. To compensate for this, the min-resolution media query allows us to target devices by 1x (for regular) or 2x (for retina) so that we can serve images only to those devices that will appreciate them. Safari doesn't support min-resolution, so it has its own prefix to enable support.

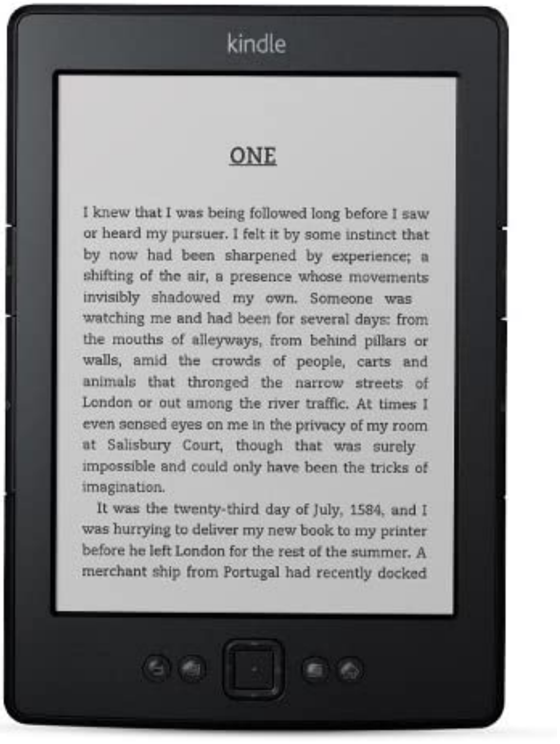
```
@media (min-resolution: 1x), (-webkit-min-device-pixel-ratio: 1) {

}
@media (min-resolution: 2x), (-webkit-min-device-pixel-ratio: 2) {

}
```

Monochrome

As web developers, we take color support for granted. We use it liberally within our layouts and assume (naturally) that our visitors can see everything that is on our screens. Occasionally, however, a visitor might (for accessibility reasons) choose to turn their screen to monochrome mode, either to save battery (on mobile devices) or to improve contrast and their ability to read content on screen. Using color and monochrome media queries can help you out here.

You could provide a dedicated experience for eInk readers (for example) which cannot see in color, or using the print media type you could give a custom layout to monochrome printers. You could swap out images for alternatives which look better without color, provide a more stripped back visual experience or a layout with less clutter in terms of background images and contrast, with more focus placed upon the content. Visitors may prefer this focused version!

```
@media (color) {

}
@media (monochrome) {

}
```



ONE

I knew that I was being followed long before I saw or heard my pursuer. I felt it by some instinct that by now had been sharpened by experience; a shifting of the air, a presence whose movements invisibly shadowed my own. Someone was watching me and had been for several days: from the mouths of alleyways, from behind pillars or walls, amid the crowds of people, carts and animals that thronged the narrow streets of London or out among the river traffic. At times I even sensed eyes on me in the privacy of my room at Salisbury Court, though that was surely impossible and could only have been the tricks of imagination.

It was the twenty-third day of July, 1584, and I was hurrying to deliver my new book to my printer before he left London for the rest of the summer. A merchant ship from Portugal had recently docked

Mouse Pointer

Next we shall examine the mouse pointer. For years, it was a hard job trying to create equal navigation systems for desktop and mobile due to CSS hover effects not being available to mobile devices. Now, thanks to this media query, you can code alternative layouts to match the type of pointer they have and avoid needing to rely upon having the same problematic browsing experience on all platforms.

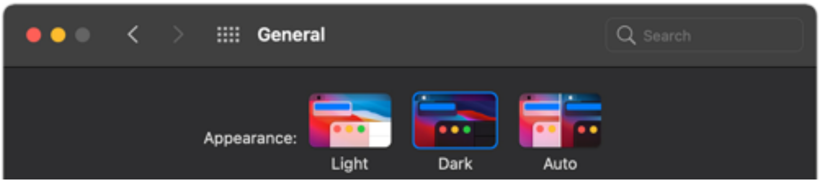
To cover all of your pointer bases, do the following: If you're on a touch only device, set the hover to none and the pointer to coarse. If you use a stylus on a touchscreen, set the hover to none and the pointer to fine. If you use a controller like a joystick or Wii-mote, then set hover to hover and pointer to coarse. And if you want a media query for normal situations, set hover to hover and pointer to fine.

```
@media (hover: none) and (pointer: coarse) {

}
```

Dark Mode

Next we'll cover the popular feature of light and dark mode, which has become a prerequisite for any site or app. Using the prefers-color-scheme media query, we can replace colors and images on our site to give a fluid experience for users who prefer to browse using the feature. Why would we do it? For those with OLED screens, it will save battery. Dark mode may help reduce eye strain too.

```
@media (prefers-color-scheme: dark) {

}
@media (prefers-color-scheme: light) {

}
```



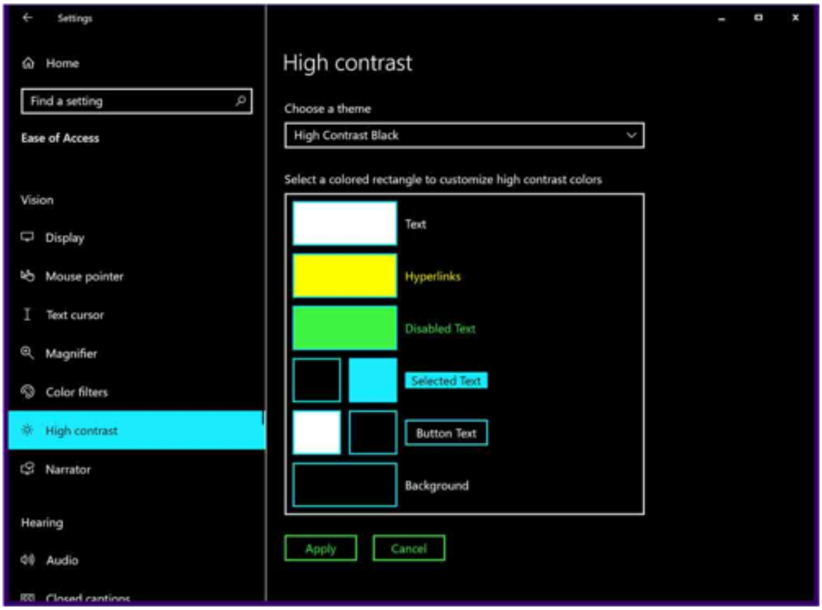
If you have the time and budget to implement a proper dark mode on your site, it's well worth it for the user-experience alone, as many will have a preference. If you don't, below is a rapid and dirty method of implementing a default (though use it at your own risk).

```
@media (prefers-color-scheme: dark) {
:root {
    background: #101010;
    filter: invert(1) hue-rotate(180deg); }
    canvas, img, video {
        filter: invert(1) hue-rotate(180deg); }
}
```

High Contrast

Much less extreme than inverting colors, the prefers-contrast media query which is used by a significant number of visually impaired web users and those with medical conditions such as migraine disorders (using a custom color profile) is a great accessibility media query to include in your toolchain. You will be able to change your layout based on whether the user prefers a reduced (regular site) or higher contrast version of your website so that your content becomes easier to read (or softer on the eyes). It's well worth investigating.

Reference	For a more comprehensive guide to WHCM mode, read https://www.smashingmagazine.com/2022/06/guide-windows-high-contrast-mode/ or https://adrianroselli.com/2021/02/whcm-and-system-colors.html


Settings

Home

Find a setting

Ease of Access

Vision

Display

Mouse pointer

Text cursor

Magnifier

Color filters

High contrast

Narrator

Hearing

Audio

Closed captions

High contrast

Choose a theme

High Contrast Black

Select a colored rectangle to customize high contrast colors

Text

Hyperlinks

Disabled Text

Selected Text

Button Text

Background

Apply

Cancel

When supported in all modern browsers, it will use the following:

```
@media (forced-colors: none) {

}

@media (forced-colors: active) {

}
```

In Apple Safari (where it works), You use the following:

```
@media (prefers-contrast: more) {

}
@media (prefers-contrast: less) {

}
```

In Microsoft Edge (where it works), You use the following:

```
@media (-ms-high-contrast: active) {

}
@media (-ms-high-contrast: black-on-white) {
}

@media (-ms-high-contrast: white-on-black) {
}
```

Data Saver

Next we need to talk about bandwidth. It's not something that's very fun to cover because every website these days is pretty greedy. And no matter how much everyone harps on about improving a site's performance, the web keeps getting increasingly bloated. Using the upcoming media query called prefers-reduced-data, we will be able to use CSS to prevent the unnecessary download of background images, fonts, or other resources requested via CSS. Useful indeed.

```
@media (prefers-reduced-data: reduce) {

}
```

Reduced Motion

Next we're going to look at an accessibility aid that can improve things for people who suffer from motion sickness, migraines, inner-ear conditions or epilepsy. The prefers-reduced-motion media query allows us to show less animated effects on the screen to those who request it. You should note that this doesn't mean that visitors want no animation to appear, it just means they want less of it, and it to be much more carefully used for their benefit as they are less tolerant.

```
@media (prefers-reduced-motion: reduce) {

}
@media (prefers-reduced-motion: no-preference) {

}
```

If you're implementing a reduced motion feature-set on your website or app, make sure to include things like CSS and JavaScript animations as well as animated GIFs and videos, as they can also be triggers for people with medical conditions. If you don't have time to craft a custom stylesheet, the below quick and dirty script should work by targeting every element and reducing its animation (it won't work on images and video though, so you'll want to expand upon the code).

```
@media (prefers-reduced-motion: reduce), (update: slow) {
    *:not(.safe-animation),
    *:not(.safe-animation)::before,
    *:not(.safe-animation)::after {
        animation-delay: -1ms !important;
        animation-duration: 1ms !important;
        animation-iteration-count: 1 !important;
        background-attachment: initial !important;
        scroll-behavior: auto !important;
        transition-delay: -1ms !important;
        transition-duration: 1ms !important; }
}
```

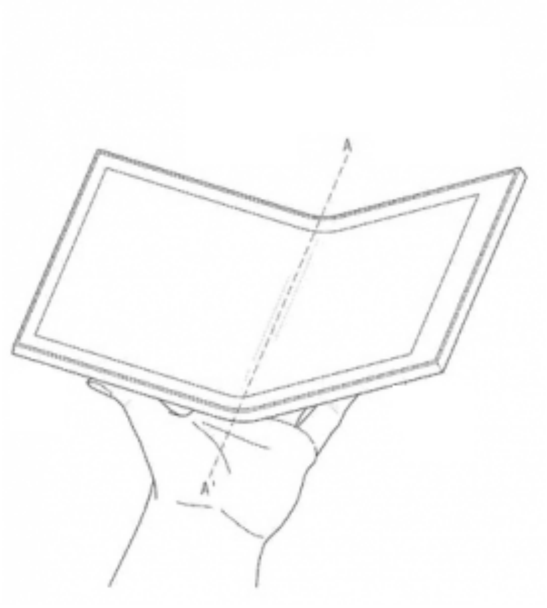
Reduced Transparency

Another accessibility aid which is upcoming in the CSS specifications is prefers-reduced-transparency, which tells browsers that the visitor would prefer not to see too many background images or overlapping features that might cause distractions. Remember that visitors with memory problems, dyslexia, or learning difficulties could be affected. Think of this media query as a minimalistic workspace where visitors can engage with content with a barebones, clean colored layout.

```
@media (prefers-reduced-transparency: reduce) {

}
@media (prefers-reduced-transparency: no-preference) {

}
```

Foldable Displays

Finally, we need to examine something pretty new and exciting that Chrome recently added (and other browsers won't be far behind) called the screen-spanning media query aimed at duel-screen devices. Yep, for all those new folding phones that use two displays, this media query can allow you to ensure your site handles the fold that occurs between both sides of a display correctly. You can also go horizontal or vertical too, as the fold changes if users rotate a screen.



```
@media (screen-spanning: single-fold-horizontal) {

}
@media (screen-spanning: single-fold-vertical) {

}
```

Reference For more information about supporting folding screens, read this article https://docs.microsoft.com/en-us/dual-screen/web/css-media-spanning

Adding these components can bring more accessibility, performance, and visual customization for the visitor. It will take time to implement the styles into your workflow, but the benefits outweigh the input. In the future, we'll likely see more media queries like detection for scripting, update refresh timing, HDR detection, and more. Until then, it's worth keeping an eye on new CSS specifications that are released.

Reference The Polypane team have put together an equally comprehensive guide to media queries online if you are interested in the latest cutting edge developments in CSS at https://polypane.app/blog/the-complete-guide-to-css-media-queries/

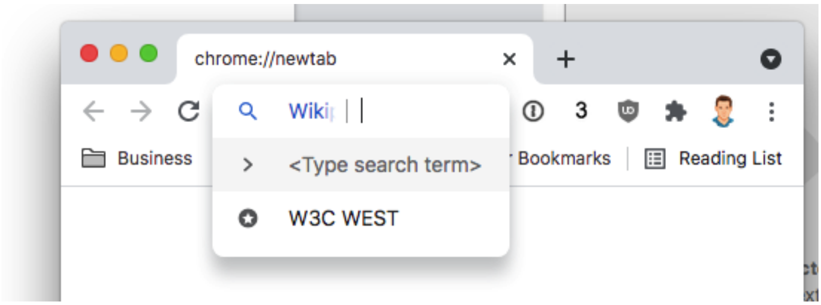
Chapter 20:

Making a web browser search compatible

Add your sites search engine easily into the user's browser to improve their access to your service.

opensearch.xml

Making your website more search engine friendly is the goal of most projects, as being visible to the widest possible audience allows you to stay in business. While traditional SEO relies on using techniques that third-party search engines will discover you using via their robots, spiders and algorithms; the OpenSearch system lets you embed your site's search engine into the users' browsers search from URL bar list.



It might not seem like much at first, but the OpenSearch file lets visitors to your website easily find items in your store without having to first load up your homepage. They can type in a shortcut followed by their search terms in their URL bar and be driven straight to your search engine's results page. It may increase sales conversions for people on desktop devices who are after information in a hurry.

