RapidWeaver 4

Theme SDK

Realmac Software Theme Tutorial: Table of Contents

RapidWeaver 4.0 Theme Tutorial

Requirements: 6

Before you start 6

Installing Your Theme

Access The Files 7

The Basics

HTML 8

Semantics 8

Tags 8

Attributes 8

ID's and Classes 9

Div's and Span's 9

Basic Page Structure 11

(Continued on next page...) 11

Summary 12

CSS 13

CSS Selectors 13

Global Selectors 14

Class Selectors 14

ID Selectors 14

Nested Styles 15

(Continued on next page...) 15

Linking the styles.css file to index.html 16
Summary 16
Rapidweaver Templates
The index.html file 17
Document Structure 17
RapidWeaver Syntax Tags 19
The %pathto(…)% Tag 19
CSS Selectors and the index.html file 21
Linking the styles.css file to index.html 21
Styling Your Theme
Common Styling 22
Link Styling 23
Site Width 24
Content and Sidebar 27
Clearers 27
Content Padding 28
Page Header 30
Title and slogan 30
Navigation Styling 32
Default Styles 33
Current Link Styling 35
Footer Styling 36
Applying a background image 38
Adding A Theme Preview 39

Conclusion

Useful CSS Links 40

Appendix

SDK Appendix 76

RapidWeaver 4.0 Theme Tutorial

Welcome to the Realmac Software Theme SDK, the aim of this tutorial is to give you a basic understanding of how you can build a unique theme for RapidWeaver. You'll learn about theme syntax, the property list file and how it controls your theme, how themes are structured and some basic styling.

Requirements:

RapidWeaver 4.0 or newer (RapidWeaver 3.6 is available for those not using Leopard)
A text editor (TextWrangler or similar)
A CSS Editor (CSSEdit or similar) recommended
Skill Level: RapidWeaver / Web enthusiast. Basic XHTML and CSS knowledge an advantage.
Objectives: To gain an understanding of how RapidWeaver themes are constructed and learn some basic CSS styling for the theme.
Time: 1 - 2 hours depending on skill level.

Before you start

Before you start you need to make sure you have RapidWeaver 4.0 or newer installed. You can download the latest version of RapidWeaver from http://www.realmacsoftware.com. This tutorial has been designed and written for 4.0, however it is perfectly apt for use with 3.6 as the theme structure and development process is identical.

You'll also need a text editor, TextWrangler from Bare Bones Software (http://www.barebones.com) is free and provides syntax highlighting which makes reading large amounts of code infinitely easier . Alternatively you can use TextEdit which is shipped with the Mac.

Finally a good CSS editor will also come in very useful. I recommend you buy a license for CSSEdit from Mac-Rabbit (http://macrabbit.com), it's not essential but it makes CSS coding a breeze!

With everything in place, let's get started...

Installing Your Theme

Install the Tutorial.rwtheme file by double-clicking it. You then need to restart RapidWeaver for the theme to become active.

Access The Files

Once you have the theme installed, open the Tutorial Site project file by double clicking it, once RapidWeaver has loaded choose the tutorial theme from the theme drawer. Right click on the theme preview and select Show Contents.

This will reveal a Contents folder in the finder, inside are all the files required to construct a RapidWeaver theme. You should have the following files:

- **index.html** - layout template file (XHTML)
- **styles.css** - base stylesheet (CSS)
- **colourtag.css** - colour style variations stylesheet (CSS)
- **print.css** - print stylesheet, styles the layout of your page when printed (CSS)
- **handheld.css** - handheld device, styles the layout of your page when viewed on a mobile device (CSS)
- **javascript.js** - required javascript for the theme
- **images** - where all images for the theme are placed
- **css folder** - theme variations CSS files are placed in here, such as sidebar location, page-width etc.
- **preview.png** - 115x125px image/logo of your theme displayed in the theme drawer. (Version 3.6 and under)
- **preview_large.jpg** - 800x950px image/logo of your theme displayed in the theme drawer. (version 4.0 and up)

The Basics

Firstly before we begin any work on the theme it is important to understand the fundamentals of a website. This chapter will cover basic XHTML and CSS as well as the general workings of a webpage. If you are comfortable with these areas then you may wish to skip this chapter.

HTML

Semantics

When designing any webpage it is important to use valid XHTML your .html pages should contain all the vital information you wish to communicate to your viewer. All other code such as styling and javascript should be contained in external files and linked into your document. The idea behind this is to ensure that if your page was viewed with no styling it reads easily and clearly.

Tags

HTML uses tags to mark up areas of the page to ensure the browser reads it correctly. For example a `<hl>` tag denotes a heading, this will be displayed more prominently on the page with a larger bold font.

Tags nearly always have both opening - - and closing - -mark-up, closing tags are prefixed with a forward slash /. For example a paragraph will be displayed as follows.

```
<p>This is a paragraph!</p>
```

Attributes

Tags can also be assigned attributes, these reside inside the opening tag and can be used to assign extra information to your code. For example a link can be marked up as follows.

```
<a href=”http://www.realmacsoftware.com” title=”Visit the Realmac
Homepage”>Visit the Realmac Homepage</a>
```

Notice the **href** and **title** inside the <a> tag. The href specifies the url the link is to go to, the title attribute gives a textual description of the urls destination.

ID's and Classes

A couple of other important attributes are ID's and Classes these can be used to assign meaning to certain parts of your page. More importantly they are used alongside CSS to specify specific areas for styling. A list of links that you wish to use for navigation could be marked up as follows;

```
<ul id="navigation">
    <li><a href="page1.html">Home</a></li>
    <li><a href="page2.html">Page 2</a></li>
    <li><a href="page3.html">Page 3</a></li>
</ul>
```

Don't worry too much if this doesn't make sense just yet, we will return to this in the tutorial.

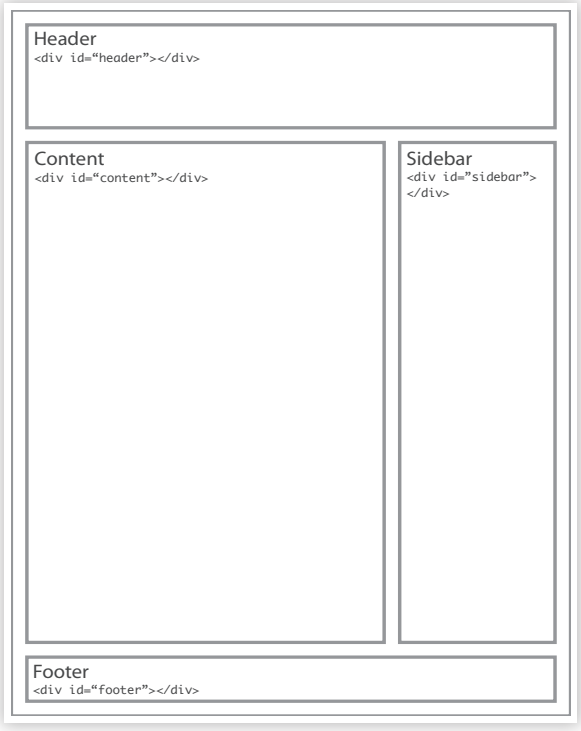
Div's and Span's

Finally the last section of this chapter focus' on and tags, 's especially are used quite frequently in the theme template. A is a block element, this means that any content inside the tag will be moved onto a new line and automatically take up the full width of the page, just like a <p> or <h1> tag.

A is an inline element, this does not disrupt the flow of the page but instead will flow with the text in the same way as <b>, <i> or <a> tags.

Whilst not having a great amount of semantic meaning, 's can be used to mark out areas of your site to allow easy styling with CSS. For example the following image illustrates a common site layout. One which we will use in the tutorial. It can be broken down into five distinct areas;

- Container
- Page Header
- Content
- Sidebar
- Footer



These can be marked out accordingly with XHTML;

```
<div id="container">
    <div id="header">Header </div>
    <div id="content">Content</div>
    <div id="sidebar">Sidebar</div>
    <div id="footer">Footer</div>
</div>
```

Notice the four distinct sections are wrapped in an outer container, this make it easier to set an overall site width. **ID** attributes are also used to identify the regions.

Basic Page Structure

Now we have seen how 's can be used to mark out the content areas of the page we can focus on how this is inserted into a working **.html** page that can be viewed by a browser.

There are two main areas of an **XHTML** document, the <head>, which contains all the information required before the page loads such as page title and stylesheets. Second is the <body> this contains anything you wish to be visible to the user.

The <body> and <head> tags are wrapped in an <html> tag, this tells the browser that we want to render the page as <html>. All of this is preceded by a <DOCTYPE> declaration. This statement tells the browser that we would like our page to be rendered as valid XHTML. By ensuring our page validates we can ensure that the page will display as intended across all modern browsers.

The code on the following page illustrates the standard mark-up for a basic **XHTML** page including our segmented page layout.

The observant among you may have noticed a <link> tag in the <head>, this imports all of our styles into our page. We will cover this in greater detail later in the tutorial.

(Continued on next page...)

```
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Strict//EN” “http://
www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
    <head>
       <title>This is the Page Title</title>
       <link href=”styles.css” rel=”stylesheet” type=”text/css”
media=”screen” />
    </head>
    <body>
       <div id=”container”>
           <div id=”header”>Header </div>
    <div id=”content”>Content</div>
           <div id=”sidebar”>Sidebar</div>
           <div id=”footer”>Footer</div>
       </div>
    </body>
</html>
```

Summary

This has been a **VERY** quick overview of **XHTML** and its structure, hopefully however it will make the **index.html** template for the theme clearer. The next chapter will cover the basics of **CSS** and styling a webpage.

CSS

CSS stands for **Cascading Style Sheets** and is widely used by all modern web designers to style webpages. It allows for an incredible level of flexibility to be applied to your site and once understood is very simple to use.

CSS Selectors

A **CSS** statement is formed in three parts, the **selector**, the **property** and a **value**.

```
body { color: red; }
```

Here `body` is the **selector**, `color` the **attribute** and `red` the **value**. This command will style the color of all of the text on the page `red`. Simple.

A selector can have as many properties as you wish. These are wrapped in curly braces.

```
p {
    color: blue;
    font-size: 20px;
    font-weight: bold;
}
```

This statement styles all paragraph text blue, bold and has a font size of 20px. Properties end in a colon and values a semi-colon.

You may have noticed from the previous examples that **CSS** communicates with your **XHTML** content using the markup tags for example `<body>`, `<p>` or `<a>`.

Global Selectors

Any tag within the <body> of your document can be styled with **CSS**. For example.

```
a { font-weight: bold; }
```

All of the anchor tags in your page will now be bold.

Class Selectors

We may want to select only specific parts of our webpage for styling. For example we may want certain paragraphs in out document to be a different colour to the main text.

We looked at the class attribute in the previous chapter, now can see how it can be used to select only specific tags. For example in our **XHTML** document we can enter;

```
<p class="important">This paragraph is more important than the oth-
ers.</p>
```

Using **CSS** we can then select all elements with the class="important" attribute. Classes in **CSS** are preceded by a "." (period).

```
.important { color: red; }
```

ID Selectors

General and class selectors can apply to many tags within your document, the third type will select only one instance. This is the ID attribute again it was used in the previous chapter to define our page container, header, content, sidebar and footer.

```
<div id="container">Page content will be inserted between these
tags.</div>
```

CSS uses a # to identify **ID** attributes.

```
#container {width: 800px;}
```

This will set the width of our page to `800px`.

Nested Styles

When you look at the **styles.css** document you may notice that there are entries that have more than one selector. For example.

```
#contentContainer #content {...}
```

This simply means that the the **CSS** should style the element with the **ID** of content inside the element with the **ID** of `#contentContainer`. This is a useful way of targeting elements without having to give every element of your website and individual **class** or **ID**.

Another example would be selecting all anchor elements `<a>` inside paragraphs `<p>` in the sidebar perhaps to make them a different colour from other anchors in the page.

```
#sidebar p a {...}
```

(Continued on next page...)

Linking the styles.css file to index.html

In order for the browser to find our stylesheets we need to link the **styles.css** to the **index.html** file. This can be done with the <link> tags inserted in the document <head> if you look at the example in the previous chapter you will see the <link> after the <title> tag.

```
<link href="(styles.css)" rel="stylesheet" type="text/css"
media="screen"  />
```

This link tells the browser where to locate the styles.css file and will load it accordingly.

