HTML5

Presentation by Uplatz

Contact Us: https://training.uplatz.com/

Email: info@uplatz.com

Phone:+44 7836 212635

Uplatz

Table Of Contents:

Column Groups:

- Sometimes you may want to apply styling to a column or group of columns.
- Or for semantic purposes, you may want to group columns together.
- To do this, use and elements.
- The optional tag allows you to group columns together. elements must be child elements of a and must come after any elements and before any table content (e.g., , , , etc.). ...
- The optional tag allows you to reference individual columns or a range of columns without applying a logical grouping. elements are optional, but if present, they must be inside a element.
- The following CSS styles can be applied to and elements:

- border
- Background
- width
- visibility
- display (as in display: none)
- display: none; will actually remove the columns from the display, causing the table to render as if those cells don't exist
- Table with thead, tbody, tfoot, and caption
- HTML also provides the tables with the <thead>, <tbody>, <tfoot>, and <caption> elements.
- These additional elements are useful for adding semantic value to your tables and for providing a place for separate CSS styling.

- When printing out a table that doesn't fit onto one (paper) page, most browsers repeat the contents of <thead> on every page.
- There's a specific order that must be adhered to, and we should be aware that not every element falls into place as one would expect.
- The following example demonstrates how our 4 elements should be placed.

<table>

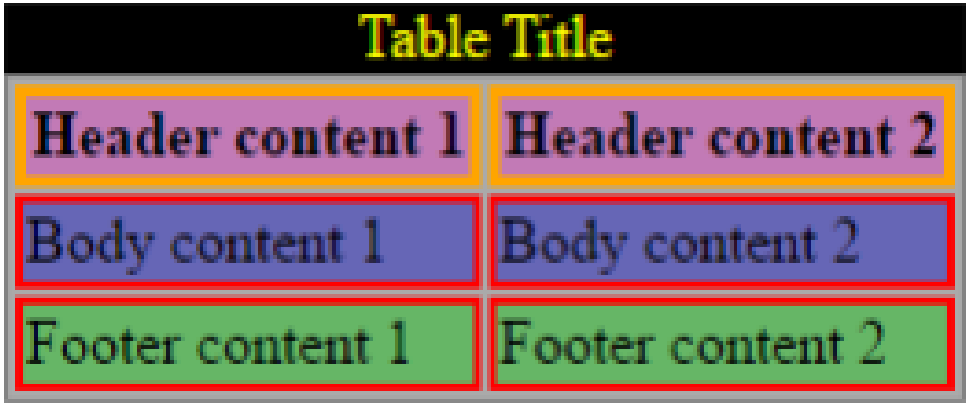
<caption>Table Title</caption> <!-- | caption is the first child of table |-->

<thead> <!--========================| thead is after caption |-->

```
<tr>
 <th>Header content 1</th>
 <th>Header content 2</th>
 </tr>
 </thead>
 <tbody> <!--=========================| tbody is after thead |-->
 <tr>
 <td>Body content 1</td>
 <td>Body content 2</td>
 </tr>
 </tbody>
 <tfoot><!--| tfoot can be placed before or after tbody, but not in a group of tbody. |-->
```

```
<!-- | Regardless where tfoot is in markup, it is rendered at the bottom. | -->
<tr>
 <td>Footer content 1</td>
 <td>Footer content 2</td>
 </tr>
 </tfoot>
</table>
```

- The following example's results are demonstrated twice--the first table lacks any styles, the second table has a few CSS properties applied: background-color, color, and border*.
- The styles are provided as a visual guide and is not an essential aspect of the topic at hand.



Element Styles	Applies
<caption>	Yellow text on black background.
<thead>	Bold text on purple background.
<tbody>	Text on blue background.
<tfoot>	Text on green background.
<th>	Orange borders.
<td>	Red borders.

Heading scope

th elements are very commonly used to indicate headings for table rows and columns, like so:

```
<table>
 <thead>
 <tr>
 <td></td>
 <th>Column Heading 1</th>
 <th>Column Heading 2</th>
 </tr>
 </thead>
 <tbody>
 <tr>
 <th>Row Heading 1</th>
 <td></td>
 <td></td>  </tr>
```

```
<tr>
 <th>Row Heading 2</th>
 <td></td>
 <td></td>
 </tr>
 </tbody>
</table>
```

- This can be improved for accessibility by the use of the scope attribute.
- The above example would be amended as follows:

```
<table>
 <thead>
 <tr>
```

```
<td></td>
 <th scope="col">Column Heading 1</th>
 <th scope="col">Column Heading 2</th>
</tr>
 </thead>
 <tbody>
 <tr>
 <th scope="row">Row Heading 1</th>
 <td></td>
 <td></td>
 </tr>
 <tr>
 <th scope="row">Row Heading 1</th>
 <td></td>
 <td></td>
 </tr> </tbody> </table>
```

Comments:

- Similar to other programming, markup, and markdown languages, comments in HTML provide other developers
- with development specific information without affecting the user interface.
- Unlike other languages however, HTML comments can be used to specify HTML elements for Internet Explorer only. comments, and their functional applications.