Reference For a comprehensive guide to OpenSearch, read the specification at https://github.com/dewitt/opensearch/blob/master/opensearch-1-1-draft-6.md

Search Creation

To create an opensearch.xml file, you'll want to create a blank document in the base directory of your website. Within it, you will need an XML declaration and some OpenSearchDescription elements to contain all the available tags to describe your search engine to browsers. Below is a sample of the code you'll need to successfully create an OpenSearch document with a table listing all the available elements which exist. As you can see, URL has no closing tag but contains a MIME type attribute, plus a template (URL) attribute.

```
<?xml version="1.0" encoding="utf-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/
opensearch/1.1/">
    <ShortName>Title</ShortName>
    <Description>Description.</Description>
    <Url type="text/html" template="http://www.google.com/
    search?
    sitesearch=https%3A%2F%2Fexample.com%2F&amp;as_q={sea
    rchTerms}"/>
</OpenSearchDescription>
```

Element	Required	Description
AdultContent	No	True or False to being child-friendly.
Attribution	No	Copyright details for the document.
Contact	No	Email of documents maintainer.
Description	Yes	Description of the search engine.
Developer	No	The creator of the document.
Image	No	ICO (16×16) & PNG (64×64) with width, height and (MIME) type attributes set.
InputEncoding	No	Encoding formats search support.
Language	No	The language of search results.
LongName	No	48 or less character extended title.
OutputEncoding	No	Encoding formats results supports.
Query	No	Search query that can be performed via searchTerms and role attributes.
ShortName	Yes	Title that describes the search engine.
SyndicationRight	No	Stop requesting, showing or sending search results to visitors or others.
Tags	No	Space separated tags to identify data.
Url	Yes	Url for the search. Include parameters like {searchTerms}, {startPage?} and pageOffset (page number) attribute.

Integration

Once you've included any elements you wish to add into your file, we must include a reference to the completed OpenSearch document within the head of our HTML document. This enables browsers to find and utilize the format and add the search engine into the list of choices for visitors. Include it within every page to ensure that browsers add it to their list when visitors arrive at your site.

```
<link rel="search" type="application/opensearchdescription+xml"
href="opensearch.xml">
```

With this small file On your website, you can make your content easier to search. It's tiny files like this that can make a real difference to your user-experience within the web browser. If you don't have a search engine on your site and just have pages filled with content, you could link it to a Google site search (as used in the example earlier) to help visitors find a specific page there after quicker. This may be especially useful if you have thousands of pages of content on your website.

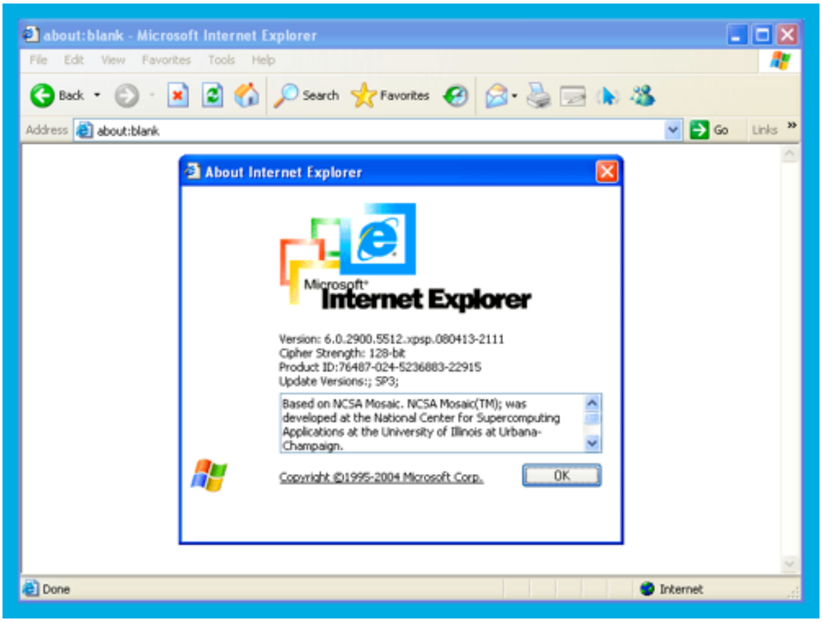
Chapter 21:

Declaring privacy rules for old IE versions

While this standard is obsolete for browsers apart from IE9 or lower, you can still build the privacy file here.

p3p.xml

Internet Explorer, the great gremlin of the Web. Feared by many web developers and causer of many a sleepless night, it certainly gained a rightful reputation over its history as a destroyer of clean markup and a breaker of semantic designs. Of all its many quirks, for this chapter we are going to explore how it (failed to) handle data collection to benefit of user privacy. This comes with a bit of history for Internet fans. It's also one of those standards you may find you don't require.


about:blank - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address about:blank
Go
Links
About Internet Explorer
Microsoft Internet Explorer
OK
Done
Internet

Version: 6.0.2900.5512.xpsp.080413-2111
Cipher Strength: 128-bit
Product ID:76487-024-5236883-22915
Update Versions:; SP3;

Based on NCSA Mosaic. NCSA Mosaic(TM); was developed at the National Center For Supercomputing Applications at the University of Illinois at Urbana-Champaign.

Copyright ©1995-2004 Microsoft Corp.

Reference For a comprehensive guide to P3P, read the specification at https://www.w3.org/TR/P3P/

History

Back before browsers had data handling features baked in, the thought among browser makers was that web developers should be the ones to protect users from bad code through coding best practices. The best example of this gone wrong is the P3P file, a standard devised by the W3C in which developers would control which data that would be harvested for their own good and the thought was this would be enough to stop bad actors from working.

Naturally, this didn't work as expected. Only Microsoft implemented the P3P policy system into their browser, thinking it was a good idea. Additionally, the only people who actually added P3P files to their sites were individuals who cared about privacy to begin with and didn't violate their users' rights (making it literal abandonware). Bad actors ignored P3P and carried on as normal, leaving most visitors to sites with no protection, hence our current situation where users control of their privacy and can block data collection at source.

So far, so historical. Why is this relevant for current users? Well, until Internet Explorer 9, the P3P policy was still actively being implemented within Microsofts browser (despite the low uptake of about 6%). So if you do have an old network you need to maintain such as an embedded system or Intranet and most users are required to use IE9 or lower, the file could be of benefit for reducing data gathering within websites, otherwise, skip this chapter entirely.

Boilerplate

To create a P3P.xml, create a blank document and place it in the base directory of your website. You will want to put all the below code within the file and save it "as is". There's not much worth configuring.

```
<META xmlns="http://www.w3.org/2002/01/P3Pv1">
    <POLICY-REFERENCES>
        <EXPIRY max-age="172800"/>
        <POLICY-REF about="/P3P/Policies.xml#first">
            <INCLUDE>/*</INCLUDE>
            <EXCLUDE>/cgi-bin/*</EXCLUDE>
        </POLICY-REF>
    </POLICY-REFERENCES>
</META>
```

Finally, to complete the process, add a reference to the file in the head of your HTML document so that Internet Explorer can utilize it.

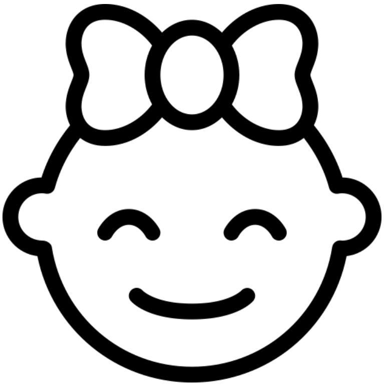
```
<link rel="P3Pv1" href="p3p.xml">
```

Remember that this specification is both obsolete and deprecated, so only use a P3P document if you have a user-base of Internet Explorer 9 or less (as mentioned above). I've included it for historical reasons, as it's a unique and another tiny web format that did for a small-time try (aimlessly in this case) to improve a user's web experience.

Chapter 22:

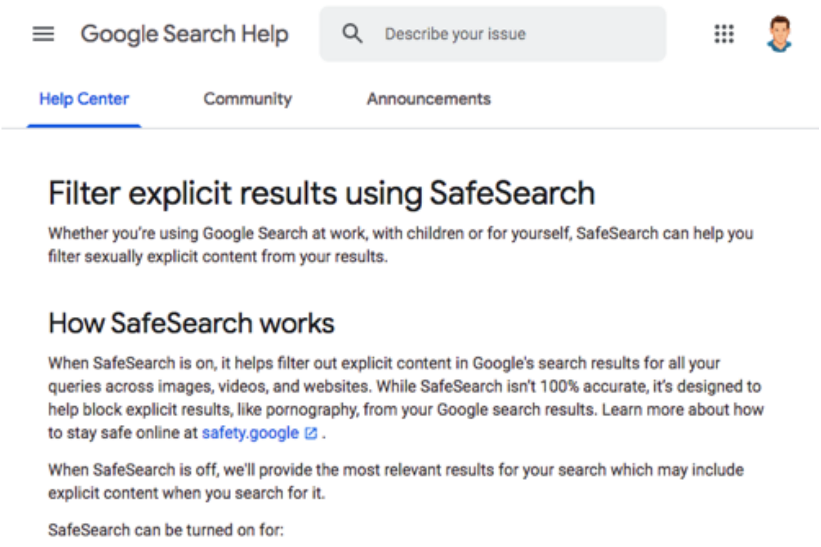
Ensuring your site's declared child-friendly

Are children visiting your site? Add content labelling plus optional support for PICS for older schemes.



powder.xml

Keeping children safe while online is one of the greatest challenges facing anyone who runs any website or app with any kind of social interaction. Critical to this challenge is the correct labelling of content and ensuring that search engines can filter anything that is unsuitable for young children effectively. Using web standards like Powder (and its predecessor PICS) and the digital labelling systems which exist, we can achieve just this on our products to help protect young people.



Filter explicit results using SafeSearch

Whether you're using Google Search at work, with children or for yourself, SafeSearch can help you filter sexually explicit content from your results.

How SafeSearch works

When SafeSearch is on, it helps filter out explicit content in Google's search results for all your queries across images, videos, and websites. While SafeSearch isn't 100% accurate, it's designed to help block explicit results, like pornography, from your Google search results. Learn more about how to stay safe online at safety.google.

When SafeSearch is off, we'll provide the most relevant results for your search which may include explicit content when you search for it.

SafeSearch can be turned on for:

The benefit of these files goes beyond content filtering in search engines. Social networks which may have website content posted can be filtered correctly. Web browsers with parental controls can block inappropriate websites. Parental control software within an operating system, router, or software can know whether your site or sections of it are safe for children to browse. As you can see, without one of these files, parents, and kids will be literally browsing blind.

Reference For a comprehensive guide to Powder, read the specification at https://www.w3.org/TR/2009/NOTE-powder-primer-20090901/

Powder Creation

So let's create a powder.xml file (the modern standard). First, we will need to make the document and place it somewhere in your site's structure. Within the file you'll need an XML doctype, a powder tag with relevant xmlns attributes, and two tags; an attribution element (for the head data) and an ol element (for the body data).

```
<?xml version="1.0" encoding="utf-8?>
<powder xmlns="http://www.w3.org/2007/05/powder#"
xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:rdf="http://
www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:icra="http://
www.icra.org/rdfs/vocabulary2008#">
    <attribution>

    </attribution>
    <ol>

    </ol>
</powder>
```

Attribution

Next, we need to add the elements within the attribution tags. Within the issuedby tag, you should provide a URL about the individual who published the Powder document (perhaps this is where a dublin.rdf file (see Chapter 8) would work well. The issued element will list the date and time the Powder file was published. The certifiedby element will link to a third party that can verify your labelling credentials.

```
<issuedby src="http://authority.example.com/company.rdf#me" />
<issued>2021-10-14T00:00:00</issued>
<certifiedby src="http://independent.example.com?
verify=http%3A%2F%2Fexample.com%2Fpowder.xml" />
```

ICRA Labels

Within the OL tags, we need to provide information about the site that will label it correctly for age appropriateness. Each section of your project will need a dr element. Within that, it will need an iriset element that has an includehosts tag with the domain of your site and an includepathstartswith tag with the path to be labelled. It will also need a descriptorset element which we will add ICRA labels and a description of the content appropriateness using the displaytext element. Sounds like a confusing mess? Check the following example.

```
<dr>
    <iriset>
        <includehosts>example.com</includehosts>
        <includepathstartswith>/path</includepathstartswith>
    </iriset>
    <descriptorset>
        <icra:nz>1</icra:nz>
        <icra:sz>1</icra:sz>
        <icra:vz>1</icra:vz>
        <icra:lz>1</icra:lz>
        <icra:hz>1</icra:hz>
        <icra:dz>1</icra:dz>
        <icra:uz>1</icra:uz>
        <icra:pa>1</icra:pa>
        <displaytext>No nudity; No sexual material; No violence; No
        potentially offensive language; No potentially harmful
        activities; No potentially disturbing material; No user-
        generated content, Contains advertising.</displaytext>
    </descriptorset>
</dr>
```

You'll probably want to replace them with your own (plus adding or removing as appropriate) to match your site's content, providing a description that matches the project's circumstances most accurately.

You see all those ICRA elements with the 1 values within them? That's what describes exactly whether the content is safe for kids or not. Let's examine this further. Using the ICRA label decoder, you browse through the list and add an element with the two required characters preceded by icra: and provide a value matching what the decoder asks for into the list. So for "Mild expletives" you would need to add <icra:lc>1</icra:lc> to descriptorset. You can build a profile using this.

Integration

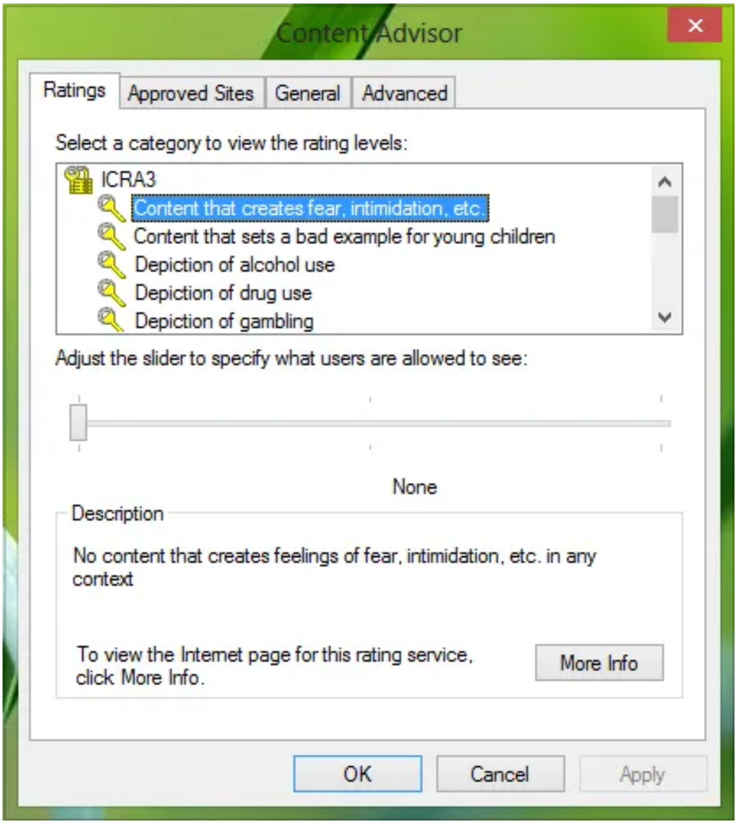
You'll also need to add a link to the Powder file in the head of your HTML file so browsers, search engines, and social networks know that the content on the website may be age appropriate.

```
<link rel="help" type="application/powder+xml"
href="powder.xml">
```

pics.rdf

While superseded and deprecated by Powder, the PICS format is still worth talking about as older web browsers (namely old versions of Internet Explorer 6-8) take advantage of it when using built-in parental control filters upon websites. There may also be cases where some older content advisory websites may look for PICS data rather than Powder data due to a lack of funding to upgrade their legacy systems (consider government IT systems as a prime example).

