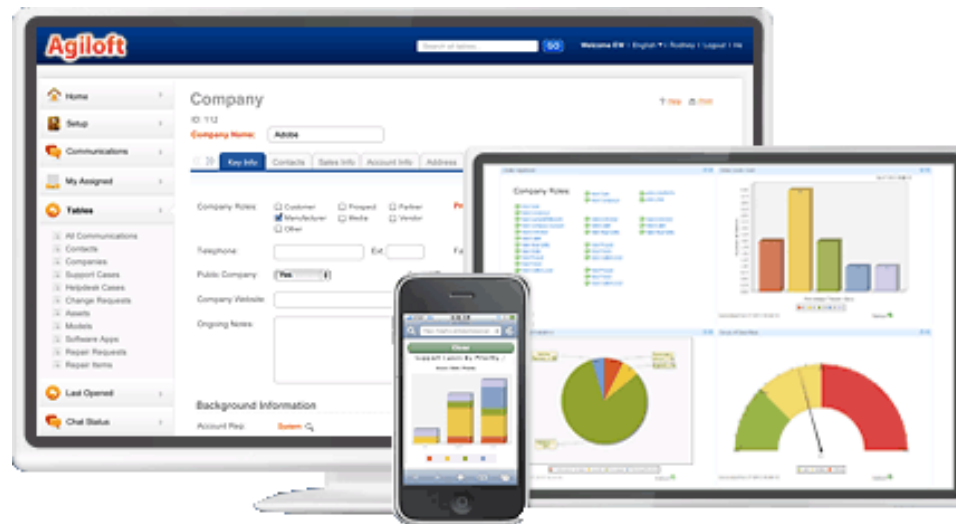


# HTML Concepts and Terms

---



# Unit Overview

---

- In case you haven't had much experience with HTML, this unit briefly addresses a few basic terms and concepts.
- We will cover tags, HTML code, and CSS concepts.
- We assume you have no prior HTML or web development knowledge. If you know your way around HTML editing already, you may want to skim or skip this short unit introduction and move on to unit 19 to review Agiloft's EUI setup, learn about the EUI Templates table, and familiarize yourself with the Agiloft-specific Velocity macros.
- If you don't have prior HTML knowledge, we recommend navigating to the EUI templates table in your KB and following along.

# HTML Building Blocks

---


- *HTML*, or **hypertext markup language**, is the language used to write instructions for web browsers to display pages.
- *Tags* are the basic building blocks of HTML. Tags describe the function of each piece of content in the document, such as a heading, paragraph or image.
- Tags begin and end with angle brackets (< and >). For example, <p> tags enclose a paragraph. Most tags come in pairs, with an opening (start) and closing (end) tag such as <p> and </p>. A closing tag contains a / (slash) before the tag name.
- An individual piece of *content* is enclosed between tags. Together, the content and tags make up an HTML *element*.
  - For example:  
<p>The start tag, content, and end tag make up this paragraph element.</p>
- *Attributes* specify additional information about elements. They come in the form of name/value pairs, like <p **class**="warning"> or <a **href**="www.agiloft.com">.

# HTML Tags – Lists and Numbering

- The following HTML tags are used frequently in the EUI. Over the next few slides are brief definitions of these tags and examples of their use in the Agiloft EUI.
- `<ul></ul>` = *unordered list*. The default display for unordered lists is a bullet point. `<ul>` tags enclose any number of list item elements.
- An `<ol>` is an *ordered list* tag, often used for numbered lists. Unordered and ordered lists both contain list items.
- `<li></li>` = *list item*. Use this tag around individual items of an un/ordered list.

Putting it together:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```



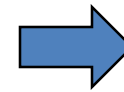
```
• Item 1
• Item 2
```

- Example Usage: In [menu.htm](#), each tab is a list element, enclosed by `<li></li>` tags. The order of tabs in the EUI menu is determined by the list items order.