Summary

Again this chapter has quickly outlined the basics of CSS and how it can be applied to a document, it may be worth your time creating an index.html file containing a basic site layout and linking it to a styles.css file. Then add some basic HTML to the page and try styling some tags, classes and ID's to get a feel of how it works.

The next chapter covers the RapidWeaver specific tags used by the program to generate the final webpages.

Rapidweaver Templates

The index.html file

The **index.html** file contained in the theme folder is the framework for all your other pages to be generated from. Open **index.html** in your text editor and you'll see that it is a standard **XHTML** page as described in the previous chapter but it also includes numerous RapidWeaver **Syntax Tags**.

Document Structure

The themes **index.html** file follows a familiar structure that is common in nearly all modern webpages. This section aims to familiarise you with the template document.

The basic layout of the page should by now be familiar;

```
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Strict//EN” “http://
www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>

    <html xmlns=”http://www.w3.org/1999/xhtml”>

        <head>
            <title>This is the Page Title</title>
            <link href=”styles.css” rel=”stylesheet” type=”text/css”
media=”screen” />
        </head>

        <body>
            <div id=”container”>
                <div id=”pageHeader”>
                    <h1>%site_title%</h1>
                    <h2>%site_slogan%</h2>
                 </div>
```

```
                <div id="contentContainer">
                    <div id="content">
                        Content
                    </div>
                </div>
                <div id="sidebarContainer">
                <div id="navcontainer">
                    %toolbar%
                </div>
                <div id="sidebar">
                    %sidebar%
                </div>
            </div>
            <div id="footer">Footer</div>
        </div>
    </body>
</html>
```

The main content areas are distinguished by elements, each open and close tag contains a **Syntax Tag**. For example.

```
<div id="content">%content%</div>
```

The majority of the syntax tags relate directly back to a text area in RapidWeaver, it should be pretty obvious which tag corresponds to which area but there is a definition list in the **RapidWeaver Theme Syntax** chapter located in the appendix at the end of this document.

We have tried to comment **index.html** document in the **tutorial.rwtheme** file to make it as clear as possible where your content is inserted by RapidWeaver.

Note

Some tags like the %plugin_sidebar% are there for code to be entered by page plugins such as the blog and thus have no user text input.

RapidWeaver Syntax Tags

The **Syntax Tags** found in the template file will be replaced with actual content when the page is published of exported by RapidWeaver.

The tags can be placed anywhere within the template. To recap a RapidWeaver Syntax Tag looks like this:

```
%content%
```

This is the tag for the main content of the page anything in the main window in RapidWeaver will be inserted here.

```
%sidebar%
```

This tag refers to the Sidebar Window in the Page Inspector, again any text or images entered into the Sidebar in RapidWeaver will be converted to **XHTML** and will replace this tag in the **.html** file.
A full list of tags can be found in the in the file **RapidWeaver Theme Syntax** chapter located in the appendix at the end of this document.

The %pathto(…)% Tag

All RapidWeaver Syntax Tags, with one exception, look exactly the same. The exception is the `%pathto(file.extension)% tag`. This tag is used for site consolidation, essentially meaning that RapidWeaver will automatically generate links to files located in the theme folder.

When exporting a site RapidWeaver places all of the theme files in a folder called **rw_common** and generates the links to the files in this folder. This should be used when linking to any file in your theme file including stylesheets, javascript files and images.

Examples:
Linking to the styles.css file in your theme directory.

```
%pathto(styles.css)%
```

the published code will look something like this:

```
../rw_common/themes/theme_name/styles.css
```

Folders can be included before the image name eg.

```
%pathto(images/image_name.jpg)%
```

The published code will look something like this:

```
../rw_common/themes/theme_name/images/image_name.jpg
```

Note

All tags can also be used inside styled text windows in RapidWeaver. The `%pathto(...)%` tag is particularly useful should you want to include images or links to files inside your theme.

CSS Selectors and the index.html file

All the syntax tags that are placed in the body of the **index.html** file are wrapped in XHTML code with relevant CSS classes & ID's. These XHTML "wrappers" allow us to style the content easily with CSS.

For example, the %content% tag is wrapped inside two XHTML tags:

```
<div id="contentContainer">
   <div id="content">
      %content%
   </div>
</div>
```

The %content% tag is directly wrapped inside the **content** CSS id attribute, which is in turn wrapped in the **contentContainer** CSS id attribute.

Each wrapper name used in the template page should be self explanatory but familiarise yourself with the index.html code as it will help you to understand the CSS section of this tutorial.

Linking the styles.css file to index.html

Have a look at the template index.html file and you will see the following <link> tag about 6 lines down.

```
<link rel="stylesheet" type="text/css" media="screen"
href="%pathto(styles.css)%"  />
```

This link tells the browser where to locate the **styles.css** file. Notice we have used the %pathto(...)% syntax to ensure that RapidWeaver can insert the path correctly.

Now we are ready to begin styling the theme.

Note
The %content% tag is wrapped inside two tags so that we can set an overall width and apply padding, this is needed due to inconsistent support of CSS across web browsers.

Styling Your Theme

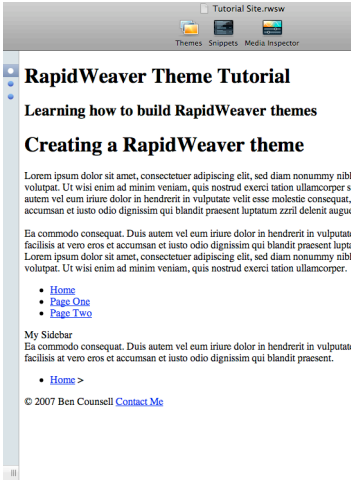
Generally we at Realmac Software use CSSEdit (www.macrabbit.com) to edit CSS files. It is particularly useful for beginners because it has an excellent graphical interface so you don't have to remember all of the CSS properties. Secondly it has the superb feature of allowing you to group areas of code in folders. However you can continue to use TextWrangler to edit the code for this section.

This section covers the CSS required to build a theme and how it can be applied to style the overall layout. A full list of RapidWeaver's style classes with explanations can be found in the CSS Styles document included in the in the file **CSS Selectors** chapter located in the appendix of this document.

Open the **Tutorial Site** in RapidWeaver, right click on the theme **RapidWeaver SDK Tutorial** and select **Show Contents** then open the **styles.css** file.

Common Styling

To add styling that will apply to your entire theme you can edit the body tag. This will style everything viewable in the browser window. This is where you should add common styles such as a standard font size, font family, font colour and the line height. To do this add the following code:




```
body {
    color: #333;
    font-size: 62.5%;
    line-height: 1.4;
    font-family: Arial, Helvetica, Geneva, sans-serif;
    padding: 0;
    margin: 0;
}
```

This code may look complex at first glance, however it's actually quite simple. Firstly to tell the CSS we want to style everything inside the <body> XHTML tag, we simply write body followed by open and close curly brackets {… }. Between the curly brackets is where the properties for each tag are placed.

Note

#333 is a hexadecimal colour value, these are used throughout CSS to assign colours.

If you are using a text editor without support for the Apple Color picker. Waffle Software http://wafflesoftware.net/hexpicker/ provide an excellent plugin that will enable you to use the Apple Color Picker in RapidWeaver (cmd+shift+c) to select your color and generate a hex code.

To add a style property to a class you firstly write the property name, such as color followed by a colon, then the style value (#333 in this case) and then a semi-colon; `color: #333;`

Next we add a `font-size` of `62.5%`. This percentage sets the standard browser font size to 10px. We can then use em units to easily size the remaining areas. An em is a unit of percentage that is specific to fonts so 1em is the equivalent of 100% of the current font size.

Therefore now we have set the browser font size to 10px it makes it easy to adjust the value to taste. If you wish all of your text to be 14px it is as simple as applying a font size of 1.4em to the container `<div>`.

The advantage of using em's is that it is a relative size, not fixed like a pixel. This allows the text to be resized for each visitors requirements. For example, a sight impaired visitor will be able to increase the size of the font without any problems.

A `line-height` of 1.4 is then added. This allows us to increase the height between each line of text giving our text a better flow and making it easier for the visitor to read. This value is a multiple so the line height will be 1.4 times the height of the font size.

Next we add a font family. I've added a web safe `font-family: Arial, Helvetica, Geneva, sans-serif`. This way I can be sure my text will display the same, or a very similar, font on each browser. I have set three alternative fonts and a default **sans-serif** type-face, if the user does not have the first font installed on their system the browser will move along the list until it finds one that is.

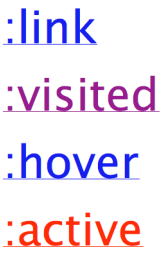
Finally we set the `padding` and `margin` to 0. This removes any `padding` and `margin` the browser may add to the window ensuring our site looks the same in all browsers.

Link Styling

Link `<a>` styles are special because they have four different states; `:link`, `:visited`, `:hover` and `:active`. Each style is applied to the link based on the users action on that link. They are all quite self explanatory, link being the original state, visited being the visited state, hover being when the user place their curser over the link and active being when the user clicks on the link.

If no styling is applied there is a default style set by the browser (usually blue, purple, blue and red) that are applied. We'll want to personalise our links and to do that we'll add the following code:

```
a:link {
    color: #82aa15;
}
a:visited {
    color: #55700f;
}
a:hover {
    color: #a2db36;
}
a:active {
    text-decoration: none;
}
```



This sets a different colour for the :link, :visited and :hover states and sets the text decoration to none for the active state. By default the link styles have an underline as their decoration. Try rolling over and clicking one of the links in RapidWeaver, you'll notice the styles change.

Site Width

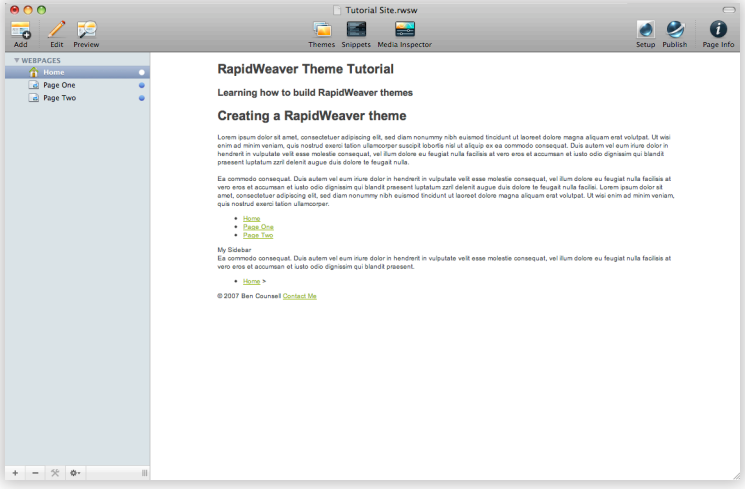
Add the following code to the #container tag:

```
#container {
    font-size: 1.2em;
    width: 720px;
    margin: 10px auto 0 auto;
}
```

The #container tag is set in the template page, it is wrapper for the entire site which is used for setting the overall width (720 pixels) and position of the site.

```
<div id="container">...Rest of Site...</div>
```

Save the **styles.css** file and refresh your window in RapidWeaver. If you now stretch the RapidWeaver window across your screen you will see that the page content is centred on the page at a fixed width.



The margin property we have added tells the browser we want the site to be centred.
In CSS you always specify properties such as margins and padding clockwise; top, right, bottom, left. So our code for the margin set the top margin to 10px, the bottom to 0px and both the right and left to auto.

The auto margin tells the browser to set the left and right margins to half the browser window width, after it has taken the container width in to consideration, therefor centring the content.