Creating comments:

- HTML comments can be used to leave notes to yourself or other developers about a specific point in code.
- They can be initiated with <!-- and concluded with -->, like so:

<!-- I'm an HTML comment! -->

They can be incorporated inline within other content:

<h1>This part will be displayed <!-- while this will not be displayed -->.</h1>

- They can also span multiple lines to provide more information:

<!-- This is a multiline HTML comment.

- Whatever is in here will not be rendered by the browser.
- You can "comment out" entire sections of HTML code.
- However, they cannot appear within another HTML tag, like this:

<h1 <!-- testAttribute="something" -->>This will not work</h1>

- This produces invalid HTML as the entire
- <h1 <!-- testAttribute="something" --> block would be considered a single start tag h1 with some other invalid information contained within it, followed by a single > closing bracket that does nothing.
- Commenting out whitespace between inline elements Inline display elements, usually such as span or a, will include up to one white-space character before and after them in the document.
- In order to avoid very long lines in the markup (that are hard to read) and unintentional white-space (which affects formatting), the white-space can be commented out.
- Try it without a comment between the inline elements, and there will be one space between them.

- Sometimes picking up the space character is desired. Example code:

a href="#">I hope there will be no extra whitespace after this!<!--

--><button>Foo</button>

<hr>

<!-- Without it, you can notice a small formatting difference: -->

I hope there will be no extra whitespace after this!

<button>Foo</button>

Classes and IDs:

Parameter Details

Class Indicates the Class of the element (non-unique)

id Indicates the ID of the element (unique in the same context)

- Classes and IDs make referencing HTML elements from scripts and stylesheets easier.
- The class attribute can be used on one or more tags and is used by CSS for styling.
- IDs however are intended to refer to a single element,meaning the same ID should never be used twice.
- IDs are generally used with JavaScript and internal document links, and are discouraged in CSS.
- This topic contains helpful explanations and examples regarding proper usage of class and ID attributes in HTML.

Giving an element a class

- Classes are identifiers for the elements that they are assigned to.
- Use the class attribute to assign a class to an element.

** **

- To assign multiple classes to an element, separate the class names with spaces.

** **

Using classes in CSS:

- Classes can be used for styling certain elements without changing all elements of that kind.
- For example, these two span elements can have completely different stylings:

****

****

- Classes of the same name can be given to any number of elements on a page and they will all receive the styling associated with that class.
- This will always be true unless you specify the element within the CSS.

For example, we have two elements, both with the class highlight:

** Lorem ipsum **

Lorem ipsum

- If our CSS is as below, then the color green will be applied to the text within both elements:

.highlight { color: green; }

- However, if we only want to target div's with the class highlight then we can add specificity like below:

div.highlight { color: green; }

- Nevertheless, when styling with CSS, it is generally recommended that only classes (e.g. .highlight) be used rather than elements with classes (e.g. div.highlight).

As with any other selector, classes can can be nested:

```
.main .highlight { color: red; } /* Descendant combinator */
.footer > .highlight { color: blue; } /* Child combinator */
```

- You can also chain the class selector to only select elements that have a combination of several classes.

For example, if this is our HTML:

```
<div class="special left menu">This text will be pink</div>
```

- And we want to colour this specific piece of text pink, we can do the following in our CSS:

```
.special.left.menu { color: pink; }
```

Giving an element an ID:

- The ID attribute of an element is an identifier which must be unique in the whole document.
- Its purpose is to uniquely identify the element when linking (using an anchor), scripting, or styling (with CSS).

** **

- You should not have two elements with the same ID in the same document, even if the attributes are attached to two different kinds of elements.
- For example, the following code is incorrect:

** **

****

- Browsers will do their best to render this code, but unexpected behavior may occur when styling with CSS or adding functionality with JavaScript.
- To reference elements by their ID in CSS, prefix the ID with #.

