# Table of Contents

## Introduction

WebMaker creates standards based web forms and applications based on HTML5, CSS3 and optional JavaScript. The WebMaker Studio has been designed and tested to ensure that the palette controls will operate correctly across all commonly used standards based browsers (Internet Explorer 9+, Chrome and Firefox). The majority of modern tablets and phones use browsers that are also standards based.
Older non standard desktop browsers (Internet Explorer 6-8, Firefox 3.x, Safari 3) will be functional but with a degraded look and feel in some cases. WebMaker does provide default CSS that can be used specifically to provide a better look and feel for IE6-8 by using a "**polyfill**" style CSS2.1 approach. There are different approaches used for some of the Themes. For example, the "**Sunrise**" theme has an IE6-8.css that uses an approach that applies some basic linear gradient effects using an open source framework. Some themes will use background-images instead, others such as "**Lavender**" will simply use the same background-color. You will find the CSS within each project/theme.

This document aims to explain the design principles that have been used to create the default WebMaker demo styling. It will provide the recommended approaches for creating application or customer specific "look and feel" themes, which still preserve the WebMaker User Interface Design Principles, and support the product features that link with the CSS.

## Theme Design Template Project

Accompanying this document is 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. It includes a number of pages with different types of layout and controls that should provide a good coverage of WebMaker application look and feel. It can be useful to use a copy of this project to design your "customer" specific theme before using it in other project specific applications.

## Simple Theme Overriding

At this time we generally recommend WebMaker users should setup up a new "customer" specific css that will contain overrides of the main styling blocks used in the default "**demo.css**" file. In most cases users will likely want to override only a few particular elements such as colors, font sizes, background colors, etc. In these simple cases the main classes can simply be overridden.
There are a set of pre-defined classes for styling that are added to types of groups, fields and labels. For example, "**LABEL.label, SPAN.label, .fieldLabel**" and "**.labelBackground**" used for standard field label text and label background respectively. Sometimes the CSS selectors will have multiple parts as they can be attached in different ways to controls to override default behaviour. So, "**fieldLabel**" can be added to a Group Label text to make it look like a field label.

In a "customer" specific CSS the default styling can be extended by using the same CSS class selector as in the demo.css, with the necessary settings e.g. fonts, colors, etc. Please refer to the section at the end of the document to find the correct CSS selectors in the CSS that should be used for each type of Label, Field and Group.

## Specific Theme Extensions

There may also be specific special classes that are required for a particular customer design in addition to the base classes supplied. In these cases new class definitions could be added to the "customer" css and then these class names applied to the relevant groups or fields on your pages. When using this approach, it may be sensible to prefix these new class names in some manner so they are clearly identifiable.
For example, there may be a requirement for two types of group headings to distinguish look and feel for perhaps major and minor sub-headings.
In this case add new classes to the "customer" CSS e.g. "**ABCLevel1GroupLabel**", "**ABCLevel2GroupLabel**". These class names will then need to be manually added to the specific Group labels as required.

## Cloning the full CSS

It is possible to "clone" the default "**demo.css**" file to create your own radically different look and feel, and consequently adjust lots of elements in order to revise all styling within a project. If this approach is adopted you will need to keep in mind the following points:

- WebMaker relies on particular CSS selector values for some functionality, so it is important to maintain the existing CSS selectors.
- Any future version improvements or fixes to the default demo.css will not be carried through into the cloned CSS. The CSS includes a lot of CSS selectors that will need to be reviewed when upgrading to see if any changes need to be added to your cloned CSS.

## User Interface Design Principles

The web applications rendered by WebMaker with HTML5 and CSS3, have been designed based on the following 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. WebMaker provides a variety of pre-defined class names to allow content to be hidden or displayed based on the type of device; desktop, tablet, phone or print.

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 layering 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 HTML5/CSS3 capabilities, but degrade to a CSS2.1 experience if the user's browser does not support the CSS3 capabilities. In some cases there will also be "poylfill" JavaScript processing to mimic the behaviour seen in modern standards based browsers.

Both of these approaches are generally recognised as "best practice" within web design communities. WebMaker 6 has also adopted what is known as modern "flex-box" layout and alignment approach. This allows us to make the experience of creating very advanced layout configurations very simple through the Design Studio interface. It has only been relatively recently that a tipping point has been reached where enough of the standards based browsers support these capabilities. For those that don't the polyfill approach is used.
As time moves on there will be less and less of a need to adopt the "polyfill" techniques for non standards based browsers on mobile or desktops. For example, Microsoft is removing support for IE6/IE7 on Windows XP in March 2014 due to the retirement of Operating System.

