Table of Contents

Introduction

WebMaker creates standards based web forms and applications based on HTML, CSS and optional Javascript. The WebMaker Studio has been designed and tested to ensure that the default palette UI controls will operate correctly across all commonly used Browsers: Internet Explorer 6+, Firefox 3.5+, Chrome, Safari 3+ and Opera. Other standards based Browsers should also operate correctly. This approach significantly reduces the workload for forms and application developers if the WebMaker design approach is exploited fully.

This FAQ aims to educate web designers and developers on the design principles that have been used to create the default WebMaker demo styling. This will empower designers and users with the knowledge of the recommended approaches to creating application or customer specific "look and feel" themes, and still preserve the WebMaker User Interface Design Principles. Accompanying this document is an alternative theme css with supporting images, plus an example "**Theme Design Template**" project that can be used to quickly test the majority of the various combinations of UI layout that can be rendered by WebMaker.

We recommend WebMaker users "clone" the default "**demo.css**" file to create their own look and feel, and test the revised styles within the example project. This approach means customers can get their look and feel completed, before rapidly constructing multiple forms and applications using the new CSS. There are further details on these steps in the last section of this document.

User Interface Design Principles

The web applications rendered by WebMaker with HTML and CSS, have been designed based on two key principles:

- Fluid / Adaptive Design
- Progressive Enhancement

Fluid / Adaptive Design approach means that the web pages can be rendered across the greatest number of screen resolutions, including mobile devices, without necessarily needing specific implementations.

Progressive Enhancement means that the web pages by default will render accessibility compliant standard HTML on all Browsers, but will enhance the user experience with a layer of more advanced look and feel characteristics if the user's Browser supports particular capabilities. This allows the web design to adapt and take advantage of features such as CSS3/HTML5.

Both of these approaches are generally recognised as "best practice" within web design communities.

The Web Page Building Blocks

The WebMaker Design Studio creates all the HTML web page controls based on a number of key product constructs:

- Groups
- Fields
- Repeats

We will examine the details of each of these structural building blocks in the following sub sections. Each detailed section will highlight how the HTML is rendered, and how CSS classes are applied.

CSS and HTML Markup Introspection Tools

WebMaker delivers thin and standard web pages. Web page content introspection tools such a **Developer Tools** in IE or **FireBug** in FireFox can be used to check the content of every page created and delivered by WebMaker.

CSS Files

WebMaker applications have a default master CSS file, which is named "**demo.css**" and located under "**Advanced Options**" on the left had panel of the FormMaker Application Map tab. You can either change or replace this file or add additional CSS files. These files are applied to all pages within an application.

If you want to control or override the styling for a specific page then you can highlight the page and add one or more CSS files under the "**Advanced Options**" section on the right hand panel within the Application map tab. You can access and edit all CSS files applicable to a particular page from within the "**Fields Details**" tab.