#example-id { color: green; }

- To jump to an element with an ID on a given page, append # with the element name in the URL.

http://example.com/about#example-id

Acceptable Values:

- **For an ID**

The only restrictions on the value of an id are:

- 1. it must be unique in the document
- 2. it must not contain any space characters
- 3. it must contain at least one character

- So the value can be all digits, just one digit, just punctuation characters, include special characters, whatever. Just no whitespace.
- So these are valid:

** ... **

** ... **

** ... **

** ... **

** ... **

** ... **

** ... **

** ... **

** ... **

Uplatz

This is invalid:

** ... **

This is also invalid, when included in the same document:

** ... **

** ... **

An id value must begin with a letter, which can then be followed only by:

letters (A-Z/a-z)

digits (0-9)

hyphens ("-")

underscores ("_")

colons (":")

periods (".")

Referring to the first group of examples in the HTML5 section above, only one is valid:

** ... **

These are also valid:

** ... **

** ... **

** ... **

** ... **

** ... **

Again, if it doesn't start with a letter (uppercase or lowercase), it's not valid.

For a Class:

- The rules for classes are essentially the same as for an id. The difference is that class values do not need to be unique in the document.
- Referring to the examples above, although this is not valid in the same document:

<

**div id="results"> ... **

** ... **

This is perfectly okay:

** ... **

** ... **

- Important Note: How ID and Class values are treated outside of HTML
- Keep in mind that the rules and examples above apply within the context of HTML.
- Using numbers, punctuation or special characters in the value of an id or a class may cause trouble in other
- contexts, such as CSS, JavaScript and regular expressions.
- For example, although the following id is valid in HTML5:

** ... **

- **... it is invalid in CSS:**
  - Problems related to duplicated IDs Having more than one element with the same ID is a hard to troubleshoot problem.
  - The HTML parser will usually try to render the page in any case.
  - Usually no error occurs.
  - But the pace could end up in a mis-behaving web page.
  - In this example: ab
  - CSS selectors still work

#aDiv { color: red; }

  - But JavaScript fails to handle both elements:

var html = document.getElementById("aDiv").innerHTML;

In this casehtml variable bears only the first div content ("a").

Data Attributes:

Value Description

somevalue Specifies the value of the attribute (as a string)

Older browsers support:

- Data attributes were introduced in HTML5 which is supported by all modern browsers, but older browsers before HTML5 don't recognize the data attributes.
- However, in HTML specifications, attributes that are not recognized by the browser must be left alone and the browser will simply ignore them when rendering the page.

- Web developers have utilized this fact to create non-standard attributes which are any attributes not part of the
- HTML specifications. For example, the value attribute in the line bellow is considered a non-standard attribute

because the specifications for the <img> tag don't have a value attribute and it is not a global attribute:

****

- This means that although data attributes are not supported in older browsers, they still work and you can set and retrieve them using the same generic JavaScript setAttribute and getAttribute methods, but you cannot use the new dataset property which is only supported in modern browsers.

Data Attribute Use

- HTML5 data-* attributes provide a convenient way to store data in HTML elements.
- The stored data can be read or modified using JavaScript

```
<div data-submitted="yes" class="user_profile">
 … some content …
</div>
```

- Data attribute structure is data-*, i.e. the name of the data attribute comes after the data- part.
- Using this name, the attribute can be accessed.
- Data in string format (including json) can be stored using data-* attribute.

JavaScript

Synchronous

<script src="path/to.js"></script>

- Standard practice is to place JavaScript <script> tags just before the closing </body> tag.
- Loading your scripts last allows your site's visuals to show up more quickly and discourages your JavaScript from trying to interact with elements that haven't loaded yet.

Asynchronous

<script src="path/to.js" async></script>

- Another alternative, when the Javascript code being loaded is not necessary for page initialization, it can be loaded asynchronously, speeding up the page load.

- Using async the browser will load the contents of the script in parallel and, once it is fully downloaded, will interrupt the HTML parsing in order to parse the Javascript file.

Deferred

<script src="path/to.js" defer></script>

- Deferred scripts are like async scripts, with the exception that the parsing will only be performed once the HTML is fully parsed.
- Deferred scripts are guaranteed to be loaded in the order of declaration, same way as synchronous scripts.

<noscript>

<noscript>JavaScript disabled</noscript>

- The <noscript> element defines content to be displayed if the user has scripts disabled or if the browser does not support using scripts.
- The <noscript> tag can be placed in either the <head> or the <body>.