## 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.

WebMaker delivers thin and standard web pages. It is generally recommended that any styling should be based on class names rather than HTML element names. The new class names can be added to any of the key components above.
Web page content introspection tools such a **Developer Tools** in Internet Explorer, FireFox and Chrome can be used to check the content of every page created and delivered by WebMaker, if a user is trying to manually create sophisticated custom CSS styling.

WebMaker applications have a default master CSS file, which is named "**demo.css**" and located under "**Application Details**" on the right hand panel of the "**Application Map**" tab when no components are selected (click on the white canvas background). You can change or replace this file, but we strongly recommend adding additional CSS files for your needs. 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 select the page on the Application Map and add one or more CSS files on the right hand panel.

You can also access and edit all CSS files applicable to a particular page from within the CSS tab on the right side of each "Page Design" studio tab. Any changes you make to the CSS files using the editor within the studio will be seen immediately on any "**Page Design**" or "**Preview**" studio tabs that you have open. If you are using an external editor then you will need to use the refresh buttons on these two tabs to pick up any changes you have made.

## Building Block - Groups

Groups are used to control the layout and positioning of nested content (Groups, Fields and Repeats). Sometimes Groups apply background styling and Label headings for a collection of content.

### Types of Group

WebMaker provides a number of "**Layout Groups**" that have a primary role of controlling only layout/organisation e.g**. Layout Group**, **Layout Grid** and **Containers**. These are important building blocks to control layout aspects such as alignment, wrapping and sizing. These groups do not provide any look and feel aspects by default. They are also only shown in "**Design Mode**" for the "**Page View**" as they will not be seen at runtime or on a "**Page Preview**".
They are often used to control whether fields are conditionally displayed or disabled, or when there is a need to control the organisation of nested content. Any Group can be nested in another Group, so there is a lot of flexibility to get the exact layout required.
*Note: The most important consideration is that these structural / layout type Groups do not take up any space on the rendered page by default. There is no padding or margins associated with these Groups.*
There are also various "**Visual Groups**" that are similar in capabilities, but do have default styling defined e.g. **Bordered Group**, **HTML Fieldset (Info. Group)**.

### Layout Group Capabilities

The table below describes the various Groups and how they differ in terms of capabilities and HTML structure rendered. The "**Properties**" tab of the "**Page Design**" will vary in terms of the options available. All Groups do have a "**Display Method**" indicator that controls the HTML that is rendered, which can be found in the **More Options > Historic Options** within the "**Properties**" tab. It is not expected that the value should need to be changed.

| Display Method | HTML created | Alignment | Distribution | Fit/Expand Sizing | Exact Sizing |
|---|---|---|---|---|---|
| Layout Group | div > div | Yes | Yes | Yes | Yes |
| Container | div | | | Yes | Yes |
| Layout Grid | table | | | Yes | Yes |
| | | | | | |
| Ordered List | ol > li | | | | Yes |
| Unordered List | ul > li | | | | Yes |
| Repeat Vertical Table | table | | | | Yes |
| Repeat Horizontal Table | table | | | | Yes |
| | | | | | |
| Horizontal (Historic Option) | span | | | | Yes |
| Vertical (Historic Option) | div | | | | Yes |

A "**Layout Group**" will create an outer **DIV** (class of **layoutContainer**) and a nested **DIV** (**layoutContainerContent**) for any content whether it is a Group or a Field. These allow the maximum flexibility in terms of alignment and positioning options.

A "**Container**" does not have the same level of alignment and positional capabilities. They are simply used to contain other content e.g. a Partial Page Container. These are most commonly used with "**Rich Composite Controls**".
A "**Layout Grid**" creates a **TABLE** (**grid**) that enables the same nested alignment options, but constrained by the implicit structure of an HTML table.

The last two options in the table above (horizontal and vertical) are the type of groups used prior to WebMaker version 6. These groups should only be used moving forward if you need to support IE6/7. They are primarily retained for backward compatibility of projects. These groups have a number of checkboxes that are related:

- "**force table layout**" - This will also impact the HTML rendered. In these cases the HTML is rendered as an HTML **TABLE** with a class of "**force**" on the TABLE element.
- **"Use a fieldset for this group"** - This will switch the group to appear like a standard HTML Fieldset with a Legend element for the Label.
- "**Extended border styling**" - This option generated additional HTML around the group to allow different background images to be used to create legacy shaded box style.