Reference	The four labelling schemes can be found at http://www.icra.org/ (ICRA), https://web.archive.org/web/20080108223152/http://www.icra.org/decode/ (RSAC), http://www.safesurf.com/ssplan.htm (SafeSurf), and https://web.archive.org/web/20060706165722/http://www.weburbia.com/safe/ (WebUrbia).



Select a category to view the rating levels:

ICRA3
- Content that creates fear, intimidation, etc.
- Content that sets a bad example for young children
- Depiction of alcohol use
- Depiction of drug use
- Depiction of gambling

Adjust the slider to specify what users are allowed to see:

None

Description

No content that creates feelings of fear, intimidation, etc. in any context

To view the Internet page for this rating service, click More Info.

PICS Creation

As such, if you're working with websites that need to be used upon legacy systems that may not be upgraded very frequently, or perhaps are known for having (or are dependent upon using) old versions of Microsoft Internet Explorer; it just might be worth building one of these files in addition to a Powder file. To create a pics.rdf file, you make your blank document in the same location as your Powder file (anywhere on your site) and include an XML doctype with RDF tags.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dcterms="http://purl.org/dc/terms/" xmlns:label="http://
www.w3.org/2004/12/q/contentlabel#" xmlns:icra="http://
www.icra.org/rdfs/vocabularyv03#" xmlns:rsac="http://
www.icra.org/rdfs/vocabularyv01#"  xmlns:ss="http://
www.safesurf.com/ssplan/" xmlns:sfk="http://
www.weburbia.com/safe/ratings/ ">

</rdf:RDF>
```

Attribution

Within the rdf tag, you will need a description element that contains creator tags referencing the three parental labelling authorities we shall use (they're the most well established providers). We also require an issued Dublin tag for when the PICS label was published, and the authorityFor element that references the labels we're utilizing. We also require a ruleset element that attaches your domain name to this file.

```
<rdf:Description rdf:about="">
    <dc:creator rdf:resource="http://www.icra.org" />
    <dc:creator rdf:resource="http://www.safesurf.com" />
    <dc:creator rdf:resource="http://www.weburbia.com/safe" />
    <dcterms:issued>2021-03-13</dcterms:issued>
    <label:authorityFor>http://www.icra.org/rdfs/
    vocabularyv03#</label:authorityFor>
</rdf:Description>
<label:Ruleset>
    <label:hasHostRestrictions><label:Hosts><label:hostRestriction>e
    xample.com</label:hostRestriction></label:Hosts></
    label:hasHostRestrictions>
    <label:hasDefaultLabel rdf:resource="#label_1" />
</label:Ruleset>
```

Labelling Schemes

Next, within the rdf tags, we need to include content labels (just like we did in the Powder file). You'll notice that I provide content not from one but from four different authorities (ICRA, RSAC, SafeSurf & WebUrbia) as back when PICS was the standard, no labelling system held dominance, and it was best to cover all of your bases to ensure compatibility. A basic example using all four is shown below.

```
<label:ContentLabel rdf:ID="label_1">
    <rdfs:comment>ICRA Ratings</rdfs:comment>
    <icra:nz>1</icra:nz>
    <icra:sz>1</icra:sz>
    <icra:vz>1</icra:vz>
    <icra:lz>1</icra:lz>
    <icra:oz>1</icra:oz>
    <icra:cz>1</icra:cz>
    <icra:xz>1</icra:xz>
</label:ContentLabel>
<label:ContentLabel rdf:ID="label_2">
    <rdfs:comment>RSAC Ratings</rdfs:comment>
    <rsac:l>0</rsac:l>
    <rsac:n>0</rsac:n>
    <rsac:s>0</rsac:s>
    <rsac:v>0</rsac:v>
</label:ContentLabel>
<label:ContentLabel rdf:ID="label_3">
    <rdfs:comment>SafeSurf Ratings</rdfs:comment>
    <ss:ss000>1</ss:ss000>
</label:ContentLabel>
<label:ContentLabel rdf:ID="label_4">
    <rdfs:comment>WebUrbia Ratings</rdfs:comment>
    <SFK:S>0</SFK:S>
</label:ContentLabel>
```

SafeSurf®
Original Internet Rating Standard

The SafeSurf Internet Rating Standard

Designed by and for parents to empower each family to make informed decisions concerning accessibility of online content.
© 1995-2017 SafeSurf™ Organization. All Rights Reserved.

PICS Version 3.0

The following is SafeSurf's System to protect children on the Internet.
It involves rating, classification, filtering and parental education.
It employ's the SafeSurf Wave™ , SS~~™ , as a certification mark.

Summary

Once you've marked up your content labels from each authority, you will want to include a reference to your finished PICS file within the head of your HTML document (like with the Powder file) so that any parental guidance software or search engine can detect the format.

```
<link rel="help" type="application/rdf+xml" href="pics.rdf">
```

As stated earlier, unless you're supporting older browsers and archaic systems, the PICS file shouldn't (thankfully) be required to get your site child-friendly. Having to support numerous ratings labels was a bit of a drag. It's worth having a powder.xml file if you have any content that serves a younger audience, or you want to ensure that kids aren't going to accidentally browse there with parental tools switched on.

Reference For a comprehensive guide to PICS, read the specification at https://www.w3.org/PICS/

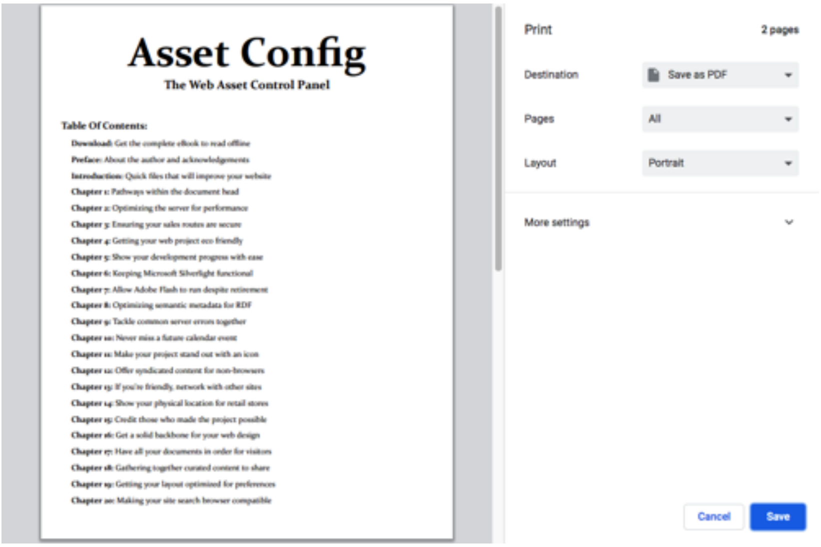
Chapter 23:

Reduce the impact of pages on printers

Set up a default stylesheet for when your website is being printed or exported to a document like PDF.

print.css

Once upon a time, the world existed to be printed. Loads of pages of content were printed onto pieces of paper regularly. These days, with the age of the Internet, it's not as common due to the digitization and connectivity of data (being that we're always online). However, don't be fooled into thinking that printing has entirely gone away. People sometimes choose to print data for good reasons. Tickets, posters, leaflets, device-free zones, images, and more being prime examples.



Asset Config

The Web Asset Control Panel

Table Of Contents:

Print 2 pages

Destination: Save as PDF

Pages: All

Layout: Portrait

Ensuring that your website works on the printed page still makes sense, even in the post paper era. Covid-19 accelerated the decline of printing due to people not wanting to pass around as many physical documents, and ink costs being more costly than fine champagne. Though, personal paper items and printing for PDF export to keep a copy offline remain just as popular as they've always been.

Reference For a comprehensive guide to how CSS handles media types, read the specification at https://www.w3.org/TR/css3-mediaqueries/ or read the guide at https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

You can create a simple printer friendly document using the print CSS stylesheet (@media print), and customize your document using the Google Chrome print preview window to see how the finished product will look. Or if you want something quicker and easier, you can use a pre-built library. Want one? I've built a library called Printed and alongside some essential components, it has some useful extras.

Reference For the latest updates to my print stylesheet and to get the latest features, visit https://printedcss.com/

CSS Reset

To get started, create a print.css file or open up an existing stylesheet and insert a @media print reference with the below sample code. It will provide a CSS reset which tweaks every HTML element to behave for printers. It also includes a websafe typeface that should look good on every operating system, plus it will print citations for quotes, list any abbreviations, show data values, display timing events, and highlight keyboard characters when they're mentioned in documents.

```
@media print {
    * { margin: 0; padding: 0; }
    body { background-color: #ffffff; color: #000000; font-family:
    Constantia, Palatino, 'Book Antiqua', 'Palatino Linotype', serif;
    line-height: 1.5; }
    blockquote code { border: 0; }
    blockquote, code, kbd, pre { border: 1px solid grey; }
    table, td, th { border-bottom: 1px solid black; }
    abbr[title], acronym[title] { border-bottom: 1px dotted grey; }
    table { border-collapse: collapse; }
    aside { border-left: 2px solid grey; padding-left: 1em; }
    del, s { color: grey; }
    a, canvas, img { color: inherit; }
    blockquote:after { content: attr(cite); }
    dt:after { content: ":"; }
    abbr:after, acronym:after { content: " (" attr(title) ")"; }
    data:after { content: " (" attr(value) ")"; }
    time:after { content: " (" attr(datetime) ")"; }
    q:after { content: "\201D" " (" attr(cite) ")"; }
    canvas, html:after, img, picture { display: block; }
    applet, audio, blink, dialog, embed, form, iframe, map, map +
    img, marquee, menu, nav, object, video { display: none; }
    code, kbd, pre, samp, tt, var { font-family: 'Courier New',
    Courier, 'Lucida Sans Typewriter', 'Lucida Typewriter',
    monospace; font-size: 0.8em; }
    h1 { font-size: 3rem; }
    h2 { font-size: 2rem; }
    h3 { font-size: 1.75rem; }
    h4 { font-size: 1.5rem; }
    h5 { font-size: 1.25rem; }
    h6 { font-size: 1rem; }
    blockquote:after, q:after { font-size: 0.75em; }
    caption, figcaption { font-style: italic; }
    dt { font-weight: bold; }
    pre { height: auto; hyphens: auto; overflow: auto; }
    h1, h2, h3, h4, h5, h6 { margin: 1rem 0; page-break-after: avoid; }
    canvas, img, svg { margin: 1em auto; }
    address, blockquote, details, dl, figure, ol, p, pre, ruby, table, ul {
    margin: 1em 0; }
    dd, ul, ol { margin-left: 2em; }
    blockquote p { margin-top: 0; }
    img { border: 0; max-width: 50vw; }
    p, pre, table { orphans: 4; widows: 4; }
    td, th { padding: 0.1em 0.2em; }
    code, kbd { padding: 0.2em 0.3em; }
    blockquote, pre { padding: 0.5em 1em; }
    html { padding: 2em 4em; }
    article { page-break-before: always; }
    a, blockquote, canvas, details, dl, figure, img, ol, picture, svg,
    table, ul { break-inside: avoid; page-break-inside: avoid; }
    canvas, figure, h1, html:after, img, picture, svg, table { text-align:
    center; }
    a, abbr, acronym, ins { text-decoration: none; }
    u { text-decoration-style: wavy; }
    table { width: 100%; }
    a, blockquote:after, pre, q:after { word-wrap: break-word; }
}
```

Minimal Theme

The next piece of code is entirely optional, known as a flag. If you want a minimal layout which uses even less paper than the default one that is provided in the reset, you can set data-theme="print-min" to the HTML body element. It works alongside the standard theme.

```
html[data-theme=print-min] mark { background-color:
transparent; }
html[data-theme=print-min] abbr, html[data-theme=print-min]
acronym, html[data-theme=print-min] aside, html[data-
theme=print-min] blockquote, html[data-theme=print-min] code,
html[data-theme=print-min] kbd, html[data-theme=print-min] pre,
html[data-theme=print-min] table, html[data-theme=print-min] td,
html[data-theme=print-min] th { border: 0; padding: 0; }
html[data-theme=print-min] a:not(:local-link):after { content: ""; }
html[data-theme=print-min] a[href^="http"]:after, html[data-
theme=print-min] a[href^="ftp"]:after { content: ""; }
html[data-theme=print-min] blockquote:after, html[data-
theme=print-min] q:after { content: ""; }
html[data-theme=print-min] canvas, html[data-theme=print-min]
img, html[data-theme=print-min] svg { display: none; }
html[data-theme=print-min] h1 { font-size: 2rem; }
html[data-theme=print-min] h2 { font-size: 1.8rem; }
html[data-theme=print-min] h3 { font-size: 1.6rem; }
html[data-theme=print-min] h4 { font-size: 1.4rem; }
html[data-theme=print-min] h5 { font-size: 1.2rem; }
html[data-theme=print-min] h6 { font-size: 1rem; }
html[data-theme=print-min] h1, html[data-theme=print-min] h2,
html[data-theme=print-min] h3, html[data-theme=print-min] h4,
html[data-theme=print-min] h5, html[data-theme=print-min] h6
{ margin: 0.5rem 0; }
html[data-theme=print-min] body { line-height: 1.2; }
html[data-theme=print-min] { padding: 1em 2em; }
```

File Types

Occasionally, when we go to print a file, we see links on sites but have no idea about what application these files might be linked with, or in many mime types, associated with. The below aims to solve the problem by covering a wide range of formats with a description, and also provide all hyperlink types with a full-length URL by the link text.