External CSS Stylesheet:

- The standard practice is to place CSS tags inside the tag at the top of your HTML.
- This way the CSS will be loaded first and will apply to your page as it is loading, rather than showing unstyled HTML until the CSS is loaded.
- The typeattribute is not necessary in HTML5, because HTML5 usually supports CSS. and ... do the same thing in HTML5.

Another, though less common practice, is to use an @import statement inside direct CSS. Like this:

```
<style type="text/css">
 @import("path/to.css")
</style>
<style>
 @import("path/to.css")
</style>
 Favicon:
<link rel="icon" type="image/png" href="/favicon.png">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
```

Use the mime-type image/png for PNG files and image/x-icon for icon (*.ico) files

- A file named favicon.ico at the root of your website will typically be loaded and applied automatically, without the need for a <link> tag.
- If this file ever changes, browsers can be slow and stubborn about updating their cache.

Alternative CSS :

- Some browsers allow alternate style sheets to apply if they are offered.
- By default they will not be applied, but usually they can be changed through the browser settings:
- Firefox lets the user select the stylesheet using the View > Page Style submenu, Internet Explorer also supports this feature (beginning with IE 8), also accessed from View > Page Style (at least as of IE 11), but Chrome requires an extension to use the feature (as of version 48).

The web page can also provide its own user interface to let the user switch styles.

Resource Hint: dns-prefetch, prefetch, prerender

Preconnect

- The preconnect relationship is similar to dns-prefetch in that it will resolve the DNS. However, it will also make the TCP handshake, and optional TLS negotiation.
- This is an experimental feature.

<link rel="preconnect" href="URL">

DNS-Prefetch:

Informs browsers to resolve the DNS for a URL, so that all assets from that URL load faster.

<link rel="dns-prefetch" href="URL">

Prefetch:

- Informs the browsers that a given resource should be prefetched so it can be loaded more quickly.

<link rel="prefetch" href="URL">

- DNS-Prefetch resolves only the domain name whereas prefetch downloads/stores the specified resources.

Prerender:

- Informs browsers to fetch and render the URL in the background, so that they can be delivered to the user instantaneously as the user navigates to that URL.
- This is an experimental feature.

<link rel="prerender" href="URL">

Link 'media' attribute:

<link rel="stylesheet" href="test.css" media="print">

- Media specifies what style sheet should be used for what type of media.
- Using the print value would only display that style sheet for print pages.
- The value of this attribute can be any of the mediatype values (similar to a CSS media query).

Prev and Next:

- When a page is part of a series of articles, for instance, one can use prev and next to point to pages that are coming before and after.

<link rel="prev" href="http://stackoverflow.com/documentation/java/topics">

<link rel="next" href="http://stackoverflow.com/documentation/css/topics">

Web Feed:

Use the rel="alternate" attribute to allow discoverability of your Atom/RSS feeds.

<link rel="alternate" type="application/atom+xml" href="http://example.com/feed.xml" />

<link rel="alternate" type="application/rss+xml" href="http://example.com/feed.xml" />

Include JavaScript Code in HTML:

Attribute Details

src Specifies the path to a JavaScript file. Either a relative or absolute URL.

type Specifies the MIME type. This attribute is required in HTML4, but optional in HTML5.

Async Specifies that the script shall be executed asynchronously (only for external scripts). This attribute does not require any value (except of XHTML).

defer Specifies that the script shall be executed when the page has finished parsing (only for external scripts). This attribute does not require any value (except of XHTML).

charset Specifies the character encoding used in an external script file, e.g. UTF-8

crossorigin How the element handles crossorigin requests

nonce Cryptographic nonce used in Content Security Policy checks

Handling disabled Javascript

- It is possible that the client browser does not support Javascript or have Javascript execution disabled, perhaps due to security reasons. To be able to tell users that a script is supposed to execute in the page, the <noscript> tag can

be used.

- The content of <noscript> is displayed whenever Javascript is disabled for the current page.

<script>

document.write("Hello, world!");

</script>

<noscript>

- This browser does not support Javascript.</noscript>

Linking to an external JavaScript file:

<script src="example.js"></script>

- The src attribute works like the href attribute on anchors: you can either specify an absolute or relative URL.
- The example above links to a file inside the same directory of the HTML document.
- This is typically added inside the <head> tags at the top of the html document

Summary:

- Comments
- Classes and IDs
- Data Attributes
- Linking Resources
- Include JavaScript Code in HTML

Thank You……

If you have any quries please write to info@uplatz.com".