*Note: None of these options have any impact on new style groups and containers, so a user should not need to view this section for new projects.*

## Visual Group Details

Visual Groups are generally defined as "**Layout Group**", "**Layout Grid**" or "**Container**", but with additional common styling. 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 when initially created. Groups with labels create a container for the label background dependent on the parent. For example, a "**Bordered Group**" at the top of a page would render a **DIV** as the Label is set on by default, and is defaulted with an associated class of "**groupLabelBackground**". This then contains a nested **H2** element with a class of "**label**" for the actual Group name text. The group labels can be positioned "**inside**" or "**outside**" the group, although they generally default to "**outside**" the group.
These groups do usually have some padding to ensure that there is a small gap at the top and bottom of the Group container. The padding is exactly the same depth as applied to individual Fields for consistency.
If "**HTML Fieldset (Info.Group)**" is selected, then a **FIELDSET** HTML control container is rendered for the Group with a "**fieldset**" class. Nested within the Group is a **LEGEND** element for the "**fieldsetLabelBackground**", with a nested **SPAN** for the actual label text with class of "**label**". The **SPAN** contains the actual Fieldset Group label text.

There are some Groups that have a specialised use, and are usually parts of larger "**Rich Composite Controls**" e.g. Accordion Pane within an Accordion Container. Details of these Group containers can be found in the section on "**Rich Composite Controls**" towards the end of this document. The primary difference is that composite controls will likely contain a number of nested Groups, Fields and Containers.

## 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.* "**borderedGroup shadowBox roundedBox**". *Each must be separated with a blank, and will be applied one after the other. Some WebMaker features may automatically add multiple class names to a control.*

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

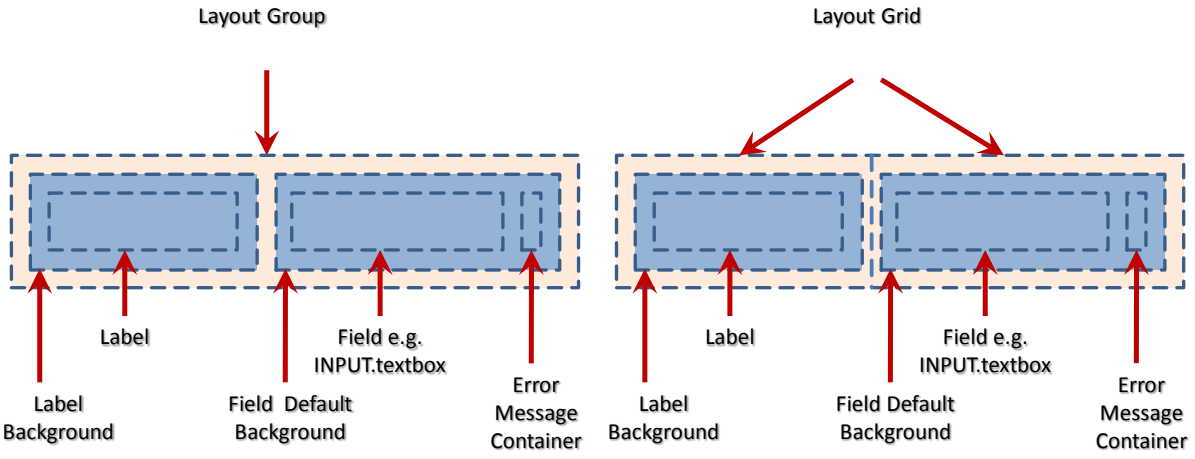
- **roundedBox** – This applies rounded corners to all four corners of a box. This style applies CSS3 based styling that will work with modern browsers supporting HTML5 and CSS3, but will not be applied for non standard Browsers: IE 6, 7 and 8. For these browsers a "poly-fill" approach can be adopted by selecting the "**demo_ie_only.css**" and defining a Browser based "**File Condition**" for it of Internet Explorer less than or equal to 8.
- **shadowBox** – This applies a shadow to the Group with shadow to the right and bottom. The same IE6-8 considerations apply.
- **hide** – This style will hide a Group from being displayed on the Browser. However, it is probably better to use conditional display capabilities.

## 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 Design**" 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 "**Layout Group**" or "**Layout Grid**" on a page. Depending on some of the WebMaker options used, some of the HTML elements may not be rendered e.g. The error/hint message container.




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

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

If a field is placed into a standard "**Layout Group**" on a page (**DIV**) or Bordered Group (**DIV**), then a "**controlContainer**" (DIV with **display:table**) and a "**controlRow**" (DIV with **display:table-row**) container are rendered that encompasses the Field Label and Field Control elements. This container exists to allow the label and field control elements to be aligned based on the specific alignment and sizing options selected, and also the "**Label position for contained components**" definition on the Group container or its field level

override. If the labels are above or below then two "**controlRow**" elements are created, so that they behave as two rows of a table.
The "**labelBackground**" and "**defaultBackground**" elements (**SPAN**s with a **display:table-cell**) are then nested inside the row. This approaches means that a consistent model is used for the HTML that is created regardless of the position of the labels, and the alignment of a number of fields behaves in the same manner whether in a "**Layout Group**" or a "**Layout Grid**".
*Note: The CSS models are consistent, but the grid will not use the "**controlContainer**" and "**controlRow**" classes.*