We can condense the code down a little. If the top and bottom margins are to be the same and the left and right are to be the same then the following code can be used;

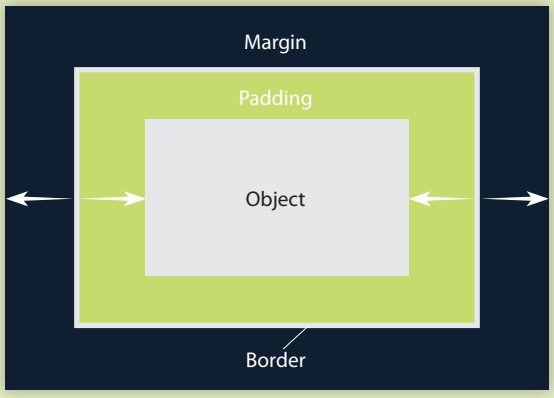
```
#container {
    font-size: 1.2em;
    width: 720px;
    margin: 0 auto;
}
```

This way the margin properties for bottom and left are inherited. If we were to simply put margin: 0; it would apply to all four sides. The same applies for padding.

CSS Tip:
This seems an opportune moment to explain a little about how padding and margin work.

- Margin is applied to the outside of an element creating white space between itself and its surroundings.
- Padding is applied inside the element creating a barrier between itself and its contents.
- Finally borders are applied these to the inside of the element.

For example:
A `<div>` with a `width` and `height` of 100px with 50px `margin`, 30px `padding` and a 20px `border` will have an overall **width** of 300px.



Content and Sidebar

Our tutorial site currently displays the sidebar content below the main content. For this tutorial we want the sidebar to be on the right side of the content.

To get the sidebar to display along side the main content, first we need to set a `width` to the `#contentContainer` and tell it to be on left side:

```
#contentContainer {
    width: 500px;
    float: left;
}
```

Next we need to set a width to the sidebar and tell it to display on the right side:

```
#sidebarContainer {
    width: 220px;
    float: right;
}
```

Note

It's important to remember that the combined width of the contentContainer and sidebarContainer should be equal to or less than the width of our container. Both margin and padding should be taken into account when calculating the width.

If the combined width is more than the container, the sidebar would be pushed below the content.

The get the content and sidebar to display next to each other the float property is used. This essentially removes the object from the flow of the document and floats it in the direction specified.

I won't go into detail about the float value but I suggested reading this W3C document: http://www.w3.org/TR/CSS21/visuren.html#propdef-float

Save your CSS and notice that the content and sidebar are now displayed side by side.

Clearers

The `#breadcrumb` and `#footer` should be displayed below the content in the theme, however at the moment they will be displayed in the sidebar area. To fix this problem we need to make sure they `clear` both the `#contentContainer` and `#sidebarContainer`.

We do this with a clearer class already set-up in the template file.

```
<div class="clearer"></div>
```

Add the following code to the CSS file to style the .clearer class:

```
.clearer {
   clear: both;
}
```

An element with the clear property applied will always reside below floated content, therefore because the is forced below the content and sidebar any code that follows it will also be pushed down. **Floats** and **clearers** are one of the most useful properties of CSS however they are also one of the more tricky aspects to grasp.

Content Padding

Adding some padding to our content area is a simple task, simply add the following to the #content like so:

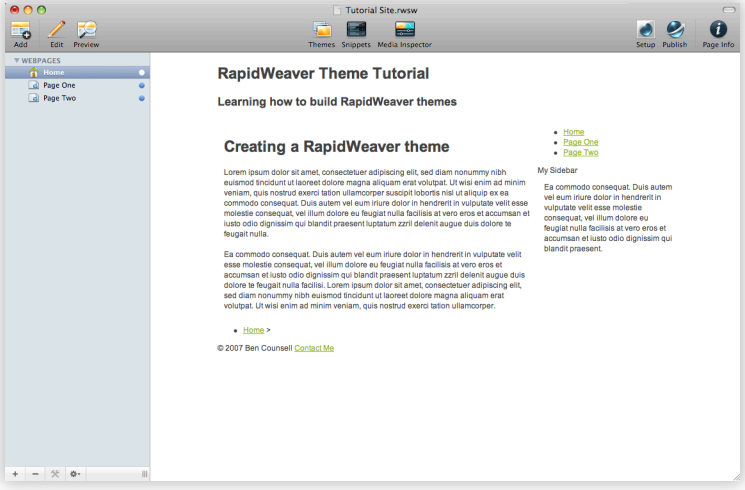
```
#contentContainer #content {
   padding: 10px;
}
```

As we have already mentioned this is slightly different to other classes we have styled before. This time there are two tags on the same line; #contentContainer #content. This tells the browser that the #content tag is wrapped inside the #contentContainer. This is important to remember as it allows you to style, for example, a link differently in the main content area than in the sidebar.

Add the same padding to the sidebar wrapper:

```
#sidebarContainer #sidebar {
    padding: 10px;
}
```

Save your CSS and you'll see your site is starting to take shape nicely!



Page Header

The page header is the area of the site where the **title**, **slogan** and **logo** is placed so it's usually important to make it stand out from the rest of the site. To do this first add some `padding`:

```
#pageHeader {
    padding: 10px;
}
```

That's nice and simple, next lets add a background colour:

```
#pageHeader {
    padding: 10px;
    background-color: #82aa15;
}
```

Title and slogan

Next change the colour and margins of the site title and slogan like so:

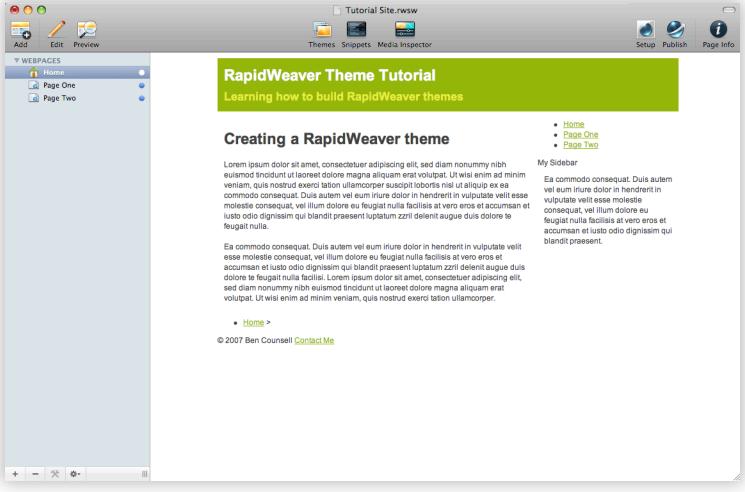
```
#pageHeader h1 {
    margin: 0 0 5px;
    color: #fff;
}

#pageHeader h2 {
    margin: 0;
    color: #e4ee3b;
}
```

The site title and slogan tags are inside a `<h1>` and `<h2>` tag respectively in the template page, these tags were nested inside the `pageHeader` **ID**.

```
<div id="pageHeader">
    <h1>%site_title%</h1>
    <h2>%site_slogan%</h2>
</div>
```

Therefor we are telling the CSS to only style the <h1> and <h2> tags this way for the pageHeader. The <h1> and <h2> tags residing in other areas of the site remain untouched.

Save your CSS and your theme should look like the preview below.



Navigation Styling

Navigation styling is an important part of your theme design, it allows users to easily navigate through your site. There are many ways to style the navigation, for this example we shall create a vertical list of links.

Before we get to the CSS a quick paragraph on how RapidWeaver builds our navigation tree. The %toolbar% **Syntax Tag** outputs the menu for the site.
RapidWeaver outputs this as an unordered list. The unordered list <ul>, is a standard XHTML tag which contains list items <li>. Inside each of these list items we insert a link to the destination page. The output for the site menu looks like this:

```
<div id="navcontainer">
   <ul>
      <li><a href="index.html" id="current">Home</a></li>
      <li><a href="page1/index.html">Page One</a></li>
      <li><a href="page2/index.html">Page Two</a></li>
   </ul>
</div>
```

The other important piece of code to take note of is the id="current" which is applied to the first link <a> tag. This style is only applied to the current page and allows us to style the current page link differently from the rest of the site. This is important to let visitors to the site know which page they are currently on. We'll look at styling current links in a few minutes.

New navigation links will be added as list items <li>..., fortunately RapidWeaver creates all of our navigation code for us, all we need to worry about is the styling. So now we have a nice bullet point list that can clearly be identifies as a contents menu for your site.

Now we have a solid logical structure we can style the list to be a little more visually appealing. Firstly we'll add some padding to the navigation wrapper to match the content and sidebar:

```
#navcontainer {
   padding: 10px;
}
```

Default Styles

As mentioned before, standard XHTML tags have default styling values applied to them by the browser. For example the default notation for an unordered list <ul> is a bullet-point, so before we begin we need to remove the margins, padding and bullet-points from the list;

```
#navcontainer ul {
    margin: 0;
    padding: 0;
    list-style-type: none;
}
```

CSS Tip
Hopefully by now, padding and margin should be familiar to you. The third value, list-style-type, is used to define the style of graphic by each list item <li>, by default it is a **bullet**. There are several other types such as **square** and **circle**, however for this example we need to remove the graphics, so we use the value **none**. [link: css properties]

To style the navigation links add the following code:

```
#navcontainer a {
    display: block;
    padding: 5px;
    background-color: #82aa15;
    color: #fff;
    text-decoration: none;
    margin-bottom: 2px;
}
```

CSS Tip:
The `display: block;` is another new property, it allows you to set the display value of the item, this affects how the item interacts with the flow of the page. There are three main types of `display`;

block: This property disrupts the flow of the page in moves it to a new line, it will also fill the whole width of its container. **Block** elements can be assigned dimensions using the height and width properties. Examples of block elements are `<p>`, `<h1>` and `<ul>`.

inline: this moves the element inline with any other inline elements adjacent to it, for example if we use the code;

```
p { display: inline; }
```

All of the paragraphs `<p>` on our page will flow into one long line of text instead of residing in blocks. **Inline** element cannot be assigned dimensions, examples of inline elements

hidden: this will remove the element from view, the space it occupied will be filled by the next element. For example;

```
p { display: hidden; }
```

This will remove all the paragraphs from our document.

In our theme though we want our links `<a>` to display as a `block` taking up the full width of the navigation wrapper. This also increases the click-able area of the link to the full width.

`Text-decoration` will remove the underline from our links, while `padding` will increase the overall size of our link by 5px, this includes the click-able area. Finally the `margin` of 2px places a little whitespace between out links.

We will now use the :hover state of the link to change some link <a> styles when the users' mouse hovers over the link:

```
#navcontainer a:hover {
    background-color: #e4ee3b;
    color: #82aa15;
}
```

Save your CSS and view your site in RapidWeaver. Try hovering/moving the mouse over the navigation links.

Current Link Styling

As mentioned earlier, RapidWeaver inserts an **ID** attribute with the value **current** into the link <a> for the page the viewer is currently on. We can use this to style the current page link to confer this information visually to the user.

To do this add the following:

```
#navcontainer #current {
    background-color: #e4ee3b;
    color: #82aa15;
}
```

Your theme should now look like the preview below.

Footer Styling

For the final part of this section we will look at how to style the footer. To separate the footer area from the content we'll make the text smaller, centred and add a border to the top, this is achieved like so;

```
#footer {
    font-size: 0.9em;
    text-align: center;
    padding: 10px;
    border-top: 2px solid #82aa15;
}
```

CSS Tip
The border property will add a border around all four sides of your element, there are three values these are width, style and color they can be entered in any order. Width and color should hopefully be self explanatory however, there are many styles of line available, **solid**, **dashed**, **double** and **dotted** for starters. [link: css properties;]

Some CSS properties, namely margin, padding and border allow you to apply styles to just one side of the element instead of all four. This can be achieve by adding -top, -right, -bottom or -left to the end of the property.

For example

```
p {
    padding-top: 10px;
}
```

This will add 10px of padding to the top of all paragraph <p> elements.