Building Block - Groups

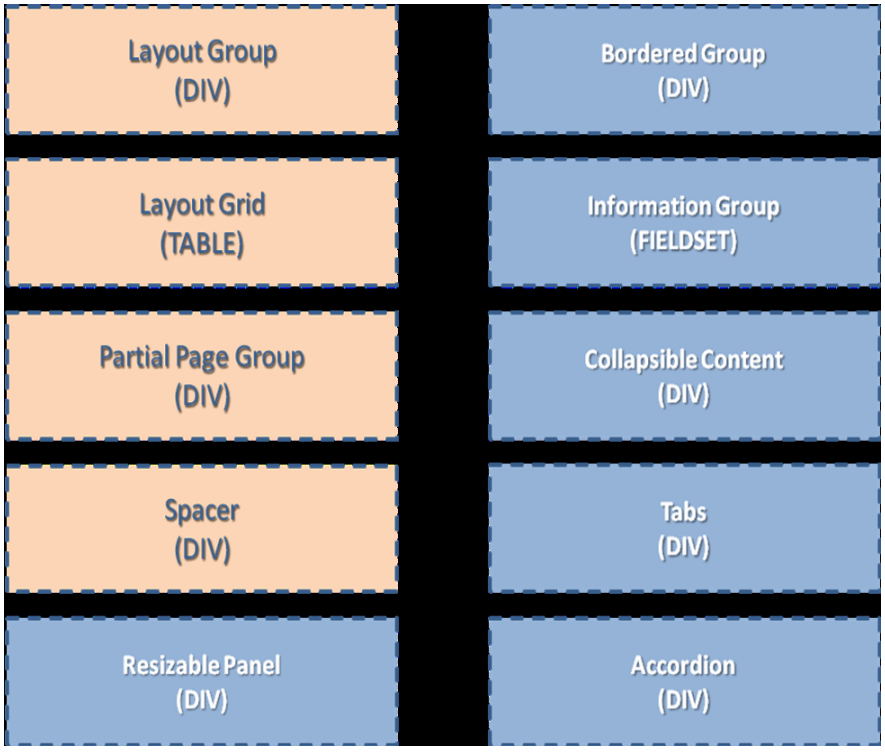
Groups are used to control the layout and positioning of content (Groups, Fields and Repeats), and sometimes apply background styling for a collection of content. Depending on the type of Group used from the standard WebMaker Palette on the Page Structure screen, and the options set on the Field Details screen, the HTML will be rendered differently, to allow flexibility in layout design.

*Note: It is highly recommended that all Groups are named appropriately to their usage once they are dragged onto a page. Simply open the "**Details**" link for the group on the "**Page Structure**" screen, double-click on a Group name in order to alter the value. The other name displayed initially is the Label associated with the group, but that is optionally displayed on forms.*

Types of Group

WebMaker provides a number of Groups that have a primary role of controlling only layout/organisation e.g. Layout Group (see cream boxes on diagram below). These are important building blocks to control layout, as they do not provide any look and feel aspects. They are often used to control whether fields are conditionally displayed or not or even disabled, or changing the display method of the content of the group. Any Group can be nested in another Group, so there is a lot of flexibility to get the exact layout required. The most important consideration is that these structural / layout type Groups do not take up any space on the rendered page. There is no padding or margins associated with these Groups.

Groups that render look and feel (see blue boxes on diagram below), add styling such as background colours and images, borders, titles, padding, margins and so on.



All the various Groups render various types of HTML structure; a **DIV** / **SPAN** container, or a **TABLE** container with Header and Detail rows and cells.

All Groups have a "**Display Method**" indicator that controls the HTML that is rendered. The options available and the associated rendered HTML are as follows: "**horizontal**" (**SPAN**), "**vertical**" (**DIV**), "**table**" (**TABLE**), "**horizontal_table**" (**DIV**) and "**grid**" (**TABLE**).

The "**force table layout**" will also impact the HTML rendered. In all cases the HTML is rendered as an HTML **TABLE** with a class of "**force**" on the TABLE element.

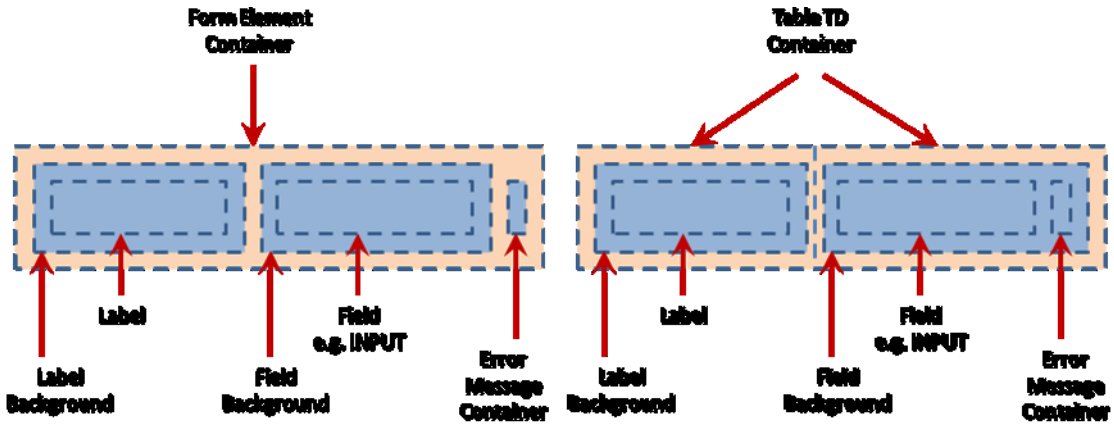
Display Method	Force table layout - false	Force table layout - true
Horizontal	span	table
Vertical	div	table
Table	table	table
Horizontal table	div	table
grid	table	table