The approach above also means that each individual field can have their label positioned altered independently of the groups default settings if required.
*Note: The "**controlContainer**" and "**controlRow**" DIVs do not provide any look and feel styling, they are present for positional alignment. This ensures that the labels and fields are spaced in a consistent manner when a number are used together.*

## Labels

Labels by default have been set with a fixed width value for convenience, so they will wrap the Label text if it is too long and ensure that the labels and fields are aligned correctly in a group. The width can be overridden by dragging the edge of the control, or applying a CSS Override to the "**labelBackground**". When a "Layout Grid" is used the Labels and Field Controls are placed into separate columns of a Table structure, so that the width of a number of Labels can be changed as a collection. 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 "Layout Grid", and it will be cascaded to all the others in the column.

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. However, the user can override the label position for individual fields.

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 View / Layout View**" 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 that is provided in the default CSS, then simply override the default CSS selector that sets the label width:
**SPAN.labelBackground, DIV.labelBackground, DIV.fieldLabelBackground,**

**TABLE.force TD.labelBackground, TABLE.table TD.labelBackground, TABLE.grid**
Use this CSS selector in your "customer" specific CSS.

## 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 Column Heading) onto a standard Field label, which is normally defaulted to "**labelBackground**".

*Note: If this approach is to be adopted on a control, then the existing style names should be replaced.*
*Note: You can also add your own additional label class names if needed.*

## 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 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 **TD** depending on the definition of the Group that the fields are contained within.

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. Depending on whether or not the number of columns or number of rows option is specified a different HTML structure will be used. If these options have not been set, then the controls will be side by side, wrapping onto a new line if needed. There will be a SPAN around each INPUT and one around the LABEL display text for each option. Each of these pairs of SPANS will be within another SPAN with class "**selectBooleanEntry**". If the number of

rows or columns has been set, then instead a **TABLE** structure will be used to achieve the required layout. The **INPUT** elements and their LABELs will be contained within separate TD elements. 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**". The display values can be styled.

- **Multi-Select Box** - The Multi-Select Box has an additional Control class of "**multiSelect**". This is specifically important in terms of controlling the height of the control.
- **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 Properties section.
- **Button** – A standard HTML button is defined as an **INPUT** element with a specific type of "**button**".
- **Enhanced Button** – This control is rendered as an HTML **A** (anchor) element with 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. "**saveIconButton**" for a button that shows a file saving image on the actual button within a nested **SPAN**.
- **Hyperlink** – This control is rendered as an HTML **A** (anchor) element, which includes two nested **SPAN** elements. These nested **SPAN** elements provide greater flexibility to add visual effects.
- **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.

## Other Considerations

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. These are generally located at the bottom of the CSS.
*Note: Due to the inheritance model of a number of CSS files used, care will be required in this area.*

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**". Many of these styles are most suitable if you want conditional styling or events processing to change the look and feel of a field.

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** – These style are used to highlight the font of a field, usually with conditional styling or events. There are a number of variants that change the color: **highlightTextOverdue** (Red), **highlightTextUrgent** (Orange), **highlightTextCompleted** (Green) and **highlightTextImportant** (Blue).
- **highlightFieldBackground** - 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**" capability. 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.
- **requiredField** - This can be manually add to a fields background to set it to a light background color.
- **highlightFocusedField** - This can be placed on a Group, so that if a field is focused on then it will be highlighted with a background color.

## 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 "**Repeat Vertical Table**" or "**Repeat 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.*

## 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, so that these only need to be processed by the pages that use these controls. 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 "**Container**" Group for the nested elements. Each of these has a specific class name to identify the type of Group e.g. **accordionContainer, partialPageContainer**.
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 "customer" specific CSS file if required. The level of customisation may impact this decision. It is preferable to keep overrides to the one main "customer" 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 number of theme examples that are shipped with the product. Each has at least a "**demo.css**". Some will have additional CSS for print, IE6-8 and mobile devices. They have been setup for illustration purposes; to show how you can override important styles shipped with WebMaker for a particular customised theme.
The themes are available within a project in the "theme" directory. You can include these CSS files on the Application Map screen. It is recommended that you remove the existing files(s) if you switch to a different theme.