The breadcrumb links still require a little styling. The breadcrumb is a trail of links through the site to the page being viewed. For Example if you were filling out a support contact form your breadcrumb might look like this:

Home > Support > Contact

In the same way with the navigation links, the breadcrumb is output as an unordered list. To remove the standard styling from the unordered lists add the following to your CSS;

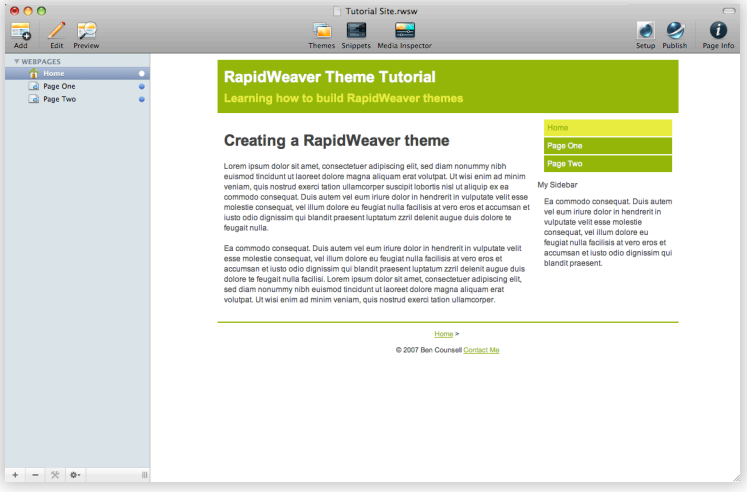
```
#breadcrumbcontainer ul {
    margin: 0;
    padding: 0;
    list-style-type: none;
}
```

Finally make the breadcrumb items display inline:

```
#breadcrumbcontainer li {
    display: inline;
}
```

Note

The separator for the breadcrumb can be defined in the **theme.plist** file. Please refer to the chapter on the .plist in the Advance Tutorials to learn more.



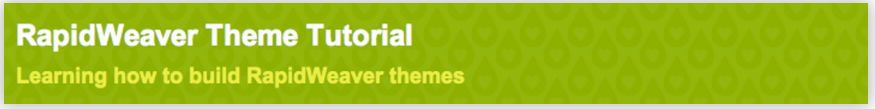
Applying a background image

Using images in your site is an excellent way of adding both texture and visual interest for the user. This final step will look at applying a background image to the header of the site.

Firstly any images you wish to use in the theme should be added to the **images** folder in the theme file. If you look inside the tutorial.rwtheme theme you will see a simple 40x40 png image named **background.png**. We will use this as the background of the header. In styles.css add the following code;

```
#pageHeader {
    padding: 10px;
    background: #82aa15 url(images/background.png) repeat left top;
}
```

Now if you view the page you will see the header has a Realmac themed pattern in the background.



CSS Tip:
You may notice the `background-color` property has merged with the new background property. The background property encompasses all the properties that can be applied to the image. They values are as follows `color`, `url`, `repeat`, `x-position` and `y-position`.

The url is inserted into the brackets, it will always be images/image-name.extention.

CSS Tip (Continued):
The url is inserted into the brackets, it will always be images/image-name.extention.

`repeat`: has 4 possible values;

- `no-repeat`; the image is displayed once.
- `repeat-x`: The image repeat horizontally.
- `repeat-y`: The image repeats vertically.
- `repeat`: The image tiles to fill the entire area.

Finally we have the `position`, this can be either co-ordinates with units such as pixels, ie: 5px 10px will display the image 5px from the left and 10px from the top. Alternatively you can assign the positions,

- `left`, `center` or `right` for the x co-ordinate.
- `top`, `center` or `bottom` for the y co-ordinate.

This tutorial has covered the basics of getting your new theme up and running, you may want to spend some time visiting the global classes such as paragraphs, headers and lists and adjusting the padding and margins, perhaps styling the sidebar differently.

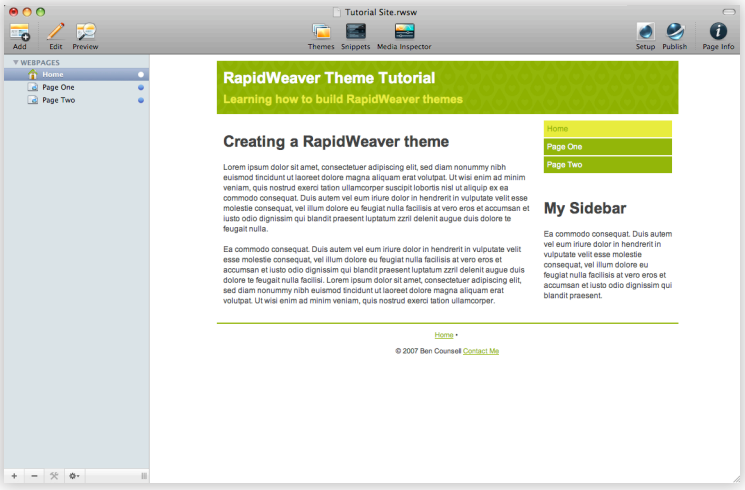
Adding A Theme Preview

Once you've finished your layout you can add a theme preview so that a screen-shot of your site will appear in the Theme Drawer when in RapidWeaver. To do this take a screen grab of your site paste it into a 800x950px document in your image editing software of choice and save it as **preview_large.jpg**, replace the existing file in the theme folder with your newly created one.

You may also wish to create a 115x125px image for use in version 3.6 and under (This should be saved as **preview.png**).

Conclusion

Hopefully by now you should have some understanding of how RapidWeaver themes are constructed. Now you can begin to build your expand your site styling the various elements as you go.



Further reference of RapidWeaver's syntax and CSS styles can be found in the appendix at the end of this document. These include Theme Syntax, Theme.plist Syntax and CSS Styles documents.

If you're feeling a little overwhelmed by this tutorial I would suggest purchasing a good web standards and CSS book. You can also visit our forums at http://www.realmacsoftware.com/support and read through the many user tutorials and of course ask lots of questions.

If you're hungry for more and looking for information on theme and colour variations as well as alternative menu's and the property list, then carry on reading as our advanced tutorials are just around the corner.

Useful CSS Links

- CSS properties: http://www.w3.org/TR/CSS21/propidx.html
- http://www.culturedcode.com/css/reference.html
- Web safe fonts: http://www.ampsoft.net/webdesign-l/WindowsMacFonts.html

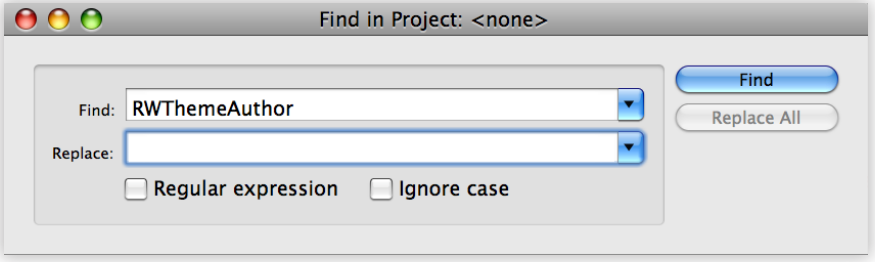
Advanced Tutorials

The Property List
Setting the Theme Author 46
Setting the Theme Name 47
Publishing Short Name 48
Adding Keywords 48
Further Property List Information
Bread-crumb Separator 50
Creating An Horizontal Navigation
Styling the links 54
Styling the Container 54
Adding Theme Variations
Removing the CSS from styles.css 58
Setting up the Property List 58
Creating the Option Group 58
Creating the Variations 59
Adding Colour Picker Functionality
Linking to the colourtag.css 64
Creating New Tags 67
Colour Maths 71
Using Colour Maths in your theme. 71

The Property List

The **theme.plist** file contains basic information for naming your theme, paths and search terms as well as more advanced tools such as Colour Picker integration and Theme Variations. This file is a a large XML file that can be edited with a text editor but it's much easier to use the **Property List Editor** found in the developer tools on your OS X installation disk.

For these tutorials we will edit the XML file in a text editor but the keys are the same in the Property List Editor. Open the **Theme.plist** file in your text editor. Using the search/find function will make things much quicker here. Just press **cmd-F** to open the window.



Setting the Theme Author

I'm sure now you've built a theme yourself you'd like to be credited for your achievement. Open the find window and enter **RWThemeAuthor** then hit find. You should see the following.

```
<key>RWThemeAuthor</key>
<string>Realmac Software SDK</string>
```

Replace `Realmac Software` with your own name. Now if you open the Theme Browser in RapidWeaver and search for your name, your theme will appear.

Note

When saving changes made in the `.plist` RapidWeaver may need to be restarted before changes will take effect.

Setting the Theme Name

Again enter **RWThemeName** into the search field.

```
<key>RWThemeName</key>
<string>Tutorial Theme</string>
```

Replace the contents of the <string> tag with your new theme name.

Currently RapidWeaver has been localised for several languages and each localisation can have its own variation on the theme name. To ensure the theme name displays correctly for your localisation search for RWThemeDisplayName. The following text should appear.

```
<key>RWThemeDisplayName</key>
<dict>
    <key>de</key>
    <string>Tutorial Theme German</string>
    <key>en</key>
    <string>Tutorial Theme</string>
    <key>fr</key>
    <string>Tutorial Theme French</string>
    <key>it</key>
    <string>Tutorial Theme Italian</string>
    <key>ja</key>
    <string>Tutorial Theme Japanese</string>
</dict>
```

Replace each of the <string>'s with your theme name, this can be the same for all or tailored to each language. Now you theme has been localised.

Publishing Short Name

The last area to fix is the name of the folder your theme will be published to. This appears when your site is exported/published. The url will look like this

/rw_common/themes/your-theme-name/

This value should be lowercase and a single word. If you must use two, separate them with an underscore or hyphen.

Search for **RWThemeShortName** and locate the following;

```
<key>RWThemeShortName</key>
<string>tutorialtheme</string>
```

Again replace the <string> with your short name.

Adding Keywords

The **plist** also allows you to add keywords to your theme to make them easier to find when using the search feature in the Theme Drawer. Search for **RWThemeKeyWords** and you should get something similar to this;

```
<key>RWThemeKeyWords</key>
<array>
    <string>realmac</string>
    <string>software</string>
    <string>tutorial</string>
    <string>SDK</string>
    <string>completed</string>
</array>
```

Each string contains one **keyword**, you can have as many or as few as you like. Replace the current entries with your desired keywords, if you wish to add more just add a new line between the <array> tags and insert the new keyword between <string></string> tags. To remove a keyword just delete the line.

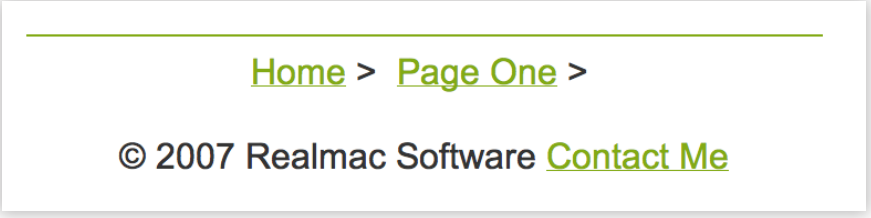
Note

The actual keywords may vary depending on the theme file you are editing. These are the keywords for the Tutorial Completed.rwtheme.

This chapter has covered the basics of personalising the .plist for your theme. It's now ready for you to use in RapidWeaver.

Further Property List Information

Bread-crumb Separator



To change the style of the separator that divides the bread-crumb search for **RWBreadCrumbSeparator**, if you are using the Property List Editor this can be found in the **RWTextToolbar** drop down.

```
<key>RWBreadCrumbSeparator</key>
<string>&amp;nbsp;&gt;&amp;nbsp;</string>
```

This string may look confusing, but all it is is the XHTML way of showing > with a blank space either side. When editing the .plist file any characters like **&, >, <** need to be converted into entities otherwise they will be treated as code and it will cause an error.

Each character has an XHTML equivalent, for example an ampersand (**&**) can be written as `&`
A right angled bracket or greater than symbol (**>**) as `>` This way the entity is treated as text and processed properly.

Finally a **blank space** should be written as ` ` meaning non-breaking-space.

So for example if you wanted a bullet point with a space either side you could use the following XHTML;

```
 &bull; 
```

However for this to work in the .plist the ampersands need to be converted into entities as well. This is referred

Note
A comprehensive list of XHTML entities can be found at http://www.digitalmediaminute.com/reference/entity/index.php.

to as escaping. So the final string will look like.