Note: Unless it is in a repeat, the horizontal table option is exactly the same as the vertical option.

It is very important to consider these variations in the HTML rendered, based on the setting of various capabilities within FormMaker. It is important that any styling changes considered are aware of the variations in HTML rendered depending on the type of Group. The default "**demo.css**" has various class definitions that have a number of CSS selectors that render the style.

Layout Group Details

When each option is selected in WebMaker via the palette or the "**Field Details**" screen, the HTML will be rendered with the addition of a specific class name for the property e.g. "**vertical**". The diagram below illustrates a couple of examples of the Group "**Display Method**" settings and the HTML rendered, and how further building blocks are nested. The left hand-side is a "**vertical**" group, and the right hand-side is a "**vertical**" with "**force**" set.




```
<DIV class="vertical">
    <DIV class="formElement">
        <DIV class="labelBackground">
            <LABEL class="label"/>
        </DIV>
        <DIV class="defaultBackground">
            <INPUT class="textbox"/>
        </DIV>
        <SPAN id="...message_container">
        </SPAN>
    </DIV>
    . . .
    . . .
```

```
<TABLE class="vertical force">
  . . .
  <TBODY>
    <TR>
        <TD class="labelBackground">
            <LABEL class="label"/>
        </TD>
        <TD class="defaultBackground">
            <INPUT class="textbox"/>
            <SPAN id="...message_container"/>
        </TD>
    </TR>
    . . .
```

It is possible to switch a "**horizontal**" (**SPAN**) or a "**vertical**" (**DIV**) container into an HTML **TABLE** layout by setting the "**Force Table Layout**" option on the Group. This approach allows the columns of the table to be automatically resized if for example, a Field's width needs to change due to the actual data rendered. By default table columns of data expand in size based on the data received. This can be controlled if required, by adding a fixed "**width**" value to a "**Label Background**" or "**Field Background**" within the appropriate "**CSS override**" on the "**Field Details**" screen. There are further specific details about Labels and Fields below.

Note: It is very important to learn the various options in order to simplify the organisation of fields. These options are very powerful, but the effect of the options needs to be understood to avoid unnecessary overriding of lots of elements.

Look and Feel Group Details

The Groups that do have look and feel aspects tend to have a Group Label by default, which takes its value from the Group name by default. There is a "**Do you want a group label?**" tick box associated with all Groups, which dictates whether the Label related HTML is rendered. For example, a "**Bordered Group**" would render a **DIV** as the Label is set on by default, and is rendered with an associated class of "**groupLabelBackground**". This then contains a nested **H2** element with a class of "**label**" for the actual Group name text.

These groups do have some padding to ensure that there is a small gap at the top and bottom of the Group container. This padding is exactly the same depth as applied to individual Fields for consistency, and to ensure that nested fields and layout groups are rendered consistently.

If "**Information Group**" (**FIELDSET**) is selected, then a **FIELDSET** HTML control container is rendered for the Group. Nested within the Group is a **LEGEND** element for the "**labelBackground**", with a nested **SPAN** for the actual label text. The **SPAN** contains the actual Fieldset Group name label text.