# HTML Tags – Divs and Tables

- `<div></div>` tags create divisions or sections within an HTML page. A `<div>` is an example of an HTML *block element*. Like other tags, they can be assigned class names and styled with CSS. We will learn more about CSS and classes shortly.
- `<table></table>` tags create a table on your page. Like *divs*, tables can be used to structure the content of webpages.
- `<tr></tr>`, *table row*, encloses each row of content.
- `<td></td>`, *table data* or *table cell*, encloses each cell of the table's content.
- Putting it all together, we nest `<td>` cells inside of `<tr>` row tags, which are finally enclosed in `<table>` tags:

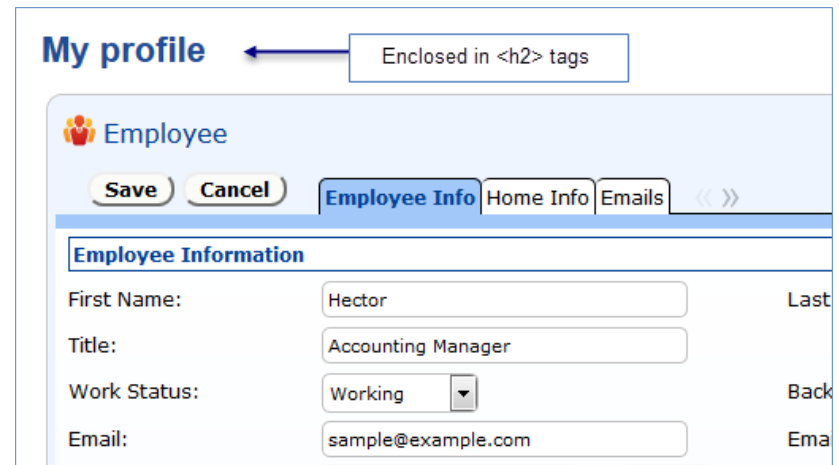
```
<table>
  <tr><td>R1, C1</td><td>R1, C2</td></tr>
  <tr><td>R2, C1</td><td>R2, C2</td></tr>
</table>
```



R1, C1	R1, C2
R2, C2	R2, C2

# HTML Tags – Headings

- *Heading tags* help browsers determine the structure of your page by indexing the content based on the heading level (<h1> through <h6>).
- <h1>Heading 1</h1>  
Example Usage: In [home.htm](#), the text “Welcome to the Agiloft Custom Interface” is enclosed in <h1> tags.
- <h2>Heading 2</h2>  
Example usage: On individual pages such as [my\\_profile.htm](#), the heading is enclosed in <h2> tags.



The screenshot shows a web application interface for a user profile. The main heading is "My profile", which is highlighted by a blue arrow and a callout box labeled "Enclosed in <h2> tags". Below the heading is a user profile card for "Employee" with a person icon. The card includes "Save" and "Cancel" buttons, and a tabbed interface with "Employee Info", "Home Info", and "Emails" tabs. The "Employee Info" tab is active, showing a form with the following fields:

Employee Information		
First Name:	<input type="text" value="Hector"/>	Last
Title:	<input type="text" value="Accounting Manager"/>	
Work Status:	<input type="text" value="Working"/> ▼	Back
Email:	<input type="text" value="sample@example.com"/>	Ema

# HTML Tags – Links and Images

---

- `<a></a>` is a hyperlink tag. It requires the **href** attribute, which specifies the link address.  
Example usage: `<a href="https://www.agiloft.com">Agiloft</a>`
  - The content between the `<a></a>` tags is the link text displayed in the browser.
  - ⚠ Browsers block insecure content from within secure webpages. If your Agiloft connection is over https, any references using just http won't display (including images).
- The `<img />` tag describes and links to an image. There is no closing tag for images. It requires the **src** (source) attribute to define the image's URL:  
Example usage:  
``
  - The **alt** attribute specifies *alternative text* for the image. *Alt text* is displayed if the image is unavailable to the user for any reason. It is also used by screen reader software and text browsers such as Lynx.