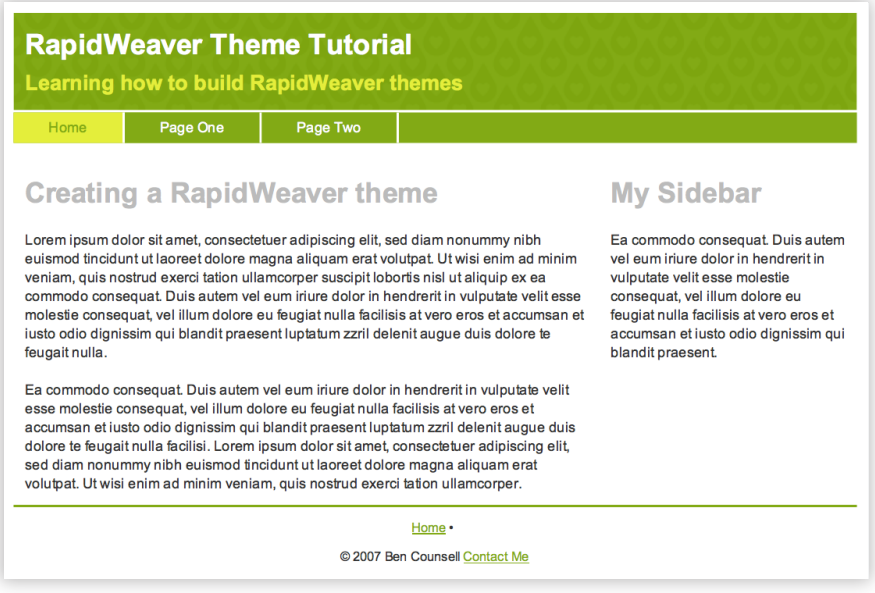
```
<string>&amp;nbsp;&amp;bull;&amp;nbsp;</string>
```

Hopefully you will be using a text editor such as Text Wrangler to edit this code in which case the entities will be highlighted in a different colour to make things easier ot read.

```
<key>RWBreadCrumbSeparator</key>
<string>&amp;nbsp;&amp;bull;&amp;nbsp;</string>
```

Creating An Horizontal Navigation

This tutorial aims to show you how to create a basic horizontal navigation for your site like the one in the following screen-shot.



In order to do this we need to first modify the location of out navigation in the **index.html** file. So open the file in your text editor. Locate the following lines;

```
<div id=”navcontainer”>
   %toolbar%
</div>
```

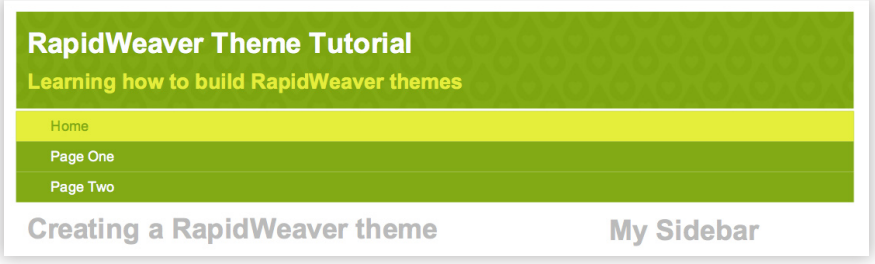
Highlight the three rows and use **edit > cut** command to copy them to the clipboard and remove them from the document.

Now we need to paste them back in just below the header area your file should now look like this, as always modified text is highlighted with an asterisk *.

```
<div id="pageHeader">
    %logo%
    <h1>%site_title%</h1>
    <h2>%site_slogan%</h2>
</div>

<div id="navcontainer">
    %toolbar%
</div>

<div id="contentContainer">
```

I prefer this method of navigation because it places the navigation above the page content. When you read through the page with no styling it makes far more sense to have the site contents listed near the top of the document.

Save **index.html** and view the page in RapidWeaver you will see the navigation has moved out of the sidebar and is now located just below the header. Although the links are listed vertically and stretch across the page.



Styling the links

Open up **styles.css** and add the following to the navigation <li> and <a> styles;

```
#navcontainer li {
    display: block;
    float: left;
}

#navcontainer a {
    display: block;
    padding: 5px 30px;
    background-color: #82aa15;
    color: #fff;
    text-decoration: none;
    float: left;
}
```

Save and preview the changes in RapidWeaver you should see that the the list items are now in one line. To do this we have just floated the anchors <a> and list items <li> to the left.

Styling the Container

We are left with an unattractive white gap to the left of our links. In order to solve this we can set the height of the navcontainer to that of out links and set the same background colour;

Note

You may be wondering why we have to float the anchors if they are inside the floated list items. This is due to an inconsistency with Internet Explorer 6 which will still display the anchors in a vertical list so we add the extra float to get around this.

```
#navcontainer {
    margin-top: 2px;
    height: 26px;
    background-color: #82aa15;
}
```

Finally we can define the links a little better by adding a 2px `border` to the right of all links;

Note

We have removed the 5px padding here and replaced it with an upper `margin` of 2px to seperate the navigation from the header.

```
#navcontainer a {
    display: block;
    padding: 5px 30px;
    background-color: #82aa15;
    color: #fff;
    text-decoration: none;
    float: left;
    border-right: 2px solid #fff;
}
```

There you have it, a simple horizontal navigation bar.

Adding Theme Variations

RapidWeaver offers a really simple method of adding variations to your theme such as sidebar positions, page widths or font families using the Page Inspector. This tutorial will show you how to implement three sidebar variations into your theme, left, right and none.

Creating the CSS files

This feature works by the user selecting the style option, RapidWeaver then inserts an extra stylesheet into the page using the <link> tag. This stylesheet contains **ONLY** the CSS required to change the specific area of the page.

So in order to create a Sidebar Location variation, we must create three new stylesheets, one with the CSS to position the sidebar on the left, one to position the sidebar on the right and one to hide the sidebar from view.

First we will create the file that positions the sidebar on the right, this has already been done so all we have to do is lift the required CSS from **styles.css**. Create a new blank text file and copy the following lines from the **styles.css** into the new document;

```
#contentContainer {
    width: 500px;
    float: left;
}

#sidebarContainer {
    width: 220px;
    float: right;
}
```

Save this file to the desktop and name it **right.css**. Now open up the theme folder in the finder and create a new folder inside the **css** folder named **sidebar**, copy the file into the new sidebar folder.

contents > css > sidebar > right.css

Now to make the sidebar reside on the left. We adjust the CSS in **styles.css** to flat the sidebar to the left and the content on the right.

```
#contentContainer {
    width: 500px;
    float: right;
}

#sidebarContainer {
    width: 220px;
    float: left;
}
```

Preview the changes in RapidWeaver to check that the code has created the desired effect. Then create a new text document and save this as **left.css** and place it into the sideabar folder.

Finally the hidden sidebar, amend the CSS to read

```
#contentContainer {
    width: 100%;
}

#sidebarContainer {
    display: none;
}
```

Note:
If you are using vertical navigation this will also be hidden as it resides inside the sidebar container.

This will set the content container `width` to 100%, and the display none will remove the sidebar container from the page view.

Save this file in the *css* > *sidebar* folder and name it **none.css**.

Removing the CSS from styles.css

Now all that is left is to remove any code relating to the sidebar position from the styles.css file. Amend the following;

```
#contentContainer {
}

#sidebarContainer {
}
```

Setting up the Property List

This is the most complicated part of the process and I heartily recommend you install the Property List Editor in the apple developer tools. For this tutorial we will continue to edit the theme.plist file in a text editor.

Creating the Option Group

Open the themes.plist file and search for RWStyleGroups, for those using the Property List Editor This is located in the **RWStyleVariations** drop down. The first part we need to edit here is the **Option Name**, in this case it's the sidebar position. So locate the following block, just below the **RWStyleGroups** <key>;

```
<dict>
    <key>de</key>
    <string>German Option Group Name</string>
    <key>en</key>
    <string>English Option Group Name</string>
    <key>fr</key>
    <string>French Option Group Name</string>
    <key>it</key>
    <string>Italian Option Group Name</string>
    <key>ja</key>
    <string>Japanese Option Group Name</string>
</dict>
```

Just like re-naming the theme in the **Property List** chapter of this document you need to replace each of the strings with the new option name, in this case “Sidebar Position”

```
<dict>
    <key>de</key>
    <string>Sidebar Position</string>
    <key>en</key>
    <string>Sidebar Position<</string>
    <key>fr</key>
    <string>Sidebar Position<</string>
    <key>it</key>
    <string>Sidebar Position<</string>
    <key>ja</key>
    <string>Sidebar Position<</string>
</dict>
```

The next step is to scroll down and find the following lines about 20 lines down;

```
<key>GroupName</key>
<string>Option Group</string>

<key>GroupSelectionLimit</key>
<integer>1</integer>
```

Rename the string under **GroupName** to `Sidebar Position`. The **GroupSelectionLimit** defines how many options can be selected at once, in this case the default `1` will suffice.

Creating the Variations

Now scroll back up to where you entered the names for the languages. below this you should see a `<key>` named GroupMembers and below this an `<array>` tag. between these array tags you should see two groups of options that look like this;

```
<dict>
    <key>DisplayName</key>
    <dict>
        <key>de</key>
        <string>German Option Name</string>
        <key>en</key>
        <string>English Option Name</string>
        <key>fr</key>
        <string>French Option Name</string>
        <key>it</key>
        <string>Italian Option Name</string>
        <key>ja</key>
        <string>Japanese Option Name</string>
    </dict>
    <key>Enabled</key>
    <true/>
    <key>Files</key>
    <array>
        <string>css/option_group/file_name.css</string>
    </array>
    <key>Name</key>
    <string>Option Name</string>
    <key>Type</key>
    <string>Stylesheet</string>
</dict>
```

This is the code required for one option. So first we need to set up the variable name. In this case ‘Right’, rename all the strings in the DisplayName group to ‘Right’, like so;

```
<key>DisplayName</key>
<dict>
    <key>de</key>
    <string>Right</string>
    <key>en</key>
    <string>Right</string>
    <key>fr</key>
    <string>Right</string>
    <key>it</key>
    <string>Right</string>
    <key>ja</key>
    <string>Right</string>
</dict>
```

Also rename the Name <key> to left also;

```
<key>Name</key>
<string>Right</string>
```

The enabled <key> has a value of either <true/> or <false/> we want this option to be on by default so set leave the selection as <true/>, we must remember to ensure that all future options have this set to <false/>.

```
<key>Enabled</key>
<true/>
```

The last thing to change is to set the destination of our CSS file,

```
<key>Files</key>
<array>
    <string>css/option_group/file_name.css</string>
</array>
```

So amend this string to read;

```
<string>css/sidebar/right.css</string>
```

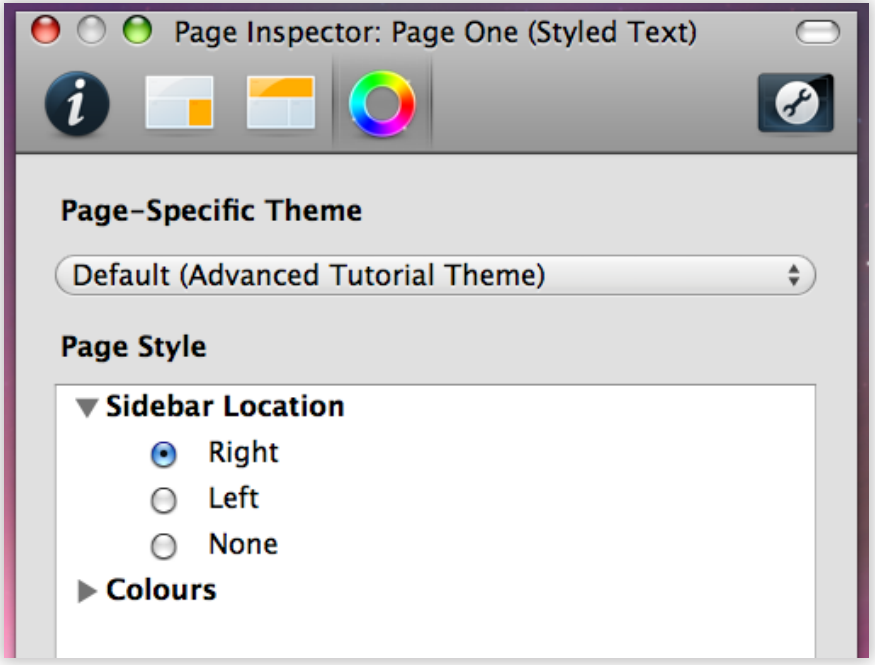
Repeat for each option.