There are some Groups that have a specialised use, and are usually parts of larger "**Composite Controls**" e.g. Accordion Pane within an Accordion Container. Details of these Group containers can be found on the section on "**Composite Controls**" towards the end of this document.

Additional Group Styles

There are various situations when you may wish to control Groups in some manner. It is recommended that where possible class names are used in order to apply a given capability, as these classes can easily be applied on a "**conditional**" basis within WebMaker.

Note: It is also very important to remember that multiple class names can be applied to a Group .e.g. "**vertical stretchedGroup roundedBox**". *Each must be separated with a blank, and will be applied one after the other.*

To assist users, there are a number of predefined classes available in the demo CSS:

- **stretchedGroup** – This will apply a 100% width in order to "stretch" the group to fill the container it is nested within, or across the whole page if there is no container.

- **roundedBox** – This applies rounded corners to all four corners of a box, such as a "**BorderedGroup**". This style applies CSS3 based styling that will work with modern browsers supporting HTML5, but will not be applied for non standard Browsers: IE 6, 7 and 8. This styling approach does not require any additional HTML, and all the look and feel is controlled with advanced CSS3 syntax. This design choice would mean that the IE pages don't look as polished. This approach can significantly reduce the web designer's activities that typically create lots of styled images. However, if this approach is not acceptable then there is a Group option that will render additional HTML for a group of images around a group – see "**corners**" below.

- **corners** – This style applies HTML and CSS2.1 based styling that will work with all Browsers including IE 6, 7 and 8. The class name should be added to the Group, and the "Extended Border Styling" option ticked. When this option is ticked WebMaker renders lots of additional HTML around the group that then allows 9 different images to be used to create the look of a "**roundedBox**".
  If you want to change these styles, then you will also need to create the 9 appropriate images. For further details, please refer to the FAQ in the community forum.

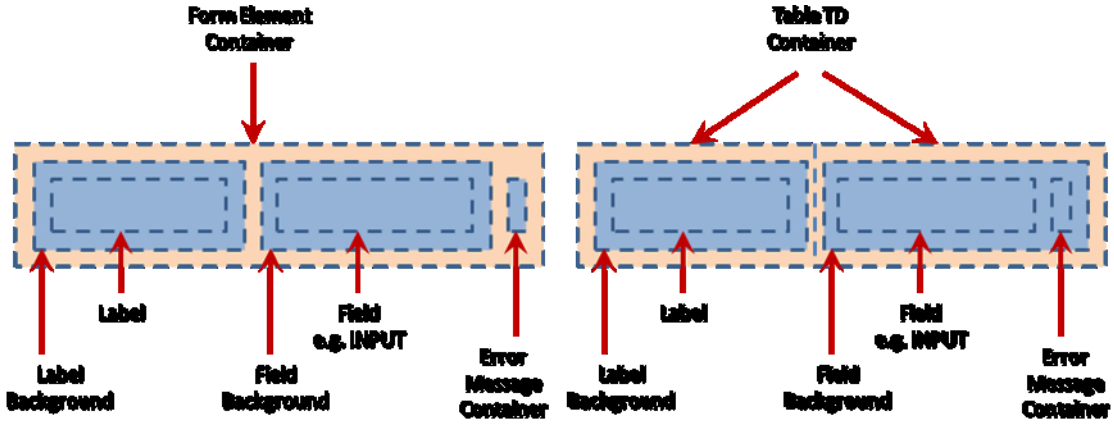
- **outline** – This style applies a border around the Group. This can be applied to allow a border around Fields that are nested within a Group whether it is a **DIV**, **SPAN** or **TABLE**.
- **hide / hidden** – This style will hide a Group from being displayed on the Browser.

Building Block - Fields

Fields are the lowest level of UI controls on a web page. All the standard form controls can be found on the WebMaker Palette on the Page Structure screen. The field controls are actually rendered as a number of HTML elements in order to provide the greatest flexibility in how they are rendered, and with various Field features enabled.