```
$a:after > img { content: ""; }
a[href^="#"]:after, a[href^="javascript:"]:after { content: ""; }
a:not(:local-link):after { content:" <" attr(href) "> "; }
a[href^="http"]:after, a[href^="ftp"]:after { content: ": " attr(href); }
a[href^="mailto"]:before, a[href^="message"]:before { content:
"Email: "; }
a[href^="maps"]:before { content: "Maps: "; }
a[href^="webcal"]:before { content: "Calendar: "; }
a[href^="tel"]:before, a[href^="facetime"]:before, a[href^="facetime-
audio"]:before, a[href^="sms"]:before, a[href^="irc"]:before,
a[href^="skype"]:before, a[href^="whatsapp"]:before, a[href^="fb-
messenger"]:before, a[href^="signal"]:before { content: "Contact: "; }
a[href^="twitter"]:before, a[href^="fb"]:before,
a[href^="snapchat"]:before, a[href^="instagram"]:before { content:
"Social: "; }
a[href^="spotify"]:before, a[href^="music"]:before,
a[href^="feed"]:before, a[href^="podcast"]:before { content:
"Media: "; }
a[href$=".zip"]:after, a[href$=".rar"]:after, a[href$=".7z"]:after
{ content: " (Archive)"; }
a[href$=".mp3"]:after, a[href$=".m4a"]:after, a[href$=".mp4"]:after,
a[href$=".aac"]:after, a[href$=".flac"]:after, a[href$=".ogg"]:after,
a[href$=".oga"]:after, a[href$=".opus"]:after { content: " (Music)"; }
a[href$=".mkv"]:after, a[href$=".mp4"]:after, a[href$=".mpg"]:after,
a[href$=".mpeg"]:after, a[href$=".hevc"]:after, a[href$=".mov"]:after,
a[href$=".webm"]:after, a[href$=".avi"]:after, a[href$=".ogv"]:after
{ content: " (Video)"; }
```

```
a[href$=".svg"]:after, a[href$=".png"]:after, a[href$=".webp"]:after,
a[href$=".jpg"]:after, a[href$=".tiff"]:after, a[href$=".gif"]:after
{ content: " (Image)"; }
a[href$=".pdf"]:after, a[href$=".epub"]:after, a[href$=".mobi"]:after
{ content: " (eBook)"; }
a[href$=".pages"]:after, a[href$=".doc"]:after, a[href$=".docx"]:after,
a[href$=".odt"]:after, a[href$=".rtf"]:after, a[href$=".txt"]:after
{ content: " (Document)"; }
a[href$=".numbers"]:after, a[href$=".xls"]:after, a[href$=".xlsx"]:after,
a[href$=".ods"]:after { content: " (Spreadsheet)"; }
a[href$=".key"]:after, a[href$=".ppt"]:after, a[href$=".pptx"]:after,
a[href$=".odp"]:after { content: " (Presentation)"; }
a[href$=".woff"]:after, a[href$=".woff2"]:after, a[href$=".eot"]:after,
a[href$=".otf"]:after, a[href$=".ttf"]:after { content: " (Typeface)"; }
```

Hide Content

Next, we can hide and show content. This is important for saving ink and paper, which is critical for the eco-friendly among you, and for saving your visitors money. You'll want to use this on any adverts, navigation menus, dynamic content or anything else that can't be used or interacted with on a piece of paper. Remember that we're dealing with a printed file, so clicking will have no effect.

```
.hide { display: none; }
.show { display: initial; visibility: initial; }
```

Cover Page

The next CSS class is pretty useful if you want to add a cover image or page to your printed documents. Just include it in the final element.

```
.cover { page-break-after: always; }
```

Page Break

Next, if you would like to insert a page break, you can include the below class in your document within the element you want it to appear before. This will help you avoid overflowing paragraphs.

```
.page-break { page-break-before: always; }
```

Dark Paper

One feature that not many print frameworks cover or consider is the case for if people use paper colors other than white, which is a real misstep. So, I include the below attribute selector that you can utilize on the HTML body (data-paper="dark") to invert paper colors.

```
html[data-paper=dark] body, html[data-paper=dark] del, html[data-paper=dark] hr, html[data-paper=dark] mark, html[data-paper=dark] s { background: initial; color: #ffffff; }
html[data-paper=dark] hr { background-color: #ffffff; color: #ffffff; height: 2px; }
html[data-paper=dark] blockquote, html[data-paper=dark] code, html[data-paper=dark] hr, html[data-paper=dark] kbd, html[data-paper=dark] pre { border: 0; }
html[data-paper=dark] abbr, html[data-paper=dark] acronym, html[data-paper=dark] table, html[data-paper=dark] td, html[data-paper=dark] th { border-bottom-color: #ffffff; }
html[data-paper=dark] aside { border-left-color: #ffffff; }Color Matching
```

Another useful feature you can add is color matching, so that if you wish to have accurate visuals when you print this CSS will ensure that the printer doesn't simply "best guess" and produces better results.

```
html[data-color=exact] body { print-color-adjust: exact; }
```

Filter Effects

One of the great things about CSS3 is you can do some interesting effects on images. They may not be Instagram level, but they'll do in a pinch for basic level filtering. The classes below should work well.

```
.blur { filter: blur(10px); -webkit-filter: blur(10px); }
.brightness { filter: brightness(100%); -webkit-filter: brightness(100%); }
.contrast { filter: contrast(100%); -webkit-filter: contrast(100%); }
.hue-90 { filter: hue-rotate(90deg); -webkit-filter: hue-rotate(90deg); }
.hue-180 { filter: hue-rotate(180deg); -webkit-filter: hue-rotate(180deg); }
.hue-270 { filter: hue-rotate(270deg); -webkit-filter: hue-rotate(270deg); }
.invert { filter: invert(100%); -webkit-filter: invert(100%); }
.monochrome { filter: grayscale(100%); -webkit-filter: grayscale(100%); }
.saturate { filter: saturate(100%); -webkit-filter: saturate(100%); }
.sepia { filter: sepia(100%); -webkit-filter: sepia(100%); }
```

Image Print

Next, there are attributes for data-img="full-page" to scale an image to the full page of a piece of paper, data-img="gallery" to show a list of only images on the paper, and data-img="individual" to show a single image on the paper at its original size. You attach the attribute to a canvas, img, or svg element for it to work it's magic on a page.

```
html[data-img] address, html[data-img] blockquote, html[data-img] details, html[data-img] dl, html[data-img] figcaption, html[data-img] h1, html[data-img] h2, html[data-img] h3, html[data-img] h4, html[data-img] h5, html[data-img] h6, html[data-img] p, html[data-img] hr, html[data-img] map, html[data-img] map + img, html[data-img] ol, html[data-img] pre, html[data-img] ruby, html[data-img] table, html[data-img] ul, html:after { display: none; }
.full-page { display: block !important; height: 100vh; max-width: initial; width: 100vw; }
html[data-img=full-page] canvas, html[data-img=full-page] img, html[data-img=full-page] svg { display: none; }
html[data-img] figure, .full-page { margin: 0; }
html[data-img=full-page] { padding: 0; }
html[data-img=gallery] { padding: 0.5em 1em; }
html[data-img=gallery] canvas, html[data-img=gallery] img, html[data-img=gallery] svg { float: left; margin: 0; }
html[data-img=individual] { align-items: center; display: flex; height: 100vh; justify-content: center; }
.individual { display: block !important; }
html[data-img=individual] canvas, html[data-img=individual] img, html[data-img=individual] svg { display: none; }
html[data-img=individual], html[data-img=individual] canvas, html[data-img=individual] img, html[data-img=individual] svg
{ margin: 0; padding: 0; }
```

Paper Size

Additionally, we can set the paper size. This requires us to put the below within the @media print, like with the rest of the above code.

```
.paper-size { box-sizing: border-box; margin: 0; overflow: hidden;
padding: 0; page-break-after: always; position: relative; }
```

We also need to add support for a data-paper attribute, and it's many possible values within the HTML element (using the orientation media query). Below, I've included basic support for 10 basic paper sizes!

```
@media (orientation: portrait) {
    html[data-paper=ledger] .paper-size { height: 17in; width: 11in; }
    html[data-paper=legal] .paper-size { height: 14in; width: 8.5in; }
    html[data-paper=letter] .paper-size { height: 11in; width: 8.5in; }
    html[data-paper=A3] .paper-size { height: 420mm; width:
    297mm; }
    html[data-paper=A4] .paper-size { height: 297mm; width:
    210mm; }
    html[data-paper=A5] .paper-size { height: 210mm; width:
    148mm; }
    html[data-paper=JIS-B4] .paper-size { height: 364mm; width:
    257mm; }
    html[data-paper=B4] .paper-size { height: 353mm; width:
    250mm; }
    html[data-paper=JIS-B5] .paper-size { height: 257mm; width:
    182mm; }
    html[data-paper=B5] .paper-size { height: 250mm; width:
    176mm; }
}
@media (orientation: landscape) {
    html[data-paper=ledger] .paper-size { height: 11in; width: 17in; }
    html[data-paper=legal] .paper-size { height: 8.5in; width: 14in; }
    html[data-paper=letter] .paper-size { height: 8.5in; width: 11in; }
    html[data-paper=A3] .paper-size { height: 297mm; width:
    420mm; }
```

```
        html[data-paper=A4] .paper-size { height: 210mm; width:
        297mm; }
        html[data-paper=A5] .paper-size { height: 148mm; width:
        210mm; }
        html[data-paper=JIS-B4] .paper-size { height: 257mm; width:
        364mm; }
        html[data-paper=B4] .paper-size { height: 250mm; width:
        353mm; }
        html[data-paper=JIS-B5] .paper-size { height: 182mm; width:
        257mm; }
        html[data-paper=B5] .paper-size { height: 176mm; width:
        250mm; }
    }
```

Multi-Column Layout

Next, for visitors browsing the website in landscape, we will also include support for multi-column layouts using the data-col attribute with two, three, and four as possible values for content overflow.

```
    @media (orientation: landscape) {
        html[data-col=two] .column { -webkit-column-count: 2; -moz-
        column-count: 2; column-count: 2; }
        html[data-col=three] .column { -webkit-column-count: 3; -moz-
        column-count: 3; column-count: 3; }
        html[data-col=four] .column { -webkit-column-count: 4; -moz-
        column-count: 4; column-count: 4; }
    }
```

Color Handling

Finally, we can round off the chapter by adding in true color handling and a monochrome CSS helper, which will grey-style images to match a printed output. While you don't need to include all these optional pieces of code within your CSS syntax, hopefully some of these will provide you with some ideas to improve the overall print experience.

```
@media print and (color) {
    * { print-color-adjust: exact; -webkit-print-color-adjust: exact; }
}
@media print and (monochrome) {
    canvas, figure, img, picture, svg { filter: grayscale(100%);
    -webkit-filter: grayscale(100%); }
    mark { background-color: #D3D3D3 !important; print-color-
    adjust: exact; -webkit-print-color-adjust: exact; }
}
```

Reference For further on creating print stylesheets, I recommend reading https://alistapart.com/article/goingtoprint/ and https://www.smashingmagazine.com/2018/05/print-stylesheets-in-2018/ and https://www.matuzo.at/blog/i-totally-forgot-about-print-style-sheets/

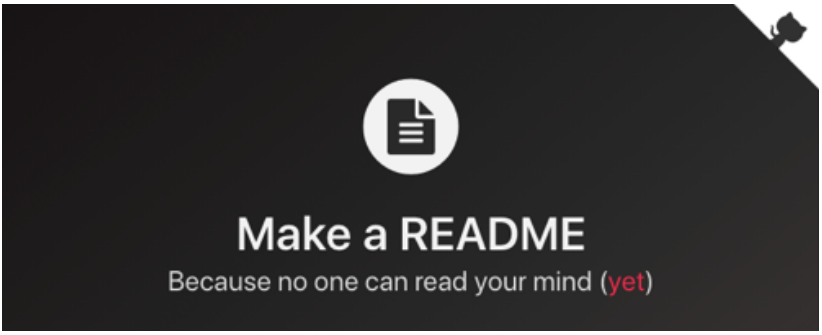
Chapter 24:

Keep your development team up-to-date

Create a README for developers working on production files, or projects, critical to the website.

README

Whether you're a regular on GitHub or a code hoarder who happens to work with several other developers contributing to the same projects (on another collaborative platform), there has never been a more important time to ensure that anyone who is involved in the building of a website or app can dive into a project as quickly as possible. The README aims to be the ultimate signposting tool by acting as a universal manual for developers who dare get involved.



Every developer has their method of creating a README file and there is no wrong way of producing one, so I've decided to share mine. The README is one of those files that encourages coders to work toward similar standards, helps reduce mistakes, and can help users who are curious about open source work not run for the hills. The benefit is there, even if it mostly occurs behind the scenes.

Reference No perfect README exists, but mine is based upon this as it's as close to a standard as one exists. https://www.makeareadme.com/ Also checkout https://github.com/hackergrrl/art-of-readme

Generation

So let's get started by creating a README file. It has no extension and should be placed on your GitHub (or kept where your versioning occurs. Don't just FTP it to your server and leave it for the public, as they won't be able to use it. Within the file, you will want to add some information about what your website or application. You should be as detailed as you can be, giving steps, images, tables and such as is required. To begin our file, we should provide a title for our README.

```
# README - <Text>
```

Overview

Next we need an overview section. Within the overview, you should give a quick rundown of what the project is about. I usually provide things like a project name (including codenames, versions, and URL's), the background (history, timeline, version releases, timeline) of the project, a detailed description (purpose, can/can't do, aims), and any badges (code maintenance, awards, certification) that are related.

```
Overview:
Name: <Text>
Background: <Text>
Description: <Text>
Badges: <List>
```

Usage

Next, you'll want a usage section. It'll serve as the basis for a project's documentation. I provide a list of features (guide, workflow, code, CLI tools), a FileMap (repo structure, syntax used, apps required), multiple visuals (screenshots, videos, podcasts, etc), and a reading list (useful links, books, Q&A's, docs) to explain how to include the project.

```
Usage:
Features: <List>
FileMap: <List>
Visuals: <Images>
Reading: <List>
```

Installation

Next is something no guide can be without, an installation section. This helps the user get the project running. I usually give download (import, clone, and sign-up) instructions, requirements (browser, application, device) to use the project, and dependancies (libraries or frameworks) needed for the project to be utilized on a website.

```
Installation:
Download: <Url>
Requirements: <List>
Dependancies: <List>
```

Support

Finally, we have the support section. This works independently of the documentation by giving developers further advice about a project. You could offer details about the project's status (activity, live/dead, deprecated), details on contributing (links, how-to, issues, translations, discussions), a legal file (license, terms, privacy, copyright), project acknowledgements (contributors, sponsors, etc), the date last updated (major, minor & revision) and authors (project, website, etc).