You can start creating your own theme by creating a new CSS file for your project in the studio, and including it on your application. Once the new file has been "included", then click on it to open it up in the CSS editor. Paste in the css selector content from the existing demo.css file for the specific styles you which to override, and then adjust the values as required. You only need to have the values that are different. Finally, save the CSS file within the Studio. With this approach you can then override the styles 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 using a **http://...** remote location. The choice will depend on how the organisation wants to manage the look and feel styling.
When creating a new theme it is recommended that you create a new directory for it under the existing "**webapp/theme**" directory in the repository for your project.
Within this you can create the CSS and images directories that will contain the required files. (Alternatively you can copy one of the existing theme directories, and replace or update the files as required.)
*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 "**DemoThemeTemplate**" 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 default theme. It is then possible for the web designer to interchange the

CSS with a corporate theme to see how these various layout scenarios are handled, and so ensure compatibility of the new theme.

## Guidelines to Overriding the CSS Theme

The following table defines the primary CSS selectors that provide styling for various palette controls. This table will help you to find the CSS selectors in the **demo.css**, so that you can then use the same selectors to override the look and feel in your own "customer" CSS.

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| **Page Level Default Styling** | | |
| Page Background | | BODY |
| Page Content | | .main_body .form .pageContainer,<br>.main_body.pageContainer |
| Page Text Defaults<br>e.g. Font-Family | | BODY, TEXTAREA, SELECT, P, LEGEND, LABEL, INPUT, H1, H2, H3, H4, H5, H6, FIELDSET, CAPTION, BUTTON, A,<br>.main_body .form .dijitTextBox, .main_body .form .dijitComboBox, .main_body .form .dijitPlaceHolder,<br>.main_body .form .repeat, .richTextEditor |
| | | |
| Field Label Text | | LABEL.label, SPAN.label, .fieldLabel |
| Field Label Background | | .labelBackground, .fieldLabelBackground |
| Field TextBox Control Text | Normal | INPUT.textbox |
| Field TextBox Control Text | Hover | INPUT.textbox:hover,<br>SELECT:focus,<br>SELECT:hover,<br>TEXTAREA:hover,<br>.main_body .form .dijitTextBox:hover,<br>.main_body .form .dijitComboBox:hover,<br>.main_body .form .dijitSpinner:hover,<br>.main_body .form .dijitTextArea:hover,<br>.main_body .form .dijitEditor:hover |
| Field Other Control Text | Normal | SELECT, TEXTAREA,<br>.main_body .form .dijitTextBox,<br>.main_body .form .dijitComboBox,<br>.main_body .form .dijitSpinner,<br>.main_body .form INPUT.dijitInputInner,<br>.main_body .form .dijitTextArea,<br>.main_body .form .dijitEditor |
| Field Other Control Text | Hover | INPUT.textbox:hover,<br>SELECT:focus,<br>SELECT:hover,<br>TEXTAREA:hover,<br>.main_body .form .dijitTextBox:hover,<br>.main_body .form .dijitComboBox:hover, |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| | | .main_body .form .dijitSpinner:hover,<br>.main_body .form .dijitTextArea:hover,<br>.main_body .form .dijitEditor:hover |
| Field TextBox Control Text | Disabled | INPUT[type='text'].disabled |
| Field TextBox Control Background | Normal | .defaultBackground |
| Field TextBox Control Background<br><br>(Shared styling) | Hover | INPUT.textbox:hover,<br>SELECT:focus,<br>SELECT:hover,<br>TEXTAREA:hover,<br>.main_body .form .dijitTextBox:hover,<br>.main_body .form .dijitComboBox:hover,<br>.main_body .form .dijitSpinner:hover,<br>.main_body .form .dijitTextArea:hover,<br>.main_body .form .dijitEditor:hover |
| Field TextBox Control Background | Disabed | .main_body .form .defaultBackground .disabled |
| | | |
| Group Heading Label Text | | .groupLabelBackground .label, H2.label,<br>.pagingTableLabelBackground .label, .groupLabel |
| Group Heading Label Background | | .groupLabelBackground |
| | | |
| **Specific Field Control Styling** | | |
| Button - Primary Action | Normal | INPUT.button |
| Button - Primary Action | Hover and Active | INPUT.button:hover,<br>INPUT.button.secondaryActionButton:hover,<br>INPUT.button:active,<br>INPUT.button.secondaryActionButton:active |
| Button - Primary Action | Disabled | .main_body .form .defaultBackground<br>INPUT.disabledButton,<br>.main_body .form .defaultBackground<br>A.disabledButton |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| | | |
| Button - Secondary Action | Normal | INPUT.button.secondaryActionButton |
| Button - Secondary Action | Disabled | .main_body .form .defaultBackground INPUT.disabledButton, .main_body .form .defaultBackground A.disabledButton |
| | | |
| Extended Button (Anchor) - Primary Action | Normal | a.button, a.button:link, a.button:visited |
| Extended Button (Anchor) - Primary Action | Hover | a.button:hover |
| Extended Button (Anchor) - Primary Action | Disabled | .main_body .form .defaultBackground INPUT.disabledButton, .main_body .form .defaultBackground A.disabledButton |
| | | |
| Extended Button (Anchor) - Secondary Action | Normal | a.secondaryActionButton |
| Extended Button (Anchor) - Secondary Action | Hover | a.button:hover |