We now have to do this a further two times, for **right.css** and **none.css.** The second option is already inserted in the **theme.plist** file we just need to change the names as before. The finished list should look like this.

```
<dict>
    <key>DisplayName</key>
    <dict>
        <key>de</key>
        <string>Left</string>
        <key>en</key>
        <string>Left</string>
        <key>fr</key>
        <string>Left</string>
        <key>it</key>
        <string>Left</string>
    </dict>
    <key>Enabled</key>
    <false/>
    <key>Files</key>
    <array>
        <string>css/sidebar/left.css</string>
    </array>
    <key>Name</key>
    <string>Left</string>
    <key>Type</key>
    <string>Stylesheet</string>
</dict>
```

Now copy this and paste it just afterwards to create a third entry, rename this for the **none** option.

Save this and view your site in RapidWeaver, open the Theme Variations window and you should see the following;



Now you have added your first theme variation you could try including one for theme widths, or font-families. Alternatively you can move on to the next tutorial where we will be adding colour picker variations to our theme.

Editing the .plist can be daunting at first but after adding your first couple of variations it should become as familiar as HTML.

Adding Colour Picker Functionality

This process is very similar to adding a style variation, although instead of creating multiple stylesheets with our variations. All the colour variations are added to the colourtag.css file and assigned a custom tag;

```
%colour_body_font%
```

These tags are similar to the standard RapidWeaver syntax although you can define them yourself. They must begin with %colour_ and end with a %. Use underscores to separate individual words. We recommend keeping them simple and easy to remember.

```
%colour_links%
```

Linking to the colourtag.css

You may have noticed in the page inspector that there is already a colour option for the background in the Theme Variations tab, however changing this value will have no effect on the page. This is because we have not yet added a link to the **colourtag.css** in out **index.html** file.
[img: sidebar option]
To do this we need to open **index.html** in a text editor and add the following line (highlighted) at the bottom of the list of stylesheets in our <head> area.

```
<link rel="stylesheet" type="text/css" media="screen"
href="%pathto(styles.css)%"  />
<link rel="stylesheet" type="text/css" media="print"
href="%pathto(print.css)%"  />
<link rel="stylesheet" type="text/css" media="handheld"
href="%pathto(handheld.css)%"  />
<link rel="stylesheet" type="text/css" media="all"
href="%pathto(colourtag.css)%"  />
```

Notice the media-”all” attribute this will apply the colours to all media, including the browser, handheld devices and print documents.

Now if you view the site in RapidWeaver you should notice the background has turned grey. Try adjusting the colour picker in the page inspector. The background of the page will change colour.

Let us look more closely at how this works, open the **colourtag.css** file in the Theme Folder. You should see one line.

```
body {
    background-color: %colour_body_background%;
}
```

Here we have the a `body` selector with the `background-color` property, this will fill the entire `<body>` tag with the specified colour value. However instead of a colour or hex value we have a RapidWeaver Syntax Tag. This will link in to our **theme.plist** to generate the menu option in the theme variations.

Open the **theme.plist** file in a text editor, or Property List Editor. Locate the Colours group in the **RWStlyeGroups** array. The colour options will be just below the Theme Variations we entered in the previous chapter. Locate the lines;

```
<key>Name</key>
<string>Body Background</string>
<key>Tag</key>
<string>%colour_body_background%</string>
<key>Type</key>
<string>Colour</string>
```

The name `<key>` Identifies the name of the colour variation in the Page Inspector, this should be localised using the **DisplayName** dictionary in the same way as in the theme variations.

Next `<key>Tag</key>` specifies the tag name for this particular variation.

<key>Type</key> Should always be Colour the other value is Stylesheet when working with theme variations.

Locate the Default colour key, about 10 lines up above the localisation options;

```
<key>DefaultColour</key>
<string>#cccccc</string>
```

Here we can specify the **default colour** for the attribute, change the string back to #ffffff to make the background white by default. Save your changes and restart RapidWeaver, now the background should be white. yet we can change it easily with the colour picker. the final detail if you have been following the tutorials and are using the horizontal navigation. The white borders we added to the left of the navigation links are still white when you change the background colour.

To fix this we open up our **styles.css** file and find the border we added to the navigation links.

```
#navcontainer a {
    display: block;
    padding: 5px 30px;
    background-color: #82aa15;
    color: #fff;
    text-decoration: none;
    float: left;
    border-right: 2px solid #fff;
}
```

We need to separate out our border values into individual attributes.

```
#navcontainer a {
    display: block;
    padding: 5px 30px;
    background-color: #82aa15;
    color: #fff;
    text-decoration: none;
    float: left;
    border-right-width: 2px;
    border-right-style: solid;
}
```

Now back in the **colourtag.css** file add the `border-color` attribute to the `#navcontainer a` selector;

```
#navcontainer a {
    border-right-color: %colour_body_background%;
}
```

Again, instead of the hex value we use the RapidWeaver syntax, as we want it to be the same colour as the background we can use the same tag as for the body.

Creating New Tags

Now you have modified an existing tag let us create one from scratch. This will change the font colour of the text. First lets set up the CSS in **styles.css** locate the color settings in the body tag.

```
body {
    margin: 0;
    padding: 0;
    color: #333;
    font-size: 62.5%;
    line-height: 1.4;
    font-family: Arial, Helvetica, Geneva, sans-serif;
}
```

Delete the colour property...

```
body {
    margin: 0;
    padding: 0;
    font-size: 62.5%;
    line-height: 1.4;
    font-family: Arial, Helvetica, Geneva, sans-serif;
}
```

Now in **colourtag.css** add the color property to the <body> tag, we will also create a new tag called %colour_body_text% and use this as our color value.

```
body {
    background-color: %colour_body_background%;
    color: %colour_body_text%;
}
```

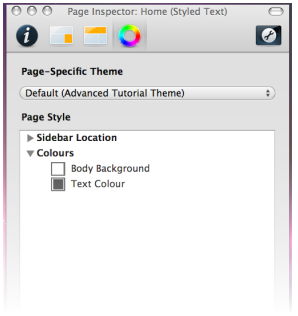
Save these files and open the **theme.plist** file. return the Colours group and copy the body background dictionary and paste it directly below the closing </dict> tag. You will now have two entries in the array.

```
<dict>
    <key>DefaultColour</key>
    <string>#ffffff</string>
    <key>DisplayName</key>
    <dict>
        <key>de</key>
        <string>Body Background</string>
        <key>en</key>
        <string>Body Background</string>
        <key>fr</key>
        <string>Body Background</string>
        <key>it</key>
        <string>Body Background</string>
        <key>ja</key>
        <string>Body Background</string>
    </dict>
    <key>Name</key>
    <string>Body Background</string>
    <key>Tag</key>
    <string>%colour_body_background%</string>
    <key>Type</key>
    <string>Colour</string>
</dict>
```

Rename the <key>Name</key>, <key>Tag</key> and <key>DefaultColour</key> <string>'s to reflect the new tag.

```
<dict>
    <key>DefaultColour</key>
    <string>#333333</string>
    <key>DisplayName</key>
    <dict>
        <key>de</key>
        <string>Text Colour</string>
        <key>en</key>
        <string>Text Colour</string>
        <key>fr</key>
        <string>Text Colour</string>
        <key>it</key>
        <string>Text Colour</string>
        <key>ja</key>
        <string>Text Colour</string>
    </dict>
    <key>Name</key>
    <string>Text Colour</string>
    <key>Tag</key>
    <string>%colour_body_text%</string>
    <key>Type</key>
    <string>Colour</string>
</dict>
```

Save this restart RapidWeaver and you will find a new entry in the colours drop-down entitled Text Colour.



While I have been recommending the use of the Property List Editor application, it is much easier to duplicate theme and colour variation in a text editor and as long as you ensure you only edit the <string>'s and not the <key>'s you should have not trouble adding many variations to your theme.

Colour Maths

Using Colour Maths in your theme.

The final thing we will cover in this advanced section of the tutorial is colour maths, this allows you to modify an existing colour tag by adding, subtracting and multiplying another colour. For example say we want all of our <h1> tags to be a lighter shade of the <body> text.

Instead of creating an entirely new picker we can just apply a white tint to the <body> text picker. Open colourtag.css and if you have followed the previous tutorial you will find the body selector looks like this;

```
body {
    background-color: %colour_body_background%;
    color: %colour_body_text%;
}
```

Let us take the color tag and apply it to our <h1> tags;

```
body {
    background-color: %colour_body_background%;
    color: %colour_body_text%;
}

h1 {
    color: %colour_body_text%;
}
```

This will apply the %colour_body_text% colour to the headers. Now in order to lighten the text we can add white. Colour Maths should be inserted inside the tags between the %.

```
 h1 {
    color: %colour_body_text + #ffffff%;
}
```

This has applied 100% white to the colour so now all your <h1> tags will be white. To reduce this effect we can reduce the amount of white added by using a shade of grey.

```
h1 {
    color: %colour_body_text + #888888%;
}
```

This will add 50% white to the headings. The best way to experiment with the options available is to experiment. There is a full list of the colour math variables in the in the file **Colour Maths** chapter located in the appedix of this document.

RapidWeaver 4

Appendix

Appendix: Table of Contents

Theme Syntax

Header Syntax....84

%header% 84
%title%....84
%user_header%....84
%style_variations%....84
%user_styles%....84
%user_javascript%....84

Body Syntax....85

%site_title%....85
%site_slogan%....85
%logo% 85
%toolbar% 85
%content% 85
%sidebar_title% 85
%sidebar%....85
%plugin_sidebar%....85
%breadcrumb% 85
%footer% 85

Extras....86

%pathto(file.extension)% 86
%base_url%....86
%last_published%....86
%email%....86

Colourtags

%colour...%....86

Theme Styles: Colour Maths

Examples of Colour Maths 88

Top Level Keys 90
ColourTagCSSFiles 90
RWCopyFiles 90
RWExportFiles 91
RWStyleVariations 91
RWTemplateHTML 91
RWTextToolbar 92
RWThemeAuthor 92
RWThemeCapabilities 92
RWThemeDisplayName 93
RWThemeImage 93
RWThemeKeyWords 94
RWThemeName 94
RWThemeShippedInVersion (PRIVATE) 94
RWThemeShortName 95
RWVersion 95
RWStyleVariations 96
RWStyleGroups 96
GroupDisplayName 96
GroupMembers 97
GroupName 97
GroupSelectionLimit 97
Theme Variations 98
DisplayName 98
Enabled 98
Files 98
Name 99
Type 99
Colour Variations 100
DefaultColour 100
This is the default colour to be applied to the specified tag. 100
DisplayName 100
Name 100

Tag 101
Type 101
Example Colour Style Group 101
Example Colour Style 102

RWTextToolBar 103

RWAlwaysDisplayFullNavigation 103
RWBreadCrumb 103
RWBreadCrumbItem 103
RWBreadCrumbItemSeparator 104
RWSupportedLevels 104
RWToolbarItemCurrent 104
RWToolbarItemCurrentAncestor 104
RWToolbarList 105
RWToolbarItemNormal 105

RWThemeCapabilities 106

RWSupportsContentOnlySubPages 106

Theme variations was a major feature added to RapidWeaver 3.2. It allows the users to mix and match pre-set variations in your theme. 106

RWSupportsDisplayName 106
RWSupportsPathTo 106
RWSupportsToolbarRelTag 107

CSS Styles

Global 110

.imageStyle { } 110
.image-left { } 110
.image-right { } 110
.blockquote { } 110
h1 { }, h2 { }, h3 { }, h4 { } 110
p { } 110
code { } 110
ul.disc { list-style-type: disc; } 110
ul.circle { list-style-type: circle; } 110
ul.square { list-style-type: square; } 111
ol.arabic-numbers { list-style-type: decimal; } 111

ol.upper-alpha { list-style-type: upper-alpha; } 111
ol.lower-alpha { list-style-type: lower-alpha; } 111
ol.upper-roman { list-style-type: upper-roman; } 111
ol.lower-roman { list-style-type: lower-roman; } 111