```
Support:
Status: <Text>
Contributing: <Url>
License: <Url>
Acknowledgments: <List>
Updated: <Date>
Authors: <List>
```

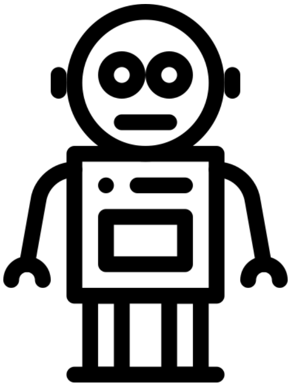
Summary

Ensuring that visitors to your codebase feel comfortable without having to reach out for support is critical in being able to collaborate. By having a README file, you can provide answers to some basic questions about the project and direct people to further info if further reading is required. It should reduce the amount of technical support requests you're handling and let you code collaboratively.

Chapter 25:

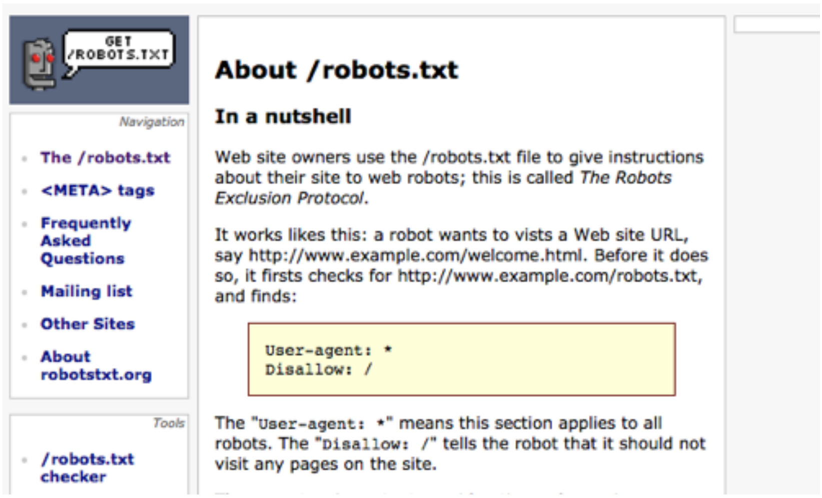
Setting the default rules for search engines

Generate a robots text file using the extended standard to ensure search engines can find all your pages easily.



robots.txt

Just like many of the text files we've covered in previous chapters, the robots file serves a unique and important purpose. What you may not know (if you're a young developer) is that the robot's file is one of the oldest web standards (dating back to the early 90s). The aim of the robots exclusion standard is to tell search engines which pages you would rather not appear in search results pages. It's really that simple.



About /robots.txt

In a nutshell

Web site owners use the /robots.txt file to give instructions about their site to web robots; this is called *The Robots Exclusion Protocol*.

It works likes this: a robot wants to vists a Web site URL, say http://www.example.com/welcome.html. Before it does so, it firsts checks for http://www.example.com/robots.txt, and finds:

```
User-agent: *
Disallow: /
```

The "`User-agent: *`" means this section applies to all robots. The "`Disallow: /`" tells the robot that it should not visit any pages on the site.

The standard has evolved from the original specification (which hasn't changed) to an extended standard, that has been partially adopted and expanded upon by search engines via the tiny but extremely useful text file. It's therefore highly recommended that every site has one of these files to control how search engines index their content.

Generation

So how do we create this useful file? Simple. Create a plaintext file called robots.txt and place it in the base directory of your website. Just like a favicon.ico, your robots file must be located in the base of your website to ensure that when search engines go looking for the rules, they are in the default location. Be aware that, like any file, any tool browsing your site can choose to ignore your robots.txt ruleset.

Comments

As with many text files, you can include comments in your robots file by preceding your text with a hash (#) character. You can do this on a new line (perhaps giving your file a title) or directly after a property - value pair. Below, I've provided a basic example of a comment.

```
# This is a comment
```

Robots Standard

Next, you'll need to put some content within the robots file for search engines to follow. There are only a few properties you can use, as the specification itself (that Google understands) is small. First up we have the User-agent property which, unless you have a reason to select something else, I'd keep it to the below value. It essentially lets you decide which search engines the robot rules below it will apply too.

```
User-agent: *
```

The next rule we can use the Disallow property, which tells the search engine to ignore the chosen folder. If User-agent is star valued, then it applies to all robots, if not, you can pick a search engine for it to apply too. The Disallow property shouldn't be used in-place of password protection either, as while it won't show in search results, visitors could potentially find the page through the robots.txt file instead.

```
Disallow: /cgi-bin/
```

Along with Disallow, you can specify a folder that must be included within search engine results (where no direct path to the document has been provided within the website or apps hierarchy) using the Allow property. You can even set Allow paths within Disallow paths.

```
Allow: /.well-known/
```

Finally, if your website has a sitemap which complies with the sitemap specification (See Chapter 28), use the below opportunity to link to it within this file so that search engines can index your pages whilst they are working out what documents should be left out of search results.

```
Sitemap: https://example.com/sitemap.xml
```

Reference To learn more about what robots properties Google has adopted, read https://developers.google.com/search/docs/advanced/robots/robots_txt

Robots Non-Standard

In addition to the standard robots properties, there are a few others which the extended standard offers, which have limited but potential support among search engines. Whilst I don't tend to use them, their use shouldn't have any impact upon your website or app if they aren't supported by a particular spider or robot which stumbles upon them.

Property	Description
Comment	Message to send to the search engine.
Request-rate	The rate a search engine should use by default.
Robot-version	Follows User-agent. Provides one X.X version.
Visit-time	Robot can only visit between given times.

Reference To learn more about the extended specification, read http://www.conman.org/people/spc/robots2.html

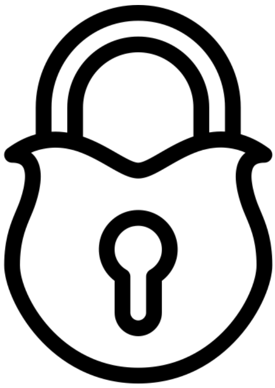
And that's all there is to the robots.txt standard. It's a basic specification which has an important role to play in the visibility of your website or app. It will definitely impact your website's search engine optimization, and it will likely also impact content visibility. As it takes so little time to implement, there is no reason why it shouldn't be used on every website or web application that you produce.

Reference To learn more about robots.txt, read the specification at http://www.robotstxt.org/robotstxt.html

Chapter 26:

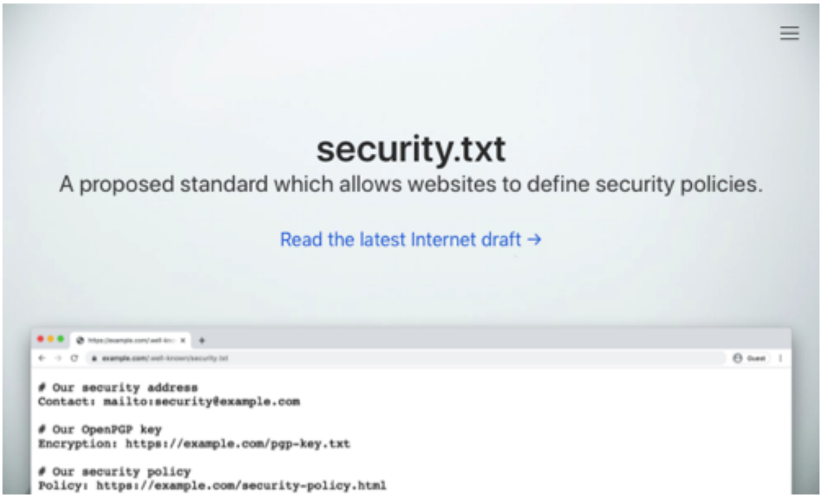
Declare security protocols and contacts

Show you mean business by giving your security related website details, and (if required) a do not track policy.



security.txt

Creating a flawless website is an ongoing process, and along with the spelling errors and visual bugs your website is likely to encounter throughout development, more serious issues such as security flaws or vulnerabilities are a coder's worst nightmare. Not only could they leave you in a legal minefield, they could expose your customers' personal information to fraudsters. Avoiding such issues is paramount.


security.txt

A proposed standard which allows websites to define security policies.

```
# Our security address
Contact: mailto:security@example.com

# Our OpenPGP key
Encryption: https://example.com/pgp-key.txt

# Our security policy
Policy: https://example.com/security-policy.html
```

The aim of a security file isn't to prevent such attacks as dealing with issues like cross-site scripting, hacking, malware and such will be for another book to handle as there are too many mechanisms to cover.

What this chapter and file covers is a simple document that will help security researchers and individuals with a keen eye for problems to direct those faults to the right individuals on your team. As this file is a proposed standard, it's worth adopting within your workflow.

Reference To learn more about the security.txt specification and generate a document, visit https://securitytxt.org/

Policy Creation

This file is intended to work in association with your existing security workflow, so with this in mind, let's build one to add to your site for visitors who might seek the file out if they discover a flaw. To begin, as with any plaintext file, you'll want to create a `security.txt` file, though in this case, you won't place it within the base directory. Instead, its recommended placement is within the `.well-known` asset directory.

Comments

As with the majority of text files, you can include comments in your security file by preceding your text with a hash (#) character. As the file is pretty short, there's unlikely to be many uses for them, thought.

```
# This is a comment
```

Acknowledgments

This should provide a link to a page where security researchers are given credit for their work uncovering any issues on your projects. Remember not to give too much information about the vulnerability to prevent further attacks from occurring if similar issues exist or are uncovered in other regions of your codebase in the future.

```
Acknowledgments: https://example.com/security.html
```

Canonical

This should point to the URL of the security.txt file. It's self-referential and exists so that digital signatures can verify the file's contents easily. You may also find that the link is used as a quick reference for users.

```
Canonical: https://example.com/.well-known/security.txt
```

Contact

This property must be included within the file. You may choose to use a link, a telephone number, an email address, a social media address or some other method of contact. The critical thing is that some easy form of contact and reporting of security issues is available at the URL.

```
Contact: https://example.com/contact.html
```

Encryption

Whether you decide to provide a fingerprint (hash) or a URL to a key, it's important that when signing a security.txt file, you must include this property to ensure that the credentials held within can be verified.

```
Encryption: https://example.com/pgp-key.txt
```

Expires

This mandatory property should contain the date and time of which the security information will become stale (and will need to be next reviewed). You should follow the datetime format below, and it's also recommended by the specification that the value be less than a year.

```
Expires: Thu, 31 Dec 2021 18:37:07 -0800
```

Hiring

If you have any security related jobs going on your site, a great way to let researchers know about them is to advertise them where they will be looking. Your security.txt file has a property for just such a URL.

```
Hiring: https://example.com/jobs.html
```

Policy

Every organization is different and their approach to dealing with any security issues will vary. If you want to declare how you handle bugs, glitches, and vulnerabilities on your projects, provide a Policy link in your file. Try to be as detailed as possible regarding submissions.

```
Policy: https://example.com/policy.html
```

Preferred Languages

This comma separated property is pretty straightforward as it allows you to include two digit languages which you allow security and bug reports to be submitted in. This can be helpful if you only understand English or if you want your reports in a language other than English.

```
Preferred-Languages: en
```

Digital Signature

It is recommended that a security.txt file is digitally signed using OpenPGP signatures, include the canonical property, and have a link to the PGP key using the Encryption property to ensure that the signature is valid. Ensuring that the file is signed correctly is the researcher's responsibility, so make sure you or your team verify files.

OpenPGP

Email encryption. For all operating systems. Standing the test of time.

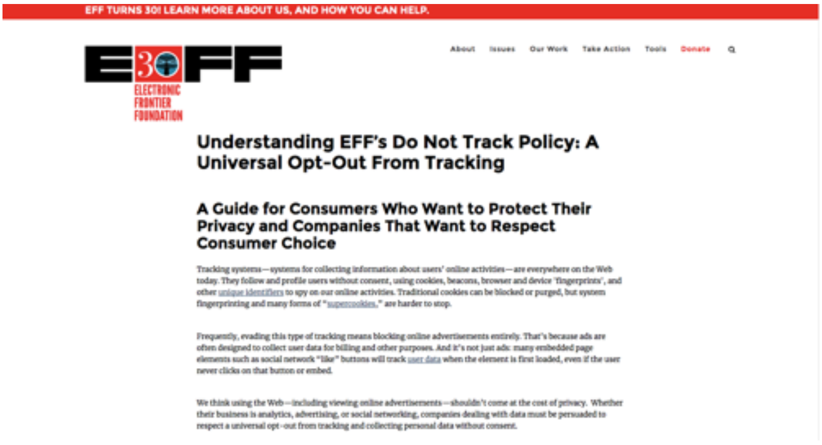
OpenPGP is the most widely used email encryption standard. It is defined by the OpenPGP Working Group of the Internet Engineering Task Force (IETF) as a Proposed Standard in RFC 4880. OpenPGP was originally derived from the PGP software, created by Phil Zimmermann.

I have provided an example of a signed document in action (without the signature, as one needs to be generated). You'll notice the PGP tags with a Hash property (disclosing the hash type), and a Version property within the signature. Open Source, Free and paid PGP software can be found on all platforms to create such signatures.