Field Containers

The illustration below shows the most commonly rendered HTML if a field is dragged onto a standard "**vertical**" or "**...table**" Group on a page. Depending on some of the WebMaker options used, some of the HTML elements may not be rendered. These variations will be discussed later.




```
<DIV class="vertical">
    <DIV class="formElement">
        <DIV class="labelBackground">
            <LABEL class="label"/>
        </DIV>
        <DIV class="defaultBackground">
            <INPUT class="textbox"/>
        </DIV>
        <SPAN id="...message_container">
        </SPAN>
    </DIV>
    . . .
    . . .
```

```
<TABLE class="vertical force">
  . . .
  <TBODY>
    <TR>
        <TD class="labelBackground">
            <LABEL class="label"/>
        </TD>
        <TD class="defaultBackground">
            <INPUT class="textbox"/>
            <SPAN id="...message_container"/>
        </TD>
    </TR>
    . . .
```

If a field is placed into a standard Group on a page such as a Layout Group (**DIV**) or Bordered Group (**DIV**), then a "**formElement**" container is rendered that encompasses the Field Label and Field Control. This container exists to allow the label and field control elements to be aligned appropriately depending on the "**Label position for contained components**" definition on the Group container.

Labels by default have been set with a fixed width of **150px** (pixels), so they will wrap the Label text if it is too long and ensure that the labels and fields are aligned correctly in a **DIV** based group. The width can be overridden by applying a CSS Override to the "**labelBackground**" if required. Switching the "**force**" option on the Group can be useful if you want a table structure and want to control the width of a number of Labels together. The width can be overridden on a single field rather than all

the "**labelBackground**" CSS override values. So, you could place a longer width on perhaps the first fields "**labelBackground**" within a Group.

The Group container approach allows the Labels to be positioned relative to the Field Control by setting the "**Label position for contained components**" option to "**Left**", "**Right**", "**Above**" or "**Below**" the fields. This method ensures that a number of Label/Fields within a Group container can be aligned in a consistent manner, whether **DIV**, **SPAN** or **TABLE** HTML is rendered.

If a Group container is rendered as a **TABLE**, then the rendering order is changed for the table. If a Group container is not rendered as a **TABLE**, then each "**Form Element Container**" has a specific CSS class of "**formElement**". However, the actual Label and Field Background HTML elements rendered will vary depending on the "**Label position for contained components**" value. The HTML created would be a **SPAN** if the "**Left**" or "**Right**" options are selected, but a **DIV** if the "**Above**" or "**Below**" options were selected. This is because a **DIV** is defined as a Block element, and so it is moved to a new line on the page when it is rendered, whereas a **SPAN** is an inline element, so stays on the same line.

*Note: The "**formElement**" class does not provide any look and feel styling; it only contains positional padding styling. This ensures that the labels and fields are spaced in a consistent manner.*

Labels

WebMaker does not contain a separate Label control. Most field controls contain a label by default, which can be shown or hidden as required. Simply right click on the label on the "**Page Structure**" screen to switch labels on/off. This approach prevents the need to paint and align separate label controls.

Standard Field labels will be rendered using the specific **LABEL** HTML element. For each label there is a label container with class name of "**labelBackground**", and then the **LABEL** element with a class name of "**label**". This provides greater flexibility of styling as the visual effects can be separated between the background and the control itself. The label background will be a **SPAN** or **TD** container depending on the "**Display Method**" of the containing Group.

If you want to override the standard width of a label, which is defaulted to 150 pixels, then set the width you want on the "**CSS Override**" for the "**Label Background Style**". There is a mini-palette on the CSS override to assist with these settings. If you require a different width for a number of fields on a frequent basis, then you may prefer to add another class onto the "**Label Background Style**" field that would override the width as required.

If you want to change something like the specific font characteristic for an individual Label, then you should apply a "**CSS Override**" to the Label Control Styling e.g. "**font-weight:bold;**". There is a mini-palette on the "**CSS Override**" to assist with these settings.