# HTML Tags – Under the Hood

---

- `<script></script>` tags describe *client-side scripts* such as JavaScript. Either the script itself or a reference to an external file is contained in the `<script>` tags.
  - Example usage: `menu.htm` contains a simple script that identifies the active tab (the current page) of the EUI, which is used to highlight the menu tab. We will be working with the following script toward the end of this unit.


```
<script language="JavaScript">
function menuChange() {
...
</script>
```

- The `<title></title>` tag defines the title of the page to be displayed in a browser's toolbar. It is enclosed in the `<head>` section and is required for all HTML documents.
- We will come across additional tags such as `<!DOCTYPE>`, `<HTML>`, `<meta>` and `<body>` as we progress through this unit. For more information on these or any other HTML tags, please consult <https://www.w3schools.com/> or other online resource.



# Cascading Style Sheets (CSS)

---

- A *style sheet* is a special file that instructs web browsers on how to display the various elements of your site.
  - Style sheets end in the file extension .css and are known as *CSS (cascading style sheet)* files.
  - Styles dictate the positioning, fonts, colors, borders, and other formatting instructions for displaying content.
  - By storing all your formatting instructions in a CSS file, it is possible to change the appearance of every page on your site by editing a single file.
-  W3Schools.com hosts an excellent demonstration of how the same page content can look dramatically different by applying different stylesheets.  
[https://www.w3schools.com/css/demo\\_default.htm](https://www.w3schools.com/css/demo_default.htm)

# Style Sheets (continued)

- A CSS file is a collection of *style rules*. Each rule is made up of a *selector*, stating which elements the rule applies to, and a *declaration*, which specifies the formatting instructions. The syntax is:

```
selector {declaration1; declaration2; declaration3;}
```

[https://www.w3schools.com/css/css\\_syntax.asp](https://www.w3schools.com/css/css_syntax.asp)

- *Selector types* include tag names, classes, ids and pseudo-classes, and can be combined with certain operators for fine-grained control over styles.

[http://www.w3schools.com/css/css\\_selectors.asp](http://www.w3schools.com/css/css_selectors.asp)

- The declaration can hold any number of property-value pairs, each separated by a semi-colon.

- Let's look at a simple style rule to set the font color in all <p> elements to blue.

- In this example, the *selector* is the paragraph tag name (p), and the *declaration* holds a single statement to set the color property to the value *blue* (color: blue;).


```
<style>
p {
  color: blue;
}
</style>
```

# The Class Attribute

- The *class selector* is essential for a stylesheet to work well. Web developers typically assign classes to make semantic distinctions in how content is displayed.
- For example, you might want “warning” text in red. In CSS we can write an instruction to set the font color to red for all elements where `class=“warning”` using a class selector:  

```
.warning {color: red;}
```
- We can then reference the “warning” class in an HTML file, such as:  

```
<p>This is normal paragraph text.</p>  
<p class=“warning”>Caution! Warning paragraph.</p>
```



This is normal paragraph text.  
Caution! Warning paragraph.
- Later in this unit we will make simple edits to some of the style rules in *style.css*, our default CSS file. For more information on stylesheets, see <https://www.w3schools.com/>

# HTML Troubleshooting Tips

---

- Before we begin the practice sections of this unit, there are a few things to look out for if you're new to HTML editing.
- ⚠ HTML does not accept *smart quotes*, the left- and right-hand versions of quotation marks and apostrophes (sometimes called curly quotes). If you copy and paste text from a program like Word (or even this PowerPoint file!), it may introduce errors into the HTML. You can avoid this by typing text manually, or pasting from a plain text editor like Notepad++.
- ⚠ CSS and most web servers are case-sensitive. We recommend using only lower-case letters in the names of EUI Template pages.

# HTML Troubleshoot Tips (continued)

---

- ⚠ Do not put spaces in the names of files, template pages, or linked images and documents used in the EUI. Like case-sensitivity, this can lead to errors later on.
- ⚠ Check your page in a variety of web browsers. Most browsers have differences in how they display web content. Creating a page that looks the same across browsers (Internet Explorer, Firefox, Chrome, etc.) is one of the challenges of web design.
- 💡 **Careful proofreading is essential.** It takes only a missingspace, slash, semi-colon, smart quote, or improperly capitalized letter to cause an error in rendering a webpage.