```
----BEGIN PGP SIGNED MESSAGE----
Hash: SHA256
Acknowledgments: https://example.com/security.html
Canonical: https://example.com/.well-known/security.txt
Contact: https://example.com/contact.html
Encryption: https://example.com/pgp-key.txt
Expires: Thu, 31 Dec 2021 18:37:07 -0800
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.2
[signature]
-----END PGP SIGNATURE----
```

dnt-policy.txt

Another file which should be placed in the .well-known directory which affects the visitors' security, and thus it's worth talking about quickly here, is the little known about dnt-policy.txt file. The idea behind the file is a three part commitment (terms of use agreement) to not track your visitors and respect their privacy. The basics would be no analytics, no cookies, no forms requesting any personal data (etc), but beyond that, we also need to disable server logging.



Understanding EFF's Do Not Track Policy: A Universal Opt-Out From Tracking

A Guide for Consumers Who Want to Protect Their Privacy and Companies That Want to Respect Consumer Choice

Tracking systems—systems for collecting information about users' online activities—are everywhere on the Web today. They follow and profile users without consent, using cookies, beacons, browser and device 'fingerprints', and other unique identifiers to spy on our online activities. Traditional cookies can be blocked or purged, but system fingerprinting and many forms of "supercookies," are harder to stop.

Frequently, evading this type of tracking means blocking online advertisements entirely. That's because ads are often designed to collect user data for billing and other purposes. And it's not just ads: many embedded page elements such as social network "like" buttons will track user data when the element is first loaded, even if the user never clicks on that button or embed.

We think using the Web—including viewing online advertisements—shouldn't come at the cost of privacy. Whether their business is analytics, advertising, or social networking, companies dealing with data must be persuaded to respect a universal opt-out from tracking and collecting personal data without consent.

This proposal by the Electronic Frontier Foundation (EFF) is quite an ask of webmasters who have spent years gouging visitors for data, but if you want to commit to promoting an ethical, data snooping free experience for visitors, you'll need to do three things to make the most of this tiny text file (this is one for real privacy evangelists).

Reference

To read more about the DNT policy from the EFF, visit https://www.eff.org/dnt-policy or to include the policy in your site, use: https://raw.githubusercontent.com/EFForg/dnt-policy/master/dnt-policy-1.0.txt

Creating The File

First is the obvious, if you collect user data (outside the necessary) - stop. Disable all that terrible data hungry script nonsense. Secondly, you'll need to add some code into your server to stop it sucking data by default. On Apache, I've provided the code you'll require below, but if you use another server (like NginX) you can look to your server's documentation for advice on how to turn off session and IP logging.

```
SetEnvIfNoCase DNT 1 DO_NOT_TRACK
CustomLog ${APACHE_LOG_DIR}/access.log combined env=!
DO_NOT_TRACK
```

Location: /etc/apache2/sites-available/example.com.conf

The above code should replace the below:

```
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

Finally, you will want to include a do not track policy. Create your dnt-policy.txt document and place it within the .well-known directory (if you haven't already), and within it include the full policy by the EFF to explain your commitment to never track visitors. This file isn't required to produce a security.txt file, it's just another way you can add something to your site to underpin your security intentions.

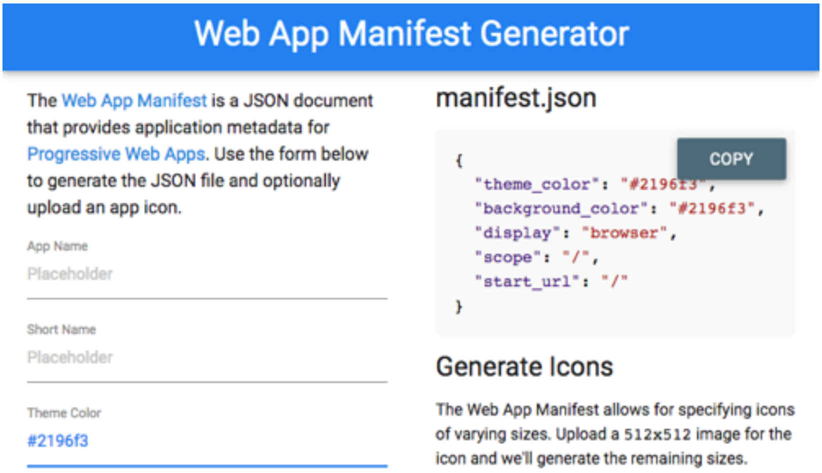
Chapter 27:

Making your PWA image conscious

Ensure your PWA is complete. Add this manifest, and a browserconfig for old IE users to add tile support.

site.webmanifest

We've already examined the benefits of building favicons and the various formats you needed to provide to ensure that every browser and device that might come across them could utilize them. We're now going to look at progressive web applications and how a couple of tiny files can make use of the images you have created and put them to use on a visitors' device's home screen or browser tile.



The Web App Manifest is a JSON document that provides application metadata for Progressive Web Apps. Use the form below to generate the JSON file and optionally upload an app icon.

App Name	Placeholder
Short Name	Placeholder
Theme Color	#2196f3

manifest.json

```
{
  "theme_color": "#2196f3",
  "background_color": "#2196f3",
  "display": "browser",
  "scope": "/",
  "start_url": "/"
}
```

Generate Icons

The Web App Manifest allows for specifying icons of varying sizes. Upload a 512x512 image for the icon and we'll generate the remaining sizes.

First up, let's look at the newest and most widely used of the files. You may have heard of the file referred to as manifest.json, but as I like to follow the specifications, I prefer to use the recommended name of site.webmanifest - though whichever name you choose, it does the same thing. This JSON file will provide some icons and information about your website regarding how mobile devices should display it.

Creating A Manifest

To create the webmanifest, you should place the document in the base directory of your website, as there is a chance that mobile browsers will automatically seek the file if a visitor tries to add your website to their home screen. Within the JSON file you will need to add several property value pairs and an array of icons, I've provided a code sample below as a boilerplate plus a table of all the potential property value pairs which you can use to guide your decisions.

```
{
    "short_name": "Name",
    "name": "Name - Details",
    "description": "Description.",
    "start_url": "/",
    "icons": [
        {
            "src": "/images/droidx192.png",
            "sizes": "192x192",
            "type": "image/png"
        },
        {
            "src": "/images/droidx512.png",
            "sizes": "512x512",
            "type": "image/png"
        }
    ],
    "theme_color": "#ffffff",
    "background_color": "#ffffff",
    "display": "standalone"
}
```

Property	Description
background_color	Visible on the splash screen on app launch.
description	The purpose of your web application.
display	fullscreen (no-UI), standalone (app), minimal-ui (basic controls) or browser (normal).
display_override	Fallback if a browser doesn't support display.
icons	Contains src (url), sizes (pixels), and type (mime). 512×512 & 192×192 must be provided.
name	Long title of the web app for other uses.
related_applications	Contains platform & url. Identifies useful apps.
scope	URL's considered to be part of your app.
screenshots	Contains src, sizes, and type. Images of your app (they should have the same aspect ratio and size from 320px to 3840px).
short_name	Short title of the web app for limited space.
shortcuts	Array of (at least) name / url pairs for menu items which only trigger within a PWA.
start_url	Directly opens the app, not a landing page.
theme_color	The color you want the UI to be for your app.

Reference You will soon be able to make your PWA integrate better with the OS by removing the window chrome: https://alistapart.com/article/breaking-out-of-the-box/

Implementation

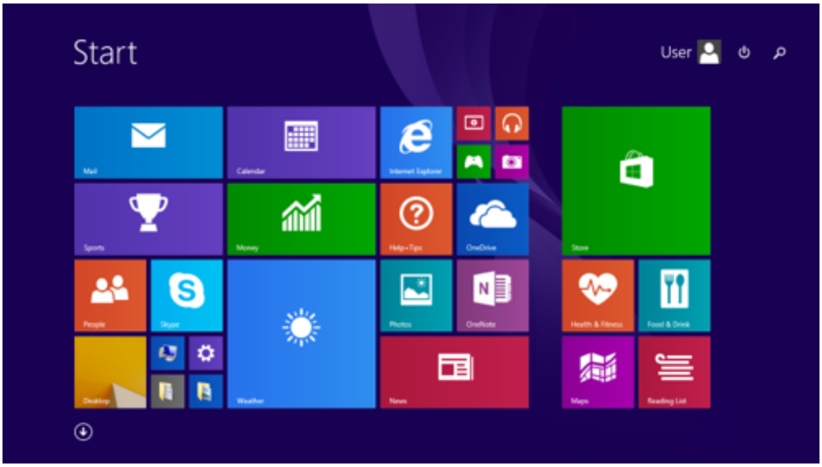
Once you have created your manifest file, you'll need to link to the file within the head of your HTML to ensure that it'll be picked up by all supporting browsers on mobile and desktop using the below code.

```
<link rel="manifest" href="site.webmanifest">
```

Reference	For a comprehensive guide to app manifest creation, read https://www.w3.org/TR/appmanifest/ and https://web.dev/add-manifest/ or https://developer.mozilla.org/en-US/docs/Web/Manifest

browserconfig.xml

Now we have to talk about Internet Explorer again (sigh). The Browser Configuration file (BrowserConfig) was created by Microsoft as their spin on the manifest format for IE11. Rather than being built using clean JSON, it's created using strict XML, and rather than being used to show icons across a variety of handheld devices, this particular file only shows tiled images on one browser, you guessed it, their own.



It's fair to say that usage of IE11 has dropped like with every other version of Internet Explorer, so unless you have an IE fanbase out there, you might decide to give this file a miss as it's near deprecation. However, as it's the last version of Internet Explorer and there are still a number of users clinging on to old technology, and you might find some fringe use-cases for it (plus it's pretty easy to implement), it might be worth integrating anyway for compatibility's sake.

Reference For a comprehensive guide to browser configuration files, the specification is at https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/dn320426(v=vs.85)

Adding Tiles

To keep things simple (as this is only for Internet Explorer users), you'll want to start by creating a browserconfig.xml and putting it in the base directory of your website. Every XML file has a doctype, so you will want to provide one, below this you'll find a browserconfig, msapplication and tile element (stacked). The elements of note are the TileColor which acts as the background transparency color, and the four URLs which cover IE's many tile sizes within IE and Windows.

```
<?xml version="1.0" encoding="utf-8"?>
<browserconfig>
    <msapplication>
        <tile>
            <square70x70logo src="images/small.png"/>
            <square150x150logo src="images/medium.png"/>
            <wide310x150logo src="images/wide.png"/>
            <square310x310logo src="images/large.png"/>
            <TileColor>#ffffff</TileColor>
        </tile>
    </msapplication>
</browserconfig>
```

Other Features

As a matter of personal preference, I usually only tend to include the square150x150logo tag alongside the TileColor element as the other square elements will scale the other image to fit, making the middle image a nice compromise of both quality and performance.

Within the msapplication element there are a couple of other things that browserconfig can do, mainly setting badge and notification reminders within Internet Explorer - though I'd argue that it's bad UX as popups and toasts will irritate visitors and diminish their good will.

```
<badge>
    <polling-uri src="badge.xml"/>
    <frequency>30</frequency>
</badge>
<notification>
    <polling-uri  src="1.xml"/>
    <polling-uri2 src="2.xml"/>
    <polling-uri3 src="3.xml"/>
    <polling-uri4 src="4.xml"/>
    <polling-uri5 src="5.xml"/>
    <frequency>30</frequency>
    <cycle>1</cycle>
</notification>
```

Implementation

Because I consider them bad practice in general, I won't go much further into how they operate, though you'll notice that they require additional markup in XML. For details, check the browserconfig.xml specification, as it provides boilerplates and live examples to use. With all these accounted for, there's little else to add, except to put a reference to the document in your HTML head (next to the manifest).

```
<meta name="msapplication-config"
content="browserconfig.xml">
```

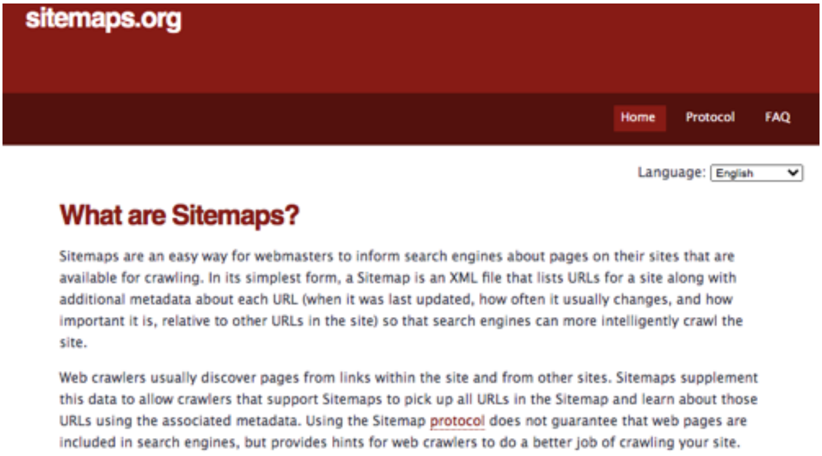
Chapter 28:

Mapping the index of your entire website

Generate a complete inventory of your pages to ensure that search engines can correctly index your site.

sitemap.xml

In Chapter 25, we covered the search engine benefits of having the robots.txt file on your website. We'll now cover the companion file for boosting the search value of your content - the sitemap. Don't be fooled by its tiny appearance, as it will declare to a search engine exactly which pages of your content you'll want to be indexed and how frequently you'd like it to be re-indexed (based on exactly how often you update it). Every site should have one of these included.



What are Sitemaps?

Sitemaps are an easy way for webmasters to inform search engines about pages on their sites that are available for crawling. In its simplest form, a Sitemap is an XML file that lists URLs for a site along with additional metadata about each URL (when it was last updated, how often it usually changes, and how important it is, relative to other URLs in the site) so that search engines can more intelligently crawl the site.

Web crawlers usually discover pages from links within the site and from other sites. Sitemaps supplement this data to allow crawlers that support Sitemaps to pick up all URLs in the Sitemap and learn about those URLs using the associated metadata. Using the Sitemap protocol does not guarantee that web pages are included in search engines, but provides hints for web crawlers to do a better job of crawling your site.

Reference For a comprehensive guide to XML sitemaps, read the spec at https://www.sitemaps.org/protocol.html

Creating The Map

Before we create our XML sitemap, it's important to make sure the file will be referenced correctly. If you've created a robots.txt file, you should have created a Sitemap link to this file (in readiness for its creation). If you haven't, add one. If you've not made a robots.txt file, it's time to make one, as the sitemap will not be linked anywhere else in our documents. Search engines seek it out in the robots file.

```
Sitemap: https://example.com/sitemap.xml
```

With that out of the way, let's build the sitemap.xml file by creating a new file in (you guessed it) the base directory of our website. Within this file, you'll want an XML doctype declaration with urlset elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

</urlset>
```

Elements

Within the urlset element, you'll want to provide an url tag for every page you want to be indexed. You'll naturally need to provide a URL of the website (using the loc tag - which is required), and there's a few other optional elements you can use to assist search engines.

```
<loc>https://example.com</loc>
```

The lastmod element does mostly what it sounds like. It declares when the website was last modified in an attempt to tell any search engines reading the document that the page needs re-indexing.

```
<lastmod>2021-01-01</lastmod>
```

Next up is the changefreq element, which works similarly to the lastmod tag, however in this case. Rather than specifying when the document was last updated, it tells search engines that the document is updated on a routine basis. You can set it to always, hourly, daily, weekly, monthly, yearly or never update - though please note that Google will only consider this request, it can choose to ignore it.