| Extended Button (Anchor) - Secondary Action | Disabled | .main_body .form .defaultBackground INPUT.disabledButton, .main_body .form .defaultBackground A.disabledButton |
| | | |
| Output Field | Normal | .output |
| | | |
| Select Drop-Down List (Only height details) | Normal | SELECT, .select |
| | Disabled | .main_body .form .defaultBackground SELECT[disabled="disabled"] > OPTION |
| | | |
| Multiple Select List | Normal | SELECT.multiSelect, SELECT[multiple], SELECT[size] |
| | Disabled | .main_body .form .defaultBackground SELECT[disabled="disabled"][multiple="multiple"] OPTION[selected="selected"] |
| | | |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| Text Area | | TEXTAREA, .textarea |
| | | |
| Paragraph | Normal | .main_body .form P, .main_body .form DIV.customControl |
| | | |
| Custom Control (HTML) | | .main_body .form P, .main_body .form DIV.customControl |
| | | |
| Radio | Normal | INPUT.radio, INPUT[type="radio"] |
| Radio | Disabled | .main_body .form .defaultBackground INPUT[type="radio"].disabled, .main_body .form .defaultBackground INPUT[type="checkbox"].disabled |
| | | |
| Checkbox | Normal | INPUT.checkbox, INPUT[type="checkbox"] |
| Checkbox | Disabled | .main_body .form .defaultBackground INPUT[type="radio"].disabled, .main_body .form .defaultBackground INPUT[type="checkbox"].disabled |
| | | |
| Radio and Multi-checkbox Labels for options | | LABEL.radio, LABEL.checkbox |
| | | |
| Hyperlink | Normal | A.hyperlink |
| Hyperlink | Hover | A.hyperlink:hover |
| Hyperlink | Disabled | .main_body .form .defaultBackground A.disabled |
| Hyperlink | Visited | A.hyperlink:visited |
| | | |
| Image | | IMG.image |
| | | |
| Rich Text Editor | Normal | TEXTAREA, .richTextEditor |
| Rich Text Editor | Disabled | BODY.mceContentBody.disabledRichTextEditor<br><br>.mceEditor .defaultSkin .mceIframeContainer |
| | | |
| **Specific Group Styling** | | |
| Bordered Group Container | Normal | .borderedGroup |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| | | |
| Fieldset Group Container | Normal | FIELDSET.fieldsetGroup |
| Fieldset Group Heading Label Text | Normal | FIELDSET.fieldsetGroup LEGEND .label, .fieldsetGroupLabel |
| Fieldset Group Heading Label Background | Normal | FIELDSET.fieldsetGroup .fieldsetGroupLabelBackground, FIELDSET.fieldsetGroup .groupLabelBackground |
| | | |
| Accordion Container | | .main_body .form .accordionContainer |
| Accordion Pane | Normal | .main_body .form .accordionContainer .accordionPane |
| Accordion Heading Text | Normal | .main_body .form .accordionContainer .dijitAccordionText |
| Accordion Heading Background | Normal | .main_body .form .accordionContainer .dijitAccordionTitle |
| Accordion Heading Background | Selected | .main_body .form .accordionContainer .dijitAccordionInnerContainer .dijitAccordionTitleSelected |
| Accordion Heading Toggle icon | Normal | .main_body .form .accordionContainer .dijitAccordionTitle .dijitAccordionArrow |
| Accordion Heading Toggle icon | Selected | .main_body .form .accordionContainer .dijitAccordionTitleSelected .dijitAccordionArrow |
| | | |
| Tab Container | | .tabContainer |
| Tab Pane | Normal | .tabContainer .tabPane |
| Tab Heading Background - Main | Selected | .tabContainer SPAN.defaultBackground A.selectedTab, .tabContainer SPAN.defaultBackground DIV.group A.selectedTab |
| Tab Heading Background - Main | Normal | .tabContainer SPAN.defaultBackground A.unselectedTab, .tabContainer SPAN.defaultBackground DIV.group A.unselectedTab |
| | | |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| Popup Dialog - Pane | Normal | BODY .main_body.dijitDialog .dijitDialogPaneContent |
| Popup Dialog - Heading Text | Normal | BODY .main_body.dijitDialog .dijitDialogTitleBar .dijitDialogTitle |
| Popup Dialog - Heading Background | Normal | BODY .main_body.dijitDialog .dijitDialogTitleBar |
| Popup Dialog - Close icon | Normal | BODY .main_body.dijitDialog .dijitDialogTitleBar .dijitDialogCloseIcon |
| | | |
| **Collapsible Section Styling** | | |
| Collapsible Section Group - Container | Normal | .collapsibleSection |
| Collapsible Section Group - Collapsed Content | Normal | .collapsibleSection .collapsibleContent, .collapsibleSection .collapsedContent |
| Collapsible Section Group - Collapsable Content | Normal | .collapsibleSection .collapsibleContent, .collapsibleSection .collapsedContent |
| Collapsible Section Group Heading Text | Normal | .collapsibleSection A.toggleVisible, .collapsibleSection A.toggleHidden |
| Collapsible Section Group Heading Background | Normal | .collapsibleSection .toggleLabelBackground |
| Collapsible Section Group Heading Toggle icon | Visible | A.toggleVisible |
| Collapsible Section Group Heading Toggle icon | Hidden | A.toggleHidden |
| Collapsible Section Group Heading Toggle icon | Visible + Hover | A.toggleVisible:hover |
| Collapsible Section Group Heading Toggle icon | Hidden + Hover | A.toggleHidden:hover |
| | | |
| | | |
| **Repeating Table Styling** | | |
| Standard Repeating Group Container | | TABLE.repeat, DIV.repeat<br><br>TABLE.repeat TD.labelBackground, TABLE.repeat TD.defaultBackground, TABLE.repeat DIV.defaultBackground |
| Standard Repeating Group Cell | | TABLE.repeat TD.labelBackground, TABLE.repeat TD.defaultBackground, TABLE.repeat DIV.defaultBackground |
| Standard Repeat Column Heading Text | | TABLE.repeat TH .label, .repeatLabel |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| Standard Repeat Column Heading Background | Normal | TABLE.repeat TH.labelBackground, .repeatLabelBackground |