Blog 112

.blog-entry { } 112
.blog-entry-title { } 112
.blog-entry-date { } 112
.blog-entry-summary { } 112
.blog-entry-body { } 112
.blog-entry-comments { } 112
.blog-entry-category { } 112
.blog-entry-permalink { } 112
.blog-image-smiley { } 112
#blog-categories { } 112
.blog-category-link-enabled { } 112
.blog-category-link-disabled { } 113
#blog-archives { } 113
.blog-archive-link-enabled { } 113
.blog-archive-link-disabled { } 113
#blog-rss-feeds { } 113
.blog-rss-link { } 113
.blog-comments-rss-link { } 113
.blog-read-more 113

Blog Archive 114

.blog-archive-background { } 114
.blog-archive-entries-wrapper { } 114
.blog-archive-headings-wrapper { } 114
.blog-archive-month { } 114
.blog-archive-link { } 114
blog-comment-link { } 114
blog-trackback-link { } 114
h1.blog-entry-title { } 114
h1.blog-entry-title a { } 114
p.blog-entry-tags { } 114
p.blog-entry-tags a:link, 114
p.blog-entry-tags a:visited {} 114

p.blog-entry-tags a:hover { } 115
p.blog-entry-tags a:active {} 115
.blog-entry-comments-inline{ } 115
ul.blog-tag-cloud { } 115
ul.blog-tag-cloud li { } 115
ul.blog-tag-cloud li a:link, 115
ul.blog-tag-cloud li a:visited{ } 115
ul.blog-tag-cloud li a:hover { } 115
ul.blog-tag-cloud li a:active { } 115
.blog-tag-size-1 {} 115
... 115
.blog-tag-size-20 {} 115

Contact Form 116

.message-text { } 116
.required-text { } 116
.form-input-field { } 116
.form-input-button { } 116

File Sharing 116

.filesharing-description { } 116
.filesharing-item { } 116
.filesharing-item-title { } 116
.filesharing-item-description { } 116

Photo Album Thumbnails 117

.album-title { } 117
.album-description { } 117
.album-wrapper { } 117
.thumbnail-wrap { } 117
.thumbnail-frame { } 117
.thumbnail-frame img { } 117
.thumbnail-caption { } 117

Photo Album Large View 117

.photo-background { } 117
.photo-navigation { } 117
.photo-title { } 117
.photo-caption { } 117

.photo-links { } 118
.photo-frame { } 118
.exif-data { } 118
p.exif-version { } 118

Movie Album Thumbnails 119

.movie-page-title { } 119
.movie-page-description { } 119
.movie-thumbnail-frame { } 119
.movie-thumbnail-caption { } 119

Movie Pop Up Window 119

.movie-background { } 119
.movie-title { } 119
.movie-frame { } 119

Quicktime 119

.movie-description { } 119
.movie-frame { } 119

RapidWeaver Theme Syntax

Theme Syntax

The following tags are converted by RapidWeaver into the appropriate content. These tags should be added to the template file of your theme, usually index.html.

Header Syntax

`%header%`
This outputs the meta information. It should be placed between the <head> tags in your template file.

`%title%`
Outputs the title of the current page.

`%user_header%`
Allows the user to place any custom code into the header of the page.

`%style_variations%`
Outputs a series of style sheets into the head of the page, as selected by the user via theme styles tab of the Page Inspector.

`%user_styles%`
Outputs the users custom CSS into the head of the users page.

`%user_javascript%`
Outputs the users custom javascript code into the head of the users page.

`%plugin_header%`
Outputs code generated by the selected page type into the header area. Typically it will be javascript and/or css.

Body Syntax

`%site_title%`
Outputs the site title.

`%site_slogan%`
Outputs the site slogan.

`%logo%`
Outputs a user definable site wide image.

`%toolbar%`
Outputs the menu links for the site.

`%content%`
Outputs the content of the current page.

`%sidebar_title%`
Outputs the sidebar title.

`%sidebar%`
Outputs the sidebar content.

`%plugin_sidebar%`
Outputs content specific to the current page style. For example, the blog archives.

`%breadcrumb%`
Outputs a list of links from the home page to the current page.

`%footer%`
Outputs the site's footer.

Extras

`%pathto(file.extension)%`
Used for site consolidation. This can be used for any file (styles.css for example), however the file must be added to the RWCopyFiles array in the theme.plist.

`%base_url%`
Outputs the base URL.

`%last_published%`
Outputs the last published date.

`%email%`
Outputs the user's e-mail address.

Colourtags

`%colour...%`
`eg. %colour_header_background%`
This will be replaced by a colour specified in the theme.plist. This should be placed in an external style sheet. The style sheet should be included in the RWColourTagCSSFiles array in the theme.plist. There is no restriction on how many colour tags you can use. You may also use the same tag multiple times.

Colour Maths

Theme Styles: Colour Maths

Examples of Colour Maths

Colour tags will be replaced by a colour specified in the theme.plist. This should be placed in an external style sheet. The style sheet should be included in the RWColourTagCSSFiles array in the theme.plist. There is no restriction on how many colour tags you can use. You may also use the same tag multiple times.
This tag also supports the Americanized spelling of colour, color.

Colour tags support the following mathematical operators: +, – and *.

```
Example 1: %colour1 + #112233%
Example 3: %colour1 * 0.3%
Example 4: %colour1 + #112233 * 0.3 + #445566%
Example 5: %colour1 + r(0.9) g(0.1) b(0.5)%
Example 6: %colour2 + R(0.9)G(0.1)B(0.5)%
```

Take a look at the other documents included in the SDk for more information on using Colour tags.

RapidWeaver 4

Theme.plist Syntax

Theme.plist Syntax

RapidWeaver uses the theme.plist file to gather all the information it needs about the theme.
The Theme.plist file is a standard xml file, meaning, if you have the developer tools installed, you can use Apple's Property List Editor to create and edit your themes plist.
You can also use a text editor such as TextWrangler or "Text Edit" that comes free with Mac OS X.

These tags are displayed in alpabetical order. Large dictionarys such as **RWStyleVariations** and **RWTextToolBar** can be found in thier own categories.

Top Level Keys

ColourTagCSSFiles

If colour style groups are added RapidWeaver needs to know which file(s) are to be processed for colour tag conversion. Add the following array to the root dictionary.

```
<key>RWColourTagCSSFiles</key>
<array>
   <string>colourtag.css</string>
</array>
```

RWCopyFiles

The RWCopyFiles array is a list of files and folders that are to be copied when the website is published. It should be placed in the root directory.

```
<key>RWCopyFiles</key>
   <array>
      <string>styles.css</string>
      <string>images</string>
   </array>
```

RWExportFiles

The RWExportFiles array is a list of items to copy over and process when the site is built. These files can contain any of the theme syntax tags, which will be replaced with the site values on export. It should be placed in the root directory.

```
<key>RWExportFiles</key>
    <array>
        <string>text.txt</string>
    </array>
```

RWStyleVariations

The RWTemplateHTML key sets the file to use as a template. This is the file that RapidWeaver will use a base for all pages in the site. It should be placed in the root directory.

```
<key>RWStyleVariations</key>
<dict>
</dict>
```

RWTemplateHTML

The RWTemplateHTML key sets the file to use as a template. This is the file that RapidWeaver will use a base for all pages in the site. It should be placed in the root directory.

```
<key>RWTemplateHTML</key>
<string>index.html</string>
```

RWTextToolbar

The RWTextToolbar dictionary contains all the relevant keys to build a toolbar. It should be placed in the root directory.

```
<key>RWTextToolbar</key>
<dict>
</dict>
```

RWThemeAuthor

The RWThemeAuthor key sets an author name for the theme. It should be placed in the root directory.

```
<key>RWThemeAuthor</key>
<string>Realmac Software</string>
```

RWThemeCapabilities

The RWThemeCapabilities key sets which new features are supported by your theme. It is a dictionary that is placed in the root directory.

```
<key>RWThemeCapabilities</key>
<dict></dict>
```

RWThemeDisplayName

Th RWThemeDisplayName should be placed in the root directory and localized for each language you want to support.

```
<key>RWThemeDisplayName</key>
<dict>
    <key>en</key>
    <string>English Theme Name</string>
</dict>
```

RWThemeImage

The RWThemeImage key sets the file to use for the preview in theme drawer. This file should be 115 pixels wide and 125 pixels high. The key should be placed in the root directory.

```
<key>RWThemeImage</key>
<string>preview.png</string>
```

RWThemeImageLarge

The RWThemeImage key sets the file to use for the preview in theme drawer. This file should be 800 pixels wide and 950 pixels high. The key should be placed in the root directory.

```
<key>RWThemeImageLarge</key>
<string>preview_large.jpg</string>
```

RWThemeKeyWords

The RWThemeKeywords adds keywords specific to your theme. It should be added to the root directory. Authors can add as many keywords as they wish, however they should be kept relevant to the theme. Note: It is not necessary to add the author name, theme name, theme display name or theme short name as these are searched for automatically from the existing keys in the theme.plist. Adding such keywords will devalue the theme search feature.

```
<key>RWThemeKeyWords</key>
<array>
    <string>clean</string>
    <string>white</string>
</array>
```

RWThemeName

The RWThemeName key names your theme. Make sure the name doesn't clash with any other theme names. It should be placed in the root directory.

```
<key>RWThemeName</key>
<string>Theme Name</string>
```

RWThemeShippedInVersion (PRIVATE)

The RWThemeShippedInVersion key specifies which version of the RapidWeaver the theme was originally shipped with. This key is private and only used for official RapidWeaver themes.

```
<key>RWThemeShippedInVersion</key>
<string>3.6</string>
```

RWThemeShortName

The RWThemeShortName sets a name used when publishing a site. It should not contain any special characters of spaces. It should be placed in the root directory.

```
<key>RWThemeShortName</key>
<string>supporttheme</string>
```

RWVersion

The RWVersion keys sets a version number for your theme. If two or more themes named the same are installed RapidWeaver will use the most up-to-date version. It should be placed in the root directory.

```
<key>RWVersion</key>
<number>1</number>
```

RWStyleVariations

RWStyleGroups

The RWStyleGroups holds an array of all the style groups associated with the theme. It should be placed inside the RWThemeVariations dictionary.

```
<key>RWStyleGroups</key>
    <array>
        <dict>
        </dict>
    </array>
```

GroupDisplayName

The GroupDisplayName dictionary is located in the RWStyleGroups array and localizes the variation group display names in to each language. A key should be created for each language you wish to support.

```
<key>GroupDisplayName</key>
    <dict>
        <key>en</key>
        <string>English group name</string>
    </dict>
```

GroupMembers

The GroupMembers key holds all the members of each style variation group. It is a key value for each item in the RWStyleGroups array.

```
<key>GroupMembers</key>
   <array>
      <dict>
      </dict>
   </array>
```

GroupName

The GroupName sets a group display name. It is a key value for each item in the RWStyleGroups array.

```
<key>GroupName</key>
<string>Styles</string>
```

GroupSelectionLimit

The GroupSelectionLimit key sets a selection limit for each variation group. It is a key value for each item in the RWStyleGroups array.

```
<key>GroupSelectionLimit</key>
<integer>1</integer>
```

Theme Variations

DisplayName

The DisplayName key is located in each item of the GroupMembers array and localizes the display name for each variation option. Again, a key should be created for each language you wish to support.

```
<key>DisplayName</key>
<dict>
    <key>en</key>
    <string>English variation name</string>
</dict>
```

Enabled

The Enabled key sets whether or not the variation should be selected by default. It is a key value for each member of a style variation group.

```
<key>Enabled</key>
<true/>
```

Files

The Files array gives a list of files to be added when the variation is selected. It is a key value for each member of a style variation group.

```
<key>Files</key>
<array>
<string>css/styles/blue.css</string>
</array>
```

Name

The Name key sets a name for the variation. It is a key value for each member of a style variation group.

```
<key>Name</key>
<string>Support Theme</string>
```

Type

The Type key sets the type of variation. It is a key value for each member of a style variation group. RapidWeaver 3.6 introduced a color type, previously Stylesheet was the only type of variation available.

```
<key>Type</key>
<string>Stylesheet</string>
```

Colour Variations

DefaultColour

This is the default colour to be applied to the specified tag.

```
<key>DefaultColour</key>
<string>#ffffff</string>
```