```
<changefreq>monthly</changefreq>
```

Again, another useful property, the priority element, has a value which ranges from 0.0 to 1.0 (0.5 is the average) that will give search engines an idea of how a page ranks in importance against other pages on your site. Note that it won't give you a better placement in results, it lets them know what web pages you value the most.

```
<priority>0.5</priority>
```

Multi-Maps

Finally, If you want to merge multiple sitemaps together, you can do so provided the final file is no larger than 50mb and contains no more than 50,000 URLs. The code is slightly different, using a sitemapindex element instead of urlset. Within sitemapindex you'll have a sitemap tag for each sitemap and a loc element (for the URL) plus a lastmod element which should contain the date the document (each sitemap) was last modified. You can gzip such files to optimize them too.

```
<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/
sitemap/0.9">
    <sitemap>
        <loc>https://example.com/sitemap1.xml.gz</loc>
        <lastmod>2021-10-01T18:23:17+00:00</lastmod>
    </sitemap>
    <sitemap>
        <loc>https://example.com/sitemap2.xml.gz</loc>
        <lastmod>2021-01-01</lastmod>
    </sitemap>
</sitemapindex>
```

The sitemap is a simple file that starts off easy to put together, though if you have countless pages on your site, you may want to manage the entire process using a software tool (due to how time laborious it can be). Plus, maintenance could be tricky with weeding out dead links and redirecting old pages. Despite the challenges you may face, in a search and social world, it's a critical document to have.

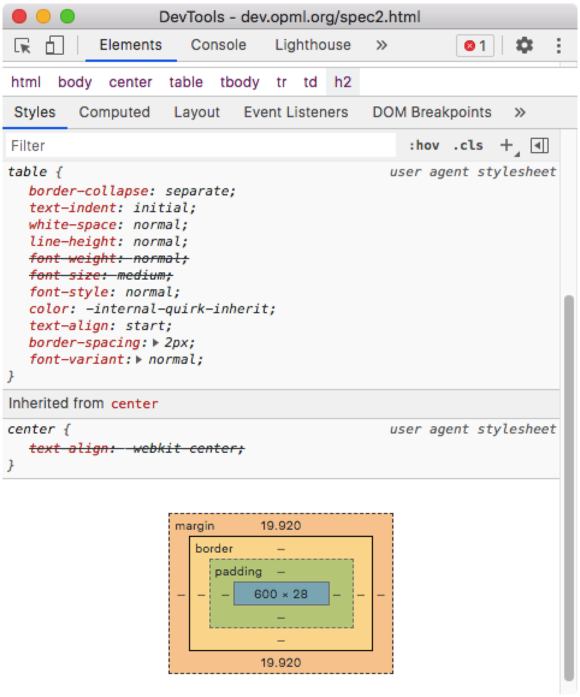
Chapter 29:

Get a solid backbone for your web layout

Every site begins with some generic styles, these are mine. Generate a DRY stylesheet for your project.

style.css

We've already paid homage to CSS within the Print chapter, ensuring that you have a stylesheet available for when your visitors want to have a PDF or paper copy of your documents. Next, we're going to create a basic stylesheet for your document to ensure that when you start coding, your styles will match as closely as possible across all browsers. Yes, what I'm talking about is a CSS reset, something which is optional, but many web developers often find helpful.



```
table {                                    user agent stylesheet
   border-collapse: separate;
   text-indent: initial;
   white-space: normal;
   line-height: normal;
   ~~font-weight: normal;~~
   ~~font-size: medium;~~
   font-style: normal;
   color: -internal-quirk-inherit;
   text-align: start;
   border-spacing: ▸ 2px;
   font-variant: ▸ normal;
}
```

```
center {                                   user agent stylesheet
   ~~text-align: -webkit-center;~~
}
```

Since web browsers have come on a long way from the old days of Internet Explorer and the days of normalize.css, the need for a reset has really diminished by quite a level. Some developers choose not to work with a base stylesheet at all; however, I find it useful to cover as many use-cases as possible and as this book is all about small useful files, I'll cover it here so that you've got one if you require it.

Reference As with HTML, don't forget to validate your code. It's worth ensuring that you don't have any syntax errors to avoid display bugs https://jigsaw.w3.org/css-validator/

Within our CSS we will have a reset plus a few optional extras that you might find useful on your development journey (I certainly keep re-using these). We'll cover hiding and showing content safely (for disabled users), web fonts will be included, and so will color through the document. You should only include the styles you'll actually use.

Reference CSS is a constantly evolving language. To keep your skills current, check the latest developments at https://www.w3.org/Style/CSS/specs.en.html or https://developer.mozilla.org/en-US/docs/Web/CSS

Reset Styles

First, let's add our reset stylesheet. I've based this upon the boilerplate provided by Andy Bell, as it's one of the cleanest and least fiddly that exist - after all, there's no point having unnecessary bloat. Saying that, it's not without its faults. I've added a few of the tags it was missing at the time of print (see H5 and H6) and made some tweaks.

```
@media screen {
    *, *::before, *::after { box-sizing: border-box; }
    html {-moz-text-size-adjust: none; -webkit-text-size-adjust:
    none; text-size-adjust: none;}
    canvas, img, picture { display: block; max-width: 100%; }
    button, input, select, textarea { font: inherit; }
    body { line-height: 1.5; min-height: 100vh; text-rendering:
    optimizeSpeed; }
    ul[role='list'], ol[role='list'] { list-style: none; }
    blockquote, body, dd, dl, figure, h1, h2, h3, h4, h5, h6, p
    { margin: 0; }
    html:focus-within { scroll-behavior: smooth; }
    a:not([class]), abbr:not([class]), acronym:not([class]),
    ins:not([class]) { text-decoration-skip-ink: auto; }
}
```

Reference To view the original CSS reset that this was based upon, visit https://piccalil.li/blog/a-modern-css-reset/

Other Code

Below you'll see the additional code I like to include in every reset. There are a number of useful things which we'll go through line by line. First up is the ability to color the foreground and background of selected text. Next is adding a border around quotes, keyboard characters and any type of code (to define easier). There is also a style for acronyms to find them better, using a dotted bottom border.

Next up, I make deleted or strikeout text grey to show that it's been removed. I also provide a base web-safe font stack that works on all devices for both the page and all code elements (though you can replace these with your own typeface choices). I make captions italic and dt tags bold for emphasis. There are classes to safely hide and show content without affecting accessibility. And for users of the u element, it'll show a wavy line, so they won't be confused with links.

```
@media screen {
    ::selection { background-color: #ffffff; color: #000000; }
    blockquote, code, kbd, pre { border: 1px solid grey; }
    abbr[title], acronym[title] { border-bottom: 1px dotted grey; }
    del, s { color: grey; }
    body { font-family: font-family: -apple-system,
    BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu,
    Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif; }
    code, kbd, pre, samp, tt, var { font-family: 'Courier New',
    Courier, 'Lucida Sans Typewriter', 'Lucida Typewriter',
    monospace; font-size: 0.8em; }
    caption, figcaption { font-style: italic; }
    dt { font-weight: bold; }
    .hide { height: 1px; left: -1000px; overflow: hidden; position:
    absolute; top: -1px; width: 1px; }
    .show { height: auto; left: auto; overflow: auto; position: relative;
    top: auto; width: auto; }
    u { text-decoration-style: wavy; }
}
```

Conclusion

As with all CSS styles, you'll need to ensure that you add a reference to the file within the head of your HTML file for it to load correctly. You could choose to keep the code within the same file as your print style sheets and media queries to reduce the HTTP request count. But the management of a CSS file should be fine if it's well documented.

```
<link rel="stylesheet" href="assets/styles.css">
```

This code serves as a basic boilerplate, and you can do with it what you like. You can use as much or as little as you prefer in your own styles. Just remember that CSS is constantly evolving, and it makes sense to keep an eye on the specifications to see what new features you can use, ensure you know how the cascade works, and don't add more code than you need as your code can quickly get cluttered.

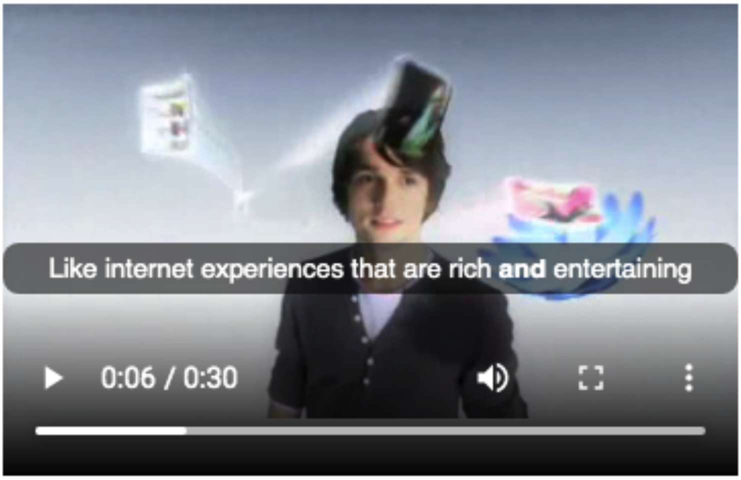
Chapter 30:

Providing accessibility for web videos

Meet accessibility standards by creating subtitles. Using timestamps, add text to speech with ease.

subtitles.vtt

Accessibility is as important on the web as it has always been. When you work with videos, a common method of ensuring inclusivity is by giving your visitors subtitles or captions so that any deaf or hard of hearing visitors can still enjoy your content. Not only is accessibility a legal requirement, but it makes sense as inclusive requirements make up quite a sizable proportion of the population and therefore could make up a large portion of your visitors. So, let's make some subtitles.



copyright © upc slovakia 2008

Using the Web Video Text Tracks Format (WebVTT) we can produce the subtitles our visitors will require. It uses plaintext to generate the text alternative. One thing to mention is that while you can create subtitles manually because it relies on timing and accuracy, doing it in your IDE or notepad can be fiddly. You may therefore want to use some software which can help create the files for you, it's a rare instance where tooling actually beats coding and there are apps available on every platform to accomplish this task for you.

Reference For a comprehensive guide to WebVTT, read the specification at https://www.w3.org/TR/webvtt1/ and use the guides at https://developer.mozilla.org/en-US/docs/Web/API/WebVTT_API and http://html5doctor.com/video-subtitling-and-webvtt/

Including Subtitles

Whereas many other files require you to include a reference to the file within the head of the HTML document, subtitles are different. This is because they are attached directly to each video, so this means you can include multiple tracks (say for different languages). You can also provide as many VTT files as required, as you are not limited to one per page. If you have 10 featured videos on the page, have 10 tracks.

```
<video height="480" width="640" controls>
    <source src="videos/video.mp4" type="video/mp4">
    <source src="videos/video.webm" type="video/webm">
    <track kind="subtitles" src="videos/subtitles.vtt" label="English"
    srclang="en">
</video>
```

Building Subtitles

Create your `subtitles.vtt` document and place it within a folder where you plan to store all your other video assets. Within the file, we must declare that the document is a WebVTT file by using the below (replacing <Title> with whatever you like, perhaps something that describes your video). This must be the first line of the document.

```
WEBVTT - <Title>
```

Next we need to add some subtitles, on each first line we need to provide an index (known as a cue identifier). Each must be unique, I like going in numerical order of when the subtitle will appear as it's easier. Below this, you have to add timing details (from > to) in the format hh:mm:ss.ttt (hours, minutes, seconds, milliseconds) as it should appear onscreen. Below the timestamp, you then write your subtitle content text, which will overflow onto multi-lines. Easy!

```
1
00:00:12.340 --> 00:00:18.642
- An example of a subtitle to be shown onscreen.
```

Styling Subtitles

Next we're going to cover the ways you can style your subtitles as you'll probably want to give them a few stylistic flourishes with fonts and color, though ensure they are accessible if you change anything. The essence of styling VTT files falls into two camps; you can either use the older B (Bold), I (Italic), and U (Underline) elements to give them some basic style, or (my preference), use CSS via the C element.

```
<c.classname>Content.</c>
```

Code Comments

If you want to include comments within your file (you may well want to, as you can imagine with a long video, it could become unwieldy), all you need to-do is include the NOTE header and place a comment below it. As illustrated by the below, it can span multiple lines.

```
NOTE
Here is a comment.
And yes, it can span multiple lines.
```

VTT Settings

Finally, there are a few settings in the VTT file you can use on each track that appears, so if you have a moment in a video where your subtitle would be better placed elsewhere on the screen, it's possible to do a bit of repositioning. Though I'd advise against this unless you need to, as visitors will expect to find their captions and subtitles in the same place each time (and might think your player is broken).

```
00:00:05.000 --> 00:00:10.000 align:start line:63% position:72%
00:00:10.000 --> 00:00:15.000 align:start line:0 position:20% size:60%
00:00:15.000 --> 00:00:20.000 align:end line:-1 vertical:rt
```

Setting	Description
align	Declares the alignment of the text. It's affected by both size and vertical:rl / vertical:lr values (if they are used).
line	Declares where text appears vertically, unless vertical:rl or vertical:lr included then it specifies horizontally.
position	Declares where text appears horizontally, unless vertical:rl or vertical:lr included then it specifies vertically.
size	Declares width of the text area, unless vertical:rl or vertical:lr included then it specifies the height.
vertical:lr	The writing direction will display left to right.
vertical:rl	The writing direction will display right to left.

Summary

Don't think this is the end of the subject (or your accessibility journey). Just as you have embedded subtitles into each of your videos, you might want to consider providing a text-based alternative to videos for people who can't view videos due to bandwidth constraints or their environment. Plus, you may want to offer a signed version for those not comfortable with subtitles too (though granted, this will cost money). As you can see, inclusive design is not one size fits all.

Don't let that discourage you, though, anything you do to promote your visitors' choice of interacting with a site is a great thing. Captions and subtitles are a really useful and proven technology and this web standard, which has been around for a while, works in every modern browser, so it's ready for you to include within your website or app. It's a useful format, and a great tiny file which serves a critical purpose.

Reference The best two free editors for creating WebVTT - if you want to avoid hand coding are https://nikse.dk/SubtitleEdit/ and https://www.jubler.org/

Chapter 31:

Getting your PWA offline and app ready

Generate a basic service worker to get your progressive web application up and running using this simple tool.

sw.js

JavaScript is a wonderfully flexible language, but it's also a terribly tricky one for web developers starting out to get to grips with. In the olden days of the web, the only JS anyone had to worry about was a bit of interactivity here and there. Now, we're using it to build fully fledged web apps, and expect them to work from the first click of a users' home-screen. It's become an all-powerful workhorse.

It's with this in mind that I want to introduce you to a feature of JavaScript that progressive web applications can take advantage of, called the Service Worker. A powerful little file that helps your app run and handle network requests in the background. This allows it to-do some pretty cool stuff that native apps can handle like working offline easily. There isn't a specific way you have to build a service worker because you can pick and choose features like offline pages, function offline, pro caching, and even background sync.

Reference For a more detailed guide to Service Workers and how to code them, read https://developers.google.com/web/fundamentals/primers/service-workers

Building Assets

So let's begin creating our ServiceWorker. It's loosely built upon a number of other existing models, and essentially, it will pre-cache any filename you list to allow the file to work offline. Additionally, it will clean up the cached item if the file is updated, and it will help the browser correctly handle the caching process. It won't, however, version your caches, nor will it cache-bust your requests, nor refresh or clean out entries in the runtime cache. Sounds good? Let's go!

You'll want to create a sw.js file and place it in the base directory of your website. It's mandatory that you should do this. Within the file you'll want to start by providing a cacheName value of the project's name and, if it's not the first version of the project, a version number to go with it - as this first block of code adds the event listener for the document. You will also want to list all of your assets in the array.