*Note: Group labels are rendered using different HTML depending on the type of Group. If a Group is "**horizontal**" or "**vertical**", then there will be a **DIV** for the "**groupLabelBackground**" class, and a nested **H2** element will be rendered rather than a **LABEL** element.*

Adding different label styles to labels

By default the Label class name is "**label**" and the background is "**labelBackground**". However, there are a number of specific context label classes that can be used in order to override default classes allocated.

These are: "**fieldLabel**", "**groupLabel**" and "**repeatLabel**" along with a "**fieldLabelBackground**", "**groupLabelBackground**" and a "**repeatLabelBackground**". These additional specialised label class names mean that they can be used to override the standard label look on different controls if required. This can be very useful to create different layout effects, especially when nesting controls.

For example, you can apply a "**repeatLabelBackground**" (typical Paging Table Heading) onto a standard Field label, which is normally defaulted to "**labelBackground**". If this approach is to be adopted on a control, then the existing style names should be replaced.

Field Controls

There are various specific types of Field controls used for different form features that can be found on the Palettes. Some of these controls are implemented as **INPUT** elements, and others include specific form controls such as: **SELECT**, **RADIO**, **CHECKBOX**, **TEXTAREA**, etc. WebMaker always uses the HTML element that is most semantically appropriate. You will also observe that any Label is semantically referenced to the Field it is used for. This enables speech based browsers to understand the appropriate association of the HTML elements.

Each of the field controls has specifically associated style class name, which is relevant to the Palette definition used. However, it is possible to add additional style names if required, to create far greater variation depending on the exact Field look and feel required.

The Field control background containers are similar to the Labels described above, in that the container may be rendered as a **DIV**, **SPAN** or **TABLE** > **TR** > **TD** depending on the definition of the Group that the fields are contained within.

There are explicit style class names of "**hidden**" and "**disabled**" to control whether a field is visible and editable respectively. These are defaulted by WebMaker if specific features are used that "**conditionally**" apply these styles.

There are some field controls that have some specific rendering characteristics to be aware of:

- **Output** – An output only field is defined with a **SPAN** for the output text, and a class name of "**output**".
- **Radio and Multiple Checkbox** – These controls create a number of **INPUT** elements to display each of the available options. If either the number of columns or number of rows option is specified, then a **TABLE** structure will be used to achieve the required layout. Otherwise the **INPUT** elements will be contained within a set of **SPAN**s. Regardless of the structure, the container for each **INPUT** will have a class name of "**selectBoolean**" and the container for each associated display value will have a class name of "**selectBooleanCaption**".
- **Multi-Select Box** - The Multi-Select Box has an additional Control style of "**multiSelect**".
- **Filtering Select** – This is a dojo control that adds an additional background class "**filteringSelect**" for the field in order to differentiate it from a standard **SELECT** box.
- **Date** – The date has an additional calendar popup control option, which has a specific icon associated with a calendar selection. It is controlled by a further **A** (anchor) HTML element within the Field background container. The class associated with the calendar container is "**datePickerIcon**". But this can be changed for a field if required on the Field Details page.
- **Button** – A standard HTML button is defined as an **INPUT** element with a specific type of "**button**".
- **Anchor and Enhanced Button** – Both these controls are rendered as an HTML **A** (anchor) element, which includes two nested **SPAN** elements. These nested **SPAN** elements provide greater flexibility to add visual effects. For example, the Enhanced Button can use one, two or three separate Background images to provide the very specific look for the buttons (left image, background central image and a right image). The set of associated **A.button** styles can be cloned to create other variants of buttons if required. The key principle would be to define a new style name for each specific purpose e.g. "**save-icon-button**" for a button that shows a file saving image on the actual button.
- **Paragraph -** A Paragraph control is a simple HTML **P** element, which is the container for the paragraph text. The CSS styles are applied to the **P** element directly. The control will be extended to fill the same space as both the Label and Field control within Group containers. The Paragraph does not have an additional container class like most other controls, as the **P** element is the container for other nested HTML elements created by the Paragraph text editor.
- **Custom Control** – A Custom Control has a simple **DIV** container with a style class of "**customControl**". The nested content will be at the users control as it contains specific manually created HTML.