DisplayName

The DisplayName key is located in each item of the GroupMembers array and localizes the display name for each variation option. Again, a key should be created for each language you wish to support.

```
<key>DisplayName</key>
   <dict>
      <key>en</key>
      <string>Content Background</string>
   </dict>
```

Name

The Name key sets a name for the variation. It is a key value for each member of a style variation group.

```
<key>Name</key>
<string>Support Theme</string>
```

Tag

This is the tag to be used in the colourtag.css file to link the colour picker in RapidWeaver to the stylesheet.

```
<key>Tag</key>
<string>%colour_content_background%</string>
```

Type

The Type key sets the type of variation. In this case **Colour** defines the variation.

```
<key>Type</key>
<string>Colour</string>
```

Example Colour Style Group

To add colour styles to your theme it needs to be added to the RWStyleGroups array in the same way as an original variation group with the exception that GroupSelectionLimit is not required.

```
<dict>
    <key>GroupDisplayName</key>
    <dict>
        <key>en</key>
        <string>Colours</string>
    </dict>
    <key>GroupMembers</key>
    <array></array>
    <key>GroupName</key>
    <string>Colours</string>
</dict>
```

Example Colour Style

To add a colour style you should add the following to the GroupMembers array of the colour style group dictionary. Adding support for localisation is an (recommended) option, more information is available under the 3.5 & above keys section of this document.

```
<key>DefaultColour</key>
<string>#ffffff</string>
<key>DisplayName</key>
    <dict>
        <key>en</key><string>Content Background</string>
    </dict>
<key>Name</key>
<string>Content Background</string>
<key>Tag</key>
<string>%colour_content_background%</string>
<key>Type</key>
<string>Colour</string>
```

Localization was added in RapidWeaver 3.5. For themes to be displayed in each language correctly. The following additions have been made to the theme.plist.
Note: The following languages are official supported English (en), French (fr), German (de), Japanese (ja) and Italian (it).

RWTextToolBar

RWAlwaysDisplayFullNavigation

The RWAlwaysDisplayFullNavigation key set whether the entire menu should be output on every page or if only the current page's child items should be displayed. It is a key value of the RWTextToolbar dictionary.

```
<key>RWAlwaysDisplayFullNavigation</key>
<false> or <true>
```

RWBreadCrumb

The RWBreadCrumb key sets the output for the opening and closing tags of an unordered list for bread-crumb navigation. It is a key value of the RWTextToolbar dictionary.

```
<key>RWBreadCrumb</key>
<string><ul>%items%<ul></string>
```

RWBreadCrumbItem

The RWBreadCrumbItem key sets the output for each item in the unordered list for bread-crumb navigation. It is a key value of the RWTextToolbar dictionary.

```
<key>RWBreadCrumbItem</key>
<string><li><a href="%url%">%title%</a>%separator%</li></string>
```

RWBreadCrumbItemSeparator

The RWBreadCrumbItemSeparator key defines the characters used to seperate the list of breadcrumb items. If you are using special characters here it is best to escape them to XHTML entities.

```
<key>RWBreadCrumbItemSeparator</key>
<string> > </string>
```

RWSupportedLevels

The RWSupportedLevels key sets how many levels of navigation the theme officially supports. It is a key value of the RWTextToolbar dictionary.

```
<key>RWSupportedLevels</key>
<integer> 5 </integer>
```

RWToolbarItemCurrent

The RWToolbarItemCurrent sets the output for the current page in the unordered list for main navigation. It is a key value of the RWTextToolbar dictionary.

```
<key>RWToolbarItemCurrent</key>
<string> <li><a href=”%url%” id=”current”>%title%</a></li> </string>
```

RWToolbarItemCurrentAncestor

The RWToolbarItemCurrentAncestor key styles each parent level of the menu with a currentAncestor CSS class tag. It is a key value of RWTextToolbar.

```
<key>RWToolbarItemCurrentAncestor</key>
<string>
   <li><a href=”%url%” rel=”%rel%” class=”currentAncestor”>%title
%</a>%subitems%</li>
</string>
```

RWToolbarList

The RWToolbarList key sets the opening and closing tags of an unordered list for main navigation. It is a key value of the RWTextToolbar dictionary.

```
<key>RWToolbarList</key>
<string><ul>%items%<ul></string>
```

RWToolbarItemNormal

The RWToolbarItemNormal sets the output for out each item in the unordered list for main navigation. It is a key value of the RWTextToolbar dictionary.

```
<key>RWToolbarItemNormal</key>
<string> <li><a href=”%url%”>%title%</a></li> </string>
```

RWThemeCapabilities

RWSupportsContentOnlySubPages

The RWSupportsContentOnlySubPages key is located in the RWThemeCapabilities dictionary and allows RapidWeaver to create content only sub pages. Used for coherent site design. It is part of the RWThemeCapabilities dictionary.

```
<key>RWSupportsContentOnlySubPages</key>
<true/>
```

Theme variations was a major feature added to RapidWeaver 3.2. It allows the users to mix and match pre-set variations in your theme.

RWSupportsDisplayName

The RWSupportsDisplayName key is located in the RWThemeCapabilities dictionary and tells RapidWeaver that the theme supports localized display names. It is part of the RWThemeCapabilities dictionary.

```
<key>RWSupportsDisplayNames</key>
<true/>
```

RWSupportsPathTo

The RWSupportsPathTo key adds support for site consolidation. It is part of the RWThemeCapabilities dictionary.

```
<key>RWSupportsPathTo</key>
<true>
```

RWSupportsToolbarRelTag

The RWSupportsToolbarRelTag key adds support for the “rel” tag being added to toolbar link items. It is part of the RWThemeCapabilities dictionary.

```
<key>RWSupportsToolbarRelTag</key>
<true>
```

CSS Selectors

CSS Styles

Below is a full list of CSS classes with descriptions that are used by RapidWeaver to style each plugins XHTML output.

Global

`.imageStyle { }`
Styles the images within the content area of any page.

`.image-left { }`
Left align the image.

`.image-right { }`
Right align the image.

`.blockquote { }`
Should be used to make blocks of text standout.

`h1 { }, h2 { }, h3 { }, h4 { }`
Style any text with a heading tag around it.

`p { }`
Style any text with a paragraph tag around it.

`code { }`
Style any text with a code tag around it.

`ul.disc { list-style-type: disc; }`
Styles the disc list type

`ul.circle { list-style-type: circle; }`
Styles the circle list type

```
ul.square { list-style-type: square; }
```
Styles the square list type

```
ol.arabic-numbers { list-style-type: decimal; }
```
Styles the arabic numbers list type

```
ol.upper-alpha { list-style-type: upper-alpha; }
```
Styles the upper alpha list type

```
ol.lower-alpha { list-style-type: lower-alpha; }
```
Styles the lower alpha list type

```
ol.upper-roman { list-style-type: upper-roman; }
```
Styles the upper roman list type

```
ol.lower-roman { list-style-type: lower-roman; }
```
Styles the lower roman list type

Blog

`.blog-entry { }`
Styles the whole entry.

`.blog-entry-title { }`
Styles the title of each entry.

`.blog-entry-date { }`
Styles the date of each entry.

`.blog-entry-summary { }`
Styles the summary of each entry.

`.blog-entry-body { }`
Styles the main body of each entry.

`.blog-entry-comments { }`
Styles the comments at the end of each entry.

`.blog-entry-category { }`
Styles the category link of each entry.

`.blog-entry-permalink { }`
Styles the permalink link of each entry.

`.blog-image-smiley { }`
Styles the smiley images within the blog.

`#blog-categories { }`
Wrapper for the blog category side links.

`.blog-category-link-enabled { }`
Styles the category links when a post has been made in the category.

`.blog-category-link-disabled { }`
Styles the category links when no posts have been made in the category.

`#blog-archives { }`
Wrapper for the blog archive side links.

`.blog-archive-link-enabled { }`
Styles the archive links when a post has been made in that date range.

`.blog-archive-link-disabled { }`
Styles the archive links when no posts have been made in that date range.

`#blog-rss-feeds { }`
Wrapper for the blog RSS links.

`.blog-rss-link { }`
Styles the RSS feed link.

`.blog-comments-rss-link { }`
Styles the RSS comments link.

`.blog-read-more`
Styles the read more links.

Blog Archive

`.blog-archive-background { }`
Styles the background of the archive page.

`.blog-archive-entries-wrapper { }`
Styles around all entries.

`.blog-archive-headings-wrapper { }`
Styles around the archive header.

`.blog-archive-month { }`
Styles the date heading.

`.blog-archive-link { }`
Styles the month links.

`blog-comment-link { }`
Styles the comment link at the bottom of each blog post.

`blog-trackback-link { }`
Styles the trackback link at the bottom of each blog post.

`h1.blog-entry-title { }`
Styles the title of each entry.

`h1.blog-entry-title a { }`
Styles the title permalink of each entry.

`p.blog-entry-tags { }`
Paragraph class for tags in the main entry area.

```
p.blog-entry-tags a:link,
p.blog-entry-tags a:visited {}
```
Styles blog tag links normal and visited state

```
p.blog-entry-tags a:hover { }
```
Styles blog tag links hover state

```
p.blog-entry-tags a:active {}
```
Styles blog tag links active state

```
.blog-entry-comments-inline{ }
```
Styles inline comments on the main entry page

```
ul.blog-tag-cloud { }
```
Styles the tag cloud un-ordered list

```
ul.blog-tag-cloud li { }
```
Styles tag cloud list item

```
ul.blog-tag-cloud li a:link,
ul.blog-tag-cloud li a:visited{ }
```
Styles all links in the tag cloud in normal and visited state

```
ul.blog-tag-cloud li a:hover { }
```
Styles all links in the tag cloud in hover state

```
ul.blog-tag-cloud li a:active { }
```
Styles all links in the tag cloud in active state

```
.blog-tag-size-1 {}
...
.blog-tag-size-20 {}
```
Styles the sidebar tag sizes from 1 to 20

NOTE: 3.6 and above Blog Styles
The blog permalink has been removed from the date line and the link placed around the blog entry title. The blog entry title has also been placed in a header 1 tag. Therefor the .blog-entry-permalink style has been depreciated and the following styles added.

Contact Form

`.message-text { }`
Styles the messages that appears at the top of the form.

`.required-text { }`
Styles the required text.

`.form-input-field { }`
Styles the text fields.

`.form-input-button { }`
Styles the form buttons.

File Sharing

`.filesharing-description { }`
Styles the page description.

`.filesharing-item { }`
Styles around each entry.

`.filesharing-item-title { }`
Styles the title of each item.

`.filesharing-item-description { }`
Styles the description for each item.

Photo Album Thumbnails

`.album-title { }`
Styles the album title.

`.album-description { }`
Styles the album description.

`.album-wrapper { }`
Style wrap around all thumbnails.

`.thumbnail-wrap { }`
Style wraps around each album entry. Used for positional and alignment purposes.

`.thumbnail-frame { }`
Styles around the image and caption.

`.thumbnail-frame img { }`
Styles the image.

`.thumbnail-caption { }`
Styles around the thumbnail.

Photo Album Large View

`.photo-background { }`
Styles the body of the page.

`.photo-navigation { }`
Styles around the navigation.

`.photo-title { }`
Styles the photo title.

`.photo-caption { }`

Styles the photo caption.

`.photo-links { }`
Styles the links in the navigation area.

`.photo-frame { }`
Styles the photo.

`.exif-data { }`
Wrapper style for the EXIF date.

`p.exif-version { }`
The remain styles are self explanatory.

Movie Album Thumbnails

`.movie-page-title { }`
Styles the page title.

`.movie-page-description { }`
Styles the page description.

`.movie-thumbnail-frame { }`
Styles around the movie thumbnail.

`.movie-thumbnail-caption { }`
Styles the movie thumbnail caption.

Movie Pop Up Window

`.movie-background { }`
Styles the body of pop-up movie page.

`.movie-title { }`
Styles the movie title.

`.movie-frame { }`
Styles around the movie. * Shared with the standard QuickTime page.

Quicktime

`.movie-description { }`
Styles the movie title.

`.movie-frame { }`
Styles around the movie. * Shared with the standard QuickTime page.