```
var cacheName = "project-cache";
self.addEventListener("install", function(evt) {
    evt.waitUntil(precache());
});
function precache() {
    return caches.open(cacheName).then(function (cache) {
        return cache.addAll([
            "/index.html",
            "/error.html",
            "/humans.txt",
            "/.well-known/security.txt",
            "/site.webmanifest",
            "/assets/style.css",
            "/assets/script.js",
            "/apple-touch-icon.png",
            "/images/banner.png",
            "/images/x192.png",
            "/images/x512.png",
            "/images/icon.svg",
            "/favicon.ico",
            "/sw.js",
        ]);
    });
}
```

Creating The Cache

Next we need to include the code that will precache all the sites resources, clean up all the old caches, and populate the result with the results. It sounds complex and a little like disjointed, but I assure you that it does the job correctly. You'll want to include the below:

```
self.addEventListener("fetch", function(evt) {
    evt.respondWith(fromCache(evt.request));
    evt.waitUntil(update(evt.request));
});
function fromCache(request) {
    return caches.open(cacheName).then(function (cache) {
        return cache.match(request).then(function (matching) {
            return matching || Promise.reject("no-match");
        });
    });
}
function update(request) {
    return caches.open(cacheName).then(function (cache) {
        return fetch(request).then(function (response) {
            return cache.put(request, response);
        });
    });
}
```

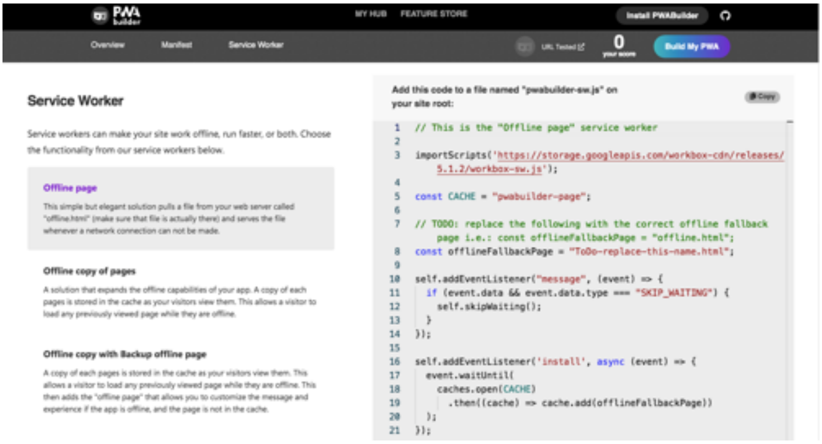
Reference If you want some more examples of Service Workers that you can implement, visit https://serviceworke.rs/ or use https://www.pwabuilder.com/serviceworker

Implementation

To add the ServiceWorker into your application, you'll want to include the below into the base of your body HTML element. You should be aware that CORS can block web apps from running locally. If your site meets the criteria for a progressive web application (HTTPS, etc) then it will work perfectly; otherwise you might find that the file won't load.

```
<script>
    "use strict";
    if ('serviceWorker' in navigator) {
        navigator.serviceWorker.register('/sw.js');
    }
</script>
```

If you are building a web app, especially one that will allow users to install it to their home screen like a native app, you will definitely want to have a ServiceWorker on your product. The compatibility of features in SW files differs wildly between different web browsers, as they are a relatively new piece of technology (Safari being the worst offender - so iOS users get a raw deal). If you're just building static pages, you can give this chapter a skip, as you may not need one.



Service Worker

Service workers can make your site work offline, run faster, or both. Choose the functionality from our service workers below.

Offline page

This simple but elegant solution pulls a file from your web server called "offline.html" (make sure that file is actually there) and serves the file whenever a network connection can not be made.

Offline copy of pages

A solution that expands the offline capabilities of your app. A copy of each pages is stored in the cache as your visitors view them. This allows a visitor to load any previously viewed page while they are offline.

Offline copy with Backup offline page

A copy of each pages is stored in the cache as your visitors view them. This allows a visitor to load any previously viewed page while they are offline. This then adds the "offline page" that allows you to customize the message and experience if the app is offline, and the page is not in the cache.

Add this code to a file named "pwabuilder-sw.js" on your site root:

```
// This is the "Offline page" service worker

importScripts('https://storage.googleapis.com/workbox-cdn/releases/
    5.1.2/workbox-sw.js');

const CACHE = "pwabuilder-page";

// TODO: replace the following with the correct offline fallback
    page i.e.: const offlineFallbackPage = "offline.html";
const offlineFallbackPage = "ToDo-replace-this-name.html";

self.addEventListener("message", (event) => {
  if (event.data && event.data.type === "SKIP_WAITING") {
    self.skipWaiting();
  }
});

self.addEventListener('install', async (event) => {
  event.waitUntil(
    caches.open(CACHE)
      .then((cache) => cache.add(offlineFallbackPage))
  );
});
```

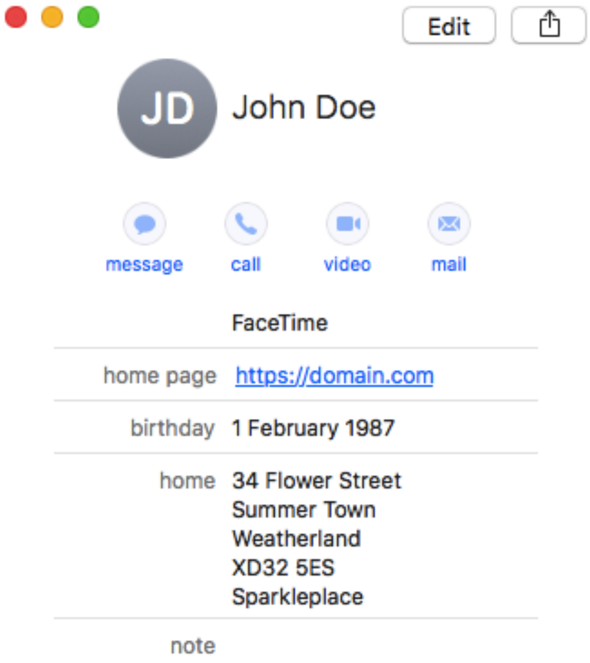
Chapter 32:

Never miss a potential contacts details

Add your business or personal details into a downloadable address book file, supported on desktop and mobile.

vcard.vcf

You've made it to the final chapter, congratulations! We've covered a lot of tiny, useful files, and this last one is no exception to that rule. The vCard is a digital business card which, upon download (and running) integrates with the users' default address book. The format of this file is very similar to that of the event.ics format. If you run any kind of business and want users to remember you, include this on your site.



John Doe

FaceTime

home page	https://domain.com
birthday	1 February 1987
home	34 Flower Street Summer Town Weatherland XD32 5ES Sparkleplace
note	

Your business might have several contact details (take for example employees) which customers can take advantage of, or you may have the physical premises where clients can choose to visit you. By having a digital business card, you allow users to take your details at a click and have them available from whatever device they have their contacts synced too. vCards are compatible with every desktop and mobile platform, so you can imagine how useful this is on a phone.

Regarding the properties, as only a few are compatible with popular apps (such as iOS, Mac, Google Calendar & Outlook), we will place most of our focus on the elements that have gained widespread implementation. For other outstanding properties and values, they will be noted but may not offer much value, though it won't hurt if they are included. For device-specific properties, they'll be listed too.

Reference For a comprehensive guide to using the vCard syntax, check out the specification at https://tools.ietf.org/html/rfc6350 and this Apple article about size limits https://support.apple.com/en-us/HT202158

Card Creation

So let's begin by creating a vCard.vcf file and placing it wherever you like as there is no set place where you need to put your digital business cards. It's also worth noting that just like with the calendar events, you can build as many as you'd like. Next you'll need to put the opening and closing tags with a reference to your business and product in PRODID, as this acts as the generator information for the vCard document (all of these components are required in the file).

```
BEGIN:VCARD
VERSION:3.0
PRODID:-//Business//Product//EN

END:VCARD
```

Name Properties

So now we have to populate some content between the VCARD elements. First up are the name properties. You have N (name) which lists every name value type in a specific order. ORG, which lets you name your brand. And there are some optional properties that Apple doesn't support, you can include these if you choose as they won't affect compatibility (iOS ignores them) or you cannot use them.

These less compatible properties include FN (full name) where you can format your name how you decide. ORG where you can list a business name. NICKNAME, where you can list your handle. GENDER where you can provide a value of M (male), F (female), or O (other). TITLE, where you can provide any job title you hold. And ROLE, where you can specify what task you perform in your current occupation.

```
N:Last;First;Middle;Prefix;Suffix
ORG:Company
FN:Full Name
GENDER:O
NICKNAME:Nick
ROLE:Role
TITLE:Job
```

Date Properties

Next up are the date properties, there are only two of them. The first is the BDAY (birthday) which will actually integrate into calendars (on compatible devices). Aside from that, you have an ANNIVERSARY item where you can provide a date (though support for this is pretty low).

```
BDAY:1999-01-01
ANNIVERSARY:19950101
```

Phone Properties

Next up, we have the properties for allowing people to call you on the phone (if you want them too). This is a really useful feature if you have a technical support line and visitors import the card into their smartphone. The formatting of the first two are defined by a type attribute which can have a value of IPHONE, HOME or WORK along with the number. You'll notice an IMPP protocol property, which can be used for connecting with Skype users via their profile identity.

```
TEL;type=IPHONE;type=CELL;type=VOICE;type=pref:01234 567890
TEL;type=HOME;type=VOICE:09876 543210
IMPP;X-SERVICE-TYPE=Skype;type=WORK;type=pref:skype:ProfileID
```

Email Properties

Most sites have an email account for technical support or sales, and this section allows you to include as many as you are likely to require. With the type attribute, you can have a value of HOME, WORK (or using the third and fourth lines special formatting - OTHER). You can also set an email as your preferred account using ";type=pref".

```
EMAIL;type=INTERNET;type=HOME;type=pref:email@example.com
EMAIL;type=INTERNET;type=WORK:email@example.com
item1.EMAIL;type=INTERNET:email@example.com
item1.X-ABLabel:_$!<Other>!$_
```

Social Properties

In this next section, we're going to cover ways visitors can contact you on the web. The first two lines of the below cover adding your homepage to your contact card (which you'll want to include). Below that you'll find social media profile references for Twitter, Facebook, and LinkedIn, which are the big three that iOS and Android support. There is also an option for GameCenter for iOS users to connect too.

```
item3.URL;type=pref:https://example.com
item3.X-ABLabel:_$!<HomePage>!$_
X-SOCIALPROFILE;type=twitter:http://twitter.com/ProfileID
X-SOCIALPROFILE;type=facebook:http://facebook.com/ProfileID
X-SOCIALPROFILE;type=uk.linkedin.com;x-user=ProfileID:http://
uk.linkedin.com/in/ProfileID
X-SOCIALPROFILE;type=gamecenter;x-
user=ProfileID:0000000000:http://link.gc.apple.com/players/
G:0000000000
```

Location Properties

Next up we have the location properties. The first two items enable the ability to add an address into your contacts. The formatting of the address is quite similar to that of N (name) in that it must be written in a specific order, and you can provide a HOME or WORK address in the type attribute. I've provided an example of this below to help you.

There are also two optional properties which aren't widely supported, called GEO where you can put map co-ordinates and LANG where you can provide the language you speak (in case users aren't aware).

```
item2.ADR;type=HOME;type=pref:;;Street;Town;Region;ZipCode;C
ountry
item2.X-ABADR:us
GEO:geo:00.000000,-000.000000
LANG;TYPE=work;PREF=1:en
```

Other Properties

There are some other properties which may be of interest. Photo can add an image of you (or your business). Most apps require the photo to be embedded using Base64 and added where the word IMAGE is (spanning over multiple lines). This will really bulk up your file size, however, so consider if you really need an image, as it's not the most performant option. Finally, there is the UID, which should be a unique code identifier (perhaps an MD5 hash as used in the event.ics file).

Optional properties (with little support) include a LOGO (similar to PHOTO, so you may as well use photo), a NOTE property where you can provide details about you or your business, and RELATED contact details where you can directly link to another useful contact card.

```
PHOTO;ENCODING=b;TYPE=JPEG:IMAGE
UID:20f72c88-7952-44fa-8801-1db5cf2bc1cb
LOGO:https://example.com/logo.jpg
NOTE:Details.
RELATED;TYPE=contact:https://example.com/contact.vcf
```

Reference For more information about the vCard format and additional syntax details, check out the Wikipedia article at https://en.wikipedia.org/wiki/VCard

Summary

And that's it. With all of these properties together, you can create a really useful contact card that your visitors can use whenever they want to get in touch with you quickly. Having your website, email, phone or physical store in their contacts means that if you clients have a problem, they won't have to jump through hoops to get answers.

This also concludes the book. We've detailed how to create every one of the tiny but useful standardized files that exist on the web. Thanks for reading, and I hope it's been useful to you. If you learned about a couple of new formats that could be added to your workflow and improve your visitors experience, it's been worth the effort of compiling this resource. Have questions or comments? Get in touch!

Developing websites has never been more complicated because more types of syntax exist than at any point in the web's history.

Alexander Dawson is on a mission. Using his skills as a web developer, he's aiming to teach you over 10 syntax languages in one go. How is this possible? Why would you need to know so many things? This is what this book aims to answer. Using concise chapters and a focus on the tiny web assets most developers fail to appreciate, you shall go on a voyage of discovery and become a master of the 52 micro-files which can underpin every website.

Visit https://alexanderdawson.com