| Standard Repeat No Data | | TABLE.repeat TD.noRepeatData |
| Standard Repeat Output field (Overrides when nested in repeat) | | TABLE.repeat TD SPAN.output |
| Standard Repeat Button (Overrides when nested in repeat) | | TABLE.repeat TD INPUT.button |
| Standard Repeat Custom Control (Overrides when nested in repeat) | | TABLE.repeat TD DIV.customControl |
| | | |
| | | |
| Paging Table Heading Text | | .pagingTableLabelBackground .label |
| Paging Table Heading Background (Same as group Label) | | . .groupLabelBackground .label, H2.label, .pagingTableLabelBackground .label, .groupLabel |
| **Paging Table Styling (In pagingtable.css)** | | |
| Paging Table Heading Row Background | Normal | table.tablesorter thead tr |
| Paging Table Heading Background - Not Sorted | Normal | table.tablesorter thead tr .header span.label |
| Paging Table Heading Background - Ascending | | table.tablesorter thead tr .headerSortUp span.label |
| Paging Table Heading Background - Descending | Normal | table.tablesorter thead tr .headerSortDown span.label |
| Paging Table Controls - First | Normal | .tablesorterPager .first |
| Paging Table Controls - First | Hover | |
| Paging Table Controls - | Normal | .tablesorterPager .prev |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| Previous | | |
| Paging Table Controls - Previous | Hover | |
| Paging Table Controls - Next | Normal | .tablesorterPager .next |
| Paging Table Controls - Next | Hover | |
| Paging Table Controls - Last | Normal | .tablesorterPager .last |
| Paging Table Controls - Last | Hover | |
| Paging Table Controls - No. of Rows | Normal | .tablesorterPager input |
| Paging Table Footer Background | Normal | |
| **Editable Table Styling (In editabletable.css)** | | |
| | | |
| Editable Table Row Container | | .editableTableContainer |
| Editable Table Row - Edited Row | | .editableTableContainer .editabletable_edit_row |
| Editable Table Icons | | .editableTableContainer TABLE.repeat TD A.editabletable_edit_btn, .editableTableContainer TABLE.repeat TD A.editabletable_remove_btn, .editableTableContainer TABLE.repeat TD A.editabletable_accept_btn, .editableTableContainer TABLE.repeat TD A.editabletable_cancel_btn, .editableTableContainer TABLE.repeat TD A.editabletable_reorder_up_btn, .editableTableContainer TABLE.repeat TD A.editabletable_reorder_down_btn |
| Editable Table Row - Display Mode - Edit icon | Normal | .editabletable_edit_btn |
| Editable Table Row - Display Mode - Edit icon | Hover | .editabletable_edit_btn:hover |
| Editable Table Row - Display Mode - Remove icon | Normal | .editabletable_remove_btn |
| Editable Table Row - Display Mode - Remove icon | Hover | .editabletable_remove_btn:hover |
| Editable Table Row - Edit Mode - Accept icon | Normal | .editabletable_accept_btn |
| Editable Table Row - Edit Mode - Accept icon | Hover | .editabletable_accept_btn:hover |
| Editable Table Row - Edit Mode - Cancel icon | Normal | .editabletable_cancel_btn |
| Editable Table Row - Edit Mode - Cancel icon | Hover | .editabletable_cancel_btn:hover |
| Editable Table Row - Reorder Up icon | Normal | .editabletable_reorder_up_btn |
| Editable Table Row - Reorder Up icon | Hover | .editabletable_reorder_up_btn:hover |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| Editable Table Row - Reorder Down icon | Normal | .editabletable_reorder_down_btn |
| Editable Table Row - Reorder Down icon | Hover | .editabletable_reorder_down_btn:hover |
| | | |
| | | |
| | | |
| **Other Feature Styling Options** | | |
| Highlight Text - Overdue | | .main_body .form .highlightTextOverdue, .main_body .form .highlightTextOverdue INPUT, .main_body .form .highlightTextOverdue LABEL |
| Highlight Text - Urgent | | .main_body .form .highlightTextUrgent, .main_body .form .highlightTextUrgent INPUT, .main_body .form .highlightTextUrgent LABEL |
| Highlight Text - Completed | | .main_body .form .highlightTextCompleted, .main_body .form .highlightTextCompleted INPUT, .main_body .form .highlightTextCompleted LABEL |
| Highlight Text - Important | | .main_body .form .highlightTextImportant, .main_body .form .highlightTextImportant INPUT, .main_body .form .highlightTextImportant LABEL |
| Highlight Mandatory Marker | | .mandatory |
| Highlight Repeat Row Background (usually on hover) | | .main_body .form .highlightRowBackground TD |
| Highlight Required Field | | INPUT.requiredField, SELECT.requiredField, TEXTAREA.requiredField, .main_body .form .dijitTextBox.requiredField |
| Highlight Field Background | | .main_body .form .highlightBackground .dijitTextBox, .main_body .form .highlightBackground INPUT, .main_body .form .highlightBackground SELECT, .main_body .form .highlightBackground TEXTAREA |
| Highlight Field Background Border | | TD.highlightFieldBackgroundBorder, SPAN.highlightFieldBackgroundBorder |
| Highlight Focused Field | | .highlightFocusedField INPUT:focus, .highlightFocusedField SELECT:focus, .highlightFocusedField TEXTAREA:focus, .main_body .form .highlightFocusedField .dijitFocused |
| Highlight Repeating Table Row - First Row | | .repeat .firstRow |
| Highlight Repeating Table Row - Alternate Row | | .repeat .alternateRow |
| Highlight Repeating Table Row - First Column | | .repeat .firstColumn |
| Highlight Repeating Table Row - Alternate Column | | .repeat .alternateColumn |
| | | |
| Nest Group Content | | .main_body .form .groupContentPadding, .main_body.groupContentPadding |
| Outline Border for Group | | TABLE.outline, DIV.outline |