Dojo Extensions

Fields with Dojo custom attributes defined will actually be rendered quite differently within the Browser, as Dojo applies javascript to perform conversions in order to enable some advanced visual capabilities. For example, the Currency Amount palette entry sets some dojo custom attributes that control the visual display of the numeric amount if script is enabled.

Therefore, there is additional HTML rendered on the page for dojo based controls. More care is required when overriding of these dojo elements and styles. The Demo Theme described at the end of this document provides a number of overrides that may be required in a custom CSS.

Note: Due to the inheritance model of a number of CSS files used, care will be required in this area.

Additional Field Styles

There are various situations when you may wish to apply various features to Fields. It is recommended that where possible class names are used in order to apply a given capability. It is also very important to remember that multiple class names can be applied to a Field .e.g. "**defaultBackground mandatory**" or "**textbox highlight**".

To assist users, there are a number of predefined classes available:

- **mandatory** – This style is used if the "*" is used to indicate a field is mandatory for validation purposes.
- **highlight** – This style is used to highlight the font of a field. By default this applies a bold color.
- **highlightBackground** - This style is used to highlight a "**defaultBackground**". By default this applies a different background color.
- **disabled** – This style will disable any content within a Group from being editable on the Browser. This style is typically used with the "**Disable If**" option in FormMaker. On modern Browsers it will also display a "**No Entry**" icon when hovering over the fields.
- **checkboxTrue** – This style can be used if you want to display an output image that looks like a checkbox. This can be used with a "**conditional**" display option if a particular data value for a Field is matched. This approach can often be used in repeating tables to display if an option is selected.
  A similar approach can be used for various types of images for displaying icons based on the value of a data field. This may often be used to display a status image icon for a status data value.

Building Block - Repeats

Repeats are used for defining fields or groups of fields that occur multiple times based on the XML data that is transformed by a page. A repeat can contain any number of groups, fields, etc., which will be output in the usual way for each data entry matched by the repeat. In this normal scenario, the repeat itself will be a **DIV** container with a default class name of "**repeat**".

There is also special handling for the situations where the repeat contains a single group that has a "**Display Method**" of "**Table**" or "**Horizontal table**". In this table mode, a **TABLE** structure is rendered with Header and Body Content with fully semantic HTML. The Header (**THEAD** > **TR** > **TH**) renders the Labels for the Table columns, and the Body (**TBODY** > **TR** > **TD**) renders the Field Controls for the Table cells that align with the column headings.

The **TH** element has the "**labelBackground**" class applied for the Column Labels, and a nested **SPAN** element for the Label with class "**label**". Whereas the **TD** elements have the "**defaultBackground**" class applied for the Field controls, and then a nested control specific to the Field type e.g. **INPUT**, **SELECT, CHECKBOX, SPAN (output)** and so on.

The horizontal table mode is similar to the table display turned on its side. The labels are shown in the first column, and each entry in the repeat causes another column of controls to be added.

Note: If TABLE containers are nested within one another, then there may be situations when additional styling needs to be added to override some of the default styling.

Additional Repeat Styles

There are some situations when you may wish to apply various features to repeat columns. To assist users, there are predefined classes available:

- **hide** - Sometimes there will be a requirement to hide a column based on some conditional data. It is important that the "**hide**" class is used and applied to both the "**Label Background Style**" and the "**Field Background Style**" conditional style options, if the entire column needs to be hidden. If the conditional data bindings are true for both then the entire column will be hidden.

- **Wrapping columns** – By default table columns for repeats are set to prevent text from wrapping: **white-space: nowrap;**. This makes the columns stretch based on the data content, which makes them very dynamic in nature. However, if you want to make the text wrap when it encounters a space within a column, then add a CSS override value of: **white-space: normal;** for the Background (class) override. This option can also be applied to standard Groups that are a table, such as when the force option is used.