| Styling Element | State | CSS Selector Rule |
|---|---|---|
| Outline Border for Group | | TABLE.outline TD.labelBackground, TABLE.outline TD.defaultBackground,<br>TABLE.outline TD P, TABLE.outline TD .customControl,<br>DIV.outline .formElement, DIV.outline .customControl,<br>DIV.outline P,<br>DIV.outline .controlContainer |
| | | |
| Number Column | | .main_body .form .repeat .numberColumn |
| | | |
| Date Picker icon | Normal | A.datePickerIcon |
| | Disabled | .main_body .form .defaultBackground INPUT[type="text"].disabled,<br>.main_body .form .defaultBackground INPUT[type="text"].disabled + SCRIPT + A.datePickerIcon |
| | | |
| Hint icon | | .hintIcon |
| Hint Tip Text Box | | #messageDiv |
| | | |
| Error icon | | .errorIcon |
| Error Message Text | | .errorMessageText, .errorMessageText SPAN |
| Error Message Background | | .errorBackground INPUT, .errorBackground SELECT, .errorBackground TEXTAREA, .errorBackground .dijitComboBox, .errorBackground .dijitTextBox |
| Top Level Error Message | | .topLevelMessage |
| | | |
| Message Box | | .messageBox |
| | | |
| Rounded Box | | .main_body .form .roundedBox,<br>.main_body.roundedBox |
| Shadow Box | | .main_body .form .shadowBox,<br>.main_body.shadowBox |
| Transparency<br>(50% opacity) | | .main_body .form .transparent |