Composite Controls

In addition to the standard building blocks of Groups, Fields and Repeats, there are "**Composite Controls**" that consist of multiple individual building block controls. Many of these composite controls have dedicated CSS and javascript files, as the composite controls are only used on some web pages. The CSS files contain only the specific additional styles required for the composite controls particular capability. Most of these controls inherit styles used in the base CSS "**demo.css**".

Most of the Composite Controls have a master Group as a container for the nested elements. Each of these has a specific class name to identify the type of Group e.g. **accordionContainer**.

The following composite controls have explicit CSS stylesheets applied when they are added to a Page, and are likely to require overrides in a customer specific CSS theme:

- Paging Table - **pagingTableContainer**
- Editable Table - **editableTableContainer**
- Collapsible Content - **collapsibleSection**

All these stylesheets can be overridden within a single application CSS theme file if required. The "**Demo Theme A**" referenced in this document adopts this approach as only a small subset of the CSS styles need to be overridden for the theme. The level of customisation may impact this decision. It is preferable to keep overrides to the one main theme CSS file if possible, as this will save unnecessary retrieval of multiple CSS files unless they are actually required for rendering a particular page.

Example CSS Theme

To support the information described in this document, there is a supporting CSS file: "**demo_theme_a.css**". The CSS has been setup for illustration purposes; to show how you can "**clone**" important default styles setup and shipped with WebMaker for a particular customised theme.

As an example the theme has been setup as an HTTP remote based CSS and collection of images. You can add a reference to the remote file by adding an Application wide CSS on the Application Map within the Application "**Advanced Options**" on the left hand panel. Add the following CSS filename after or replacing the existing "**demo.css**" filename:
http://www.hyfinity.com/demos/hyf_demo_theme_a/webapp/css/demo_theme_a.css

You can start creating your own theme by creating a new CSS file for your Application directly as another entry. The simplest approach is to start by typing in the CSS File name you want to have included to your Application defaults in the "**Advanced Options**". Once the new name has been "Inserted", then click on it to open it up in the CSS editor. Paste in the content from the file attached to this FAQ. Finally, save the CSS file within the Studio. With this approach you can then override the styles look and feel in this new CSS to get the required look and feel.

In many situations it is desirable to create a Customer specific CSS stylesheet that can then be shared across multiple projects. This approach would significantly reduce the amount of custom styling required across multiple projects within the same customer. It also means that it is much easier to change the general styling and then re-apply to a number of Projects if required.

These corporate wide CSS and images could be hosted remotely from the application in a similar manner to the "**Demo Theme A**" css. The choice will depend on how the organisation wants to manage the look and feel styling.

It is recommended that any images required for the CSS should be placed into a new directory at the same level as the existing theme css/images that can be found in the design repository "webapp" directory for your project. We have also attached a zip containing the "**Demo Theme A**" css and images. This directory can unzipped over the design repository "webapp" directory for your project, if you want all the elements stored locally.

Note: If a specific application / form require some specific styles, then it is recommended that they are added as application or page specific CSS files.

Example Theme Project

In many organisations a web designer may well be tasked with developing and defining this initial corporate theme that can then be referenced or shared across applications. To assist web designers in setting up a corporate themed CSS with images, Hyfinity have provided an example "**DemoThemeA**" project that shows the majority of the basic and composite controls used within most projects. It may be advisable to use this project to build the initial corporate look and feel. This will ensure that the majority of WebMaker rendering variations are taken into account.

Initial previewing of the project pages will show the styling that applies to the WebMaker product with the alternative theme. It is then possible for the web designer to interchange the CSS with a corporate theme to ensure compatibility. The "Demo Theme A" is also hosted on the http://www.hyfinity.com/demos/hyf_demo_theme_a/ website for remote testing purposes.