Chapter 1: Getting started with HTML

Remarks

HTML (**H**yper**t**ext **M**arkup **L**anguage) is an XML-compliant system of annotating documents with 'tags'. It is used specifically to create content for web pages and web applications, which can then be shared over a network.

Apart from text, the current version of HTML supports many different types of media, including images and videos.

Versions

Version	Specification	Release Date
1.0	N/A	1994-01-01
2.0	RFC 1866	1995-11-24
3.2	W3C: HTML 3.2 Specification	1997-01-14
4.0	W3C: HTML 4.0 Specification	1998-04-24
4.01	W3C: HTML 4.01 Specification	1999-12-24
5	WHATWG: HTML Living Standard	2014-10-28
5.1	W3C: HTML 5.1 Specification	2016-11-01

Examples

Hello World

Introduction

HTML (**H**yper**t**ext **M**arkup **L**anguage) uses a markup system composed of elements which represent specific content. *Markup* means that with HTML you declare *what* is presented to a viewer, not *how* it is presented. Visual representations are defined by Cascading Style Sheets (CSS) and realized by browsers. Still existing elements that allow for such, like e.g. `font`, "are entirely obsolete, and must not be used by authors"[1].

HTML is sometimes called a programming language but it has no logic, so is a **markup language**. HTML tags provide semantic meaning and machine-readability to the content in the page.

An element usually consists of an opening tag (`<element_name>`), a closing tag (`</element_name>`), which contain the element's name surrounded by angle brackets, and the content in between: `<element_name>...content...</element_name>`

There are some HTML elements that don't have a closing tag or any contents. These are called void elements. Void elements include `<img>`, `<meta>`, `<link>` and `<input>`.

Element names can be thought of as descriptive keywords for the content they contain, such as `video`, `audio`, `table`, `footer`.

A HTML page may consist of potentially hundreds of elements which are then read by a web browser, interpreted and rendered into human readable or audible content on the screen.

For this document it is important to note the difference between elements and tags:

Elements: `video`, `audio`, `table`, `footer`

Tags: `<video>`, `<audio>`, `<table>`, `<footer>`, `</html>`, `</body>`

Element insight

Let's break down a tag...

The `<p>` tag represents a common paragraph.

Elements commonly have an opening tag and a closing tag. The opening tag contains the element's name in angle brackets (`<p>`). The closing tag is identical to the opening tag with the addition of a forward slash (`/`) between the opening bracket and the element's name (`</p>`).

Content can then go between these two tags: `<p>This is a simple paragraph.</p>`.

Creating a simple page

The following HTML example creates a simple "Hello World" web page.

HTML files can be created using any text editor. The files must be saved with a `.html` or `.htm`[2] extension in order to be recognized as HTML files.

Once created, this file can be opened in any web browser.

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <title>Hello!</title>
    </head>

    <body>
        <h1>Hello World!</h1>
        <p>This is a simple paragraph.</p>
    </body>

</html>
```

Simple page break down

These are the tags used in the example:

Tag	Meaning
`<!DOCTYPE>`	Defines the HTML version used in the document. In this case it is HTML5. See the doctypes topic for more information.
`<html>`	Opens the page. No markup should come after the closing tag (`</html>`). The `lang` attribute declares the primary language of the page using the ISO language codes (`en` for English). See the Content Language topic for more information.
`<head>`	Opens the head section, which does not appear in the main browser window but mainly contains information *about* the HTML document, called *metadata*. It can also contain imports from external stylesheets and scripts. The closing tag is `</head>`.
`<meta>`	Gives the browser some metadata about the document. The `charset` attribute declares the character encoding. Modern HTML documents should always use UTF-8, even though it is not a requirement. In HTML, the `<meta>` tag does not require a closing tag. See the Meta topic for more information.
`<title>`	The title of the page. Text written between this opening and the closing tag (`</title>`) will be displayed on the tab of the page or in the title bar of the browser.
`<body>`	Opens the part of the document displayed to users, i.e. all the visible or audible content of a page. No content should be added after the closing tag `</body>`.
`<h1>`	A level 1 heading for the page.

Tag	Meaning
	See headings for more information.
`<p>`	Represents a common paragraph of text.

1. ↑ HTML5, 11.2 Non-conforming features
2. ↑ `.htm` is inherited from the legacy DOS three character file extension limit.

Read Getting started with HTML online: https://riptutorial.com/html/topic/217/getting-started-with-html

Chapter 2: Anchors and Hyperlinks

Introduction

Anchor tags are commonly used to link separate webpages, but they can also be used to link between different places in a single document, often within table of contents or even launch external applications. This topic explains the implementation and application of HTML anchor tags in various roles.

Syntax

- `<a href="URL or anchor">Link Text</a>`

Parameters

Parameter	Details
`href`	Specifies the destination address. It can be an absolute or relative URL, or the `name` of an anchor. An absolute URL is the complete URL of a website like http://example.com/. A relative URL points to another directory and/or document inside the same website, e.g. `/about-us/` points to the directory “about-us” inside the root directory (`/`). When pointing to another directory without explicitly specifying the document, web servers typically return the document “index.html” inside that directory.
`hreflang`	Specifies the language of the resource linked by the `href` attribute (which must be present with this one). Use language values from BCP 47 for HTML5 and RFC 1766 for HTML 4.
`rel`	Specifies the relationship between the current document and the linked document. For HTML5, the values must be defined in the specification or registered in the Microformats wiki.
`target`	Specifies where to open the link, e.g. in a new tab or window. Possible values are `_blank`, `_self`, `_parent`, `_top`, and `framename` (deprecated). Forcing such behaviour is not recommended since it violates the control of the user over a website.
`title`	Specifies extra information about a link. The information is most often shown as a tooltip text when the cursor moves over the link. This attribute is not restricted to links, it can be used on almost all HTML tags.
`download`	Specifies that the target will be downloaded when a user clicks on the hyperlink. The value of the attribute will be the name of the downloaded file. There are no restrictions on allowed values, and the browser will automatically detect the

Parameter	Details
	correct file extension and add it to the file (.img, .pdf, etc.). If the value is omitted, the original filename is used.

Examples

Link to another site

This is the basic use of the `<a>` (anchor element) element:

```
<a href="http://example.com/">Link to example.com</a>
```

It creates a hyperlink, to the URL `http://example.com/` as specified by the `href` (hypertext reference) attribute, with the anchor text "Link to example.com". It would look something like the following:

Link to example.com

To denote that this link leads to an external website, you can use the `external` link type:

```
<a href="http://example.com/" rel="external">example site</a>
```

You can link to a site that uses a protocol other than HTTP. For example, to link to an FTP site, you can do,

```
<a href="ftp://example.com/">This could be a link to a FTP site</a>
```

In this case, the difference is that this anchor tag is requesting that the user's browser connect to `example.com` using the File Transfer Protocol (FTP) rather than the Hypertext Transfer Protocol (HTTP).

This could be a link to a FTP site

Open link in new tab/window

```
<a href="example.com" target="_blank">Text Here</a>
```

The `target` attribute specifies where to open the link. By setting it to `_blank`, you tell the browser to open it in a new tab or window (per user preference).

> **SECURITY VULNERABILITY WARNING!**
>
> Using `target="_blank"` gives the opening site partial access to the `window.opener` object via JavaScript, which allows that page to then access and change the `window.opener.location` of *your* page and potentially redirect users to malware or

phishing sites.

Whenever using this for pages you do not control, add `rel="noopener"` to your link to prevent the `window.opener` object from being sent with the request.

Currently, Firefox does not support `noopener`, so you will need to use `rel="noopener noreferrer"` for maximum effect.

Link to an anchor

Anchors can be used to jump to specific tags on an HTML page. The `<a>` tag can point to any element that has an `id` attribute. To learn more about IDs, visit the documentation about Classes and IDs. Anchors are mostly used to jump to a subsection of a page and are used in conjunction with header tags.

Suppose you've created a page (`page1.html`) on many topics:

```
<h2>First topic</h2>
<p>Content about the first topic</p>
<h2>Second topic</h2>
<p>Content about the second topic</p>
```

Once you have several sections, you may want to create a Table of Contents at the top of the page with quick-links (or bookmarks) to specific sections.

If you gave an `id` attribute to your topics, you could then link to them

```
<h2 id="Topic1">First topic</h2>
<p>Content about the first topic</p>
<h2 id="Topic2">Second topic</h2>
<p>Content about the second topic</p>
```

Now you can use the anchor in your table of contents:

```
<h1>Table of Contents</h1>
    <a href='#Topic1'>Click to jump to the First Topic</a>
    <a href='#Topic2'>Click to jump to the Second Topic</a>
```

These anchors are also attached to the web page they're on (`page1.html`). So you can link across the site from one page to the other by referencing the page *and* anchor name.

```
 Remember, you can always <a href="page1.html#Topic1">look back in the First Topic</a> for
supporting information.
```

Link that runs JavaScript

Simply use the `javascript:` protocol to run the text as JavaScript instead of opening it as a normal link:

```
<a href="javascript:myFunction();">Run Code</a>
```

You can also achieve the same thing using the `onclick` attribute:

```
<a href="#" onclick="myFunction(); return false;">Run Code</a>
```

The `return false;` is necessary to prevent your page from scrolling to the top when the link to `#` is clicked. Make sure to include all code you'd like to run before it, as returning will stop execution of further code.

Also noteworthy, you can include an exclamation mark `!` after the hashtag in order to prevent the page from scrolling to the top. This works because any invalid slug will cause the link to not scroll *anywhere* on the page, because it couldn't locate the element it references (an element with `id="!"`). You could also just use any invalid slug (such as `#scrollsNowhere`) to achieve the same effect. In this case, `return false;` is not required:

```
<a href="#!" onclick="myFunction();">Run Code</a>
```

> **Should you be using any of this?**
>
> The answer is almost certainly ***no***. Running JavaScript inline with the element like this is fairly bad practice. Consider using pure JavaScript solutions that look for the element in the page and bind a function to it instead. Listening for an event
>
> Also consider whether this element is really a *button* instead of a *link*. If so, you should use `<button>`.

Link to a page on the same site

You can use a relative path to link to pages on the same website.

```
<a href="/example">Text Here</a>
```

The above example would go to the file `example` at the root directory (`/`) of the server.

If this link was on http://example.com, the following two links would bring the user to the same location

```
<a href="/page">Text Here</a>
<a href="http://example.com/page">Text Here</a>
```

Both of the above would go to the `page` file at the root directory of `example.com`.

Link that runs email client

Basic usage

If the value of the `href`-attribute begins with `mailto:` it will try to open an email client on click:

```
<a href="mailto:example@example.com">Send email</a>
```

This will put the email address `example@example.com` as the recipient for the newly created email.

Cc and Bcc

You can also add addresses for cc- or bcc-recipients using the following syntax:

```
<a href="mailto:example@example.com?cc=john@example.com&bcc=jane@example.com">Send email</a>
```

Subject and body text

You can populate the subject and body for the new email as well:

```
<a href="mailto:example@example.com?subject=Example+subject&body=Message+text">Send email</a>
```

Those values must be URL encoded.

Clicking on a link with `mailto:` will try to open the default email client specified by your operating system or it will ask you to choose what client you want to use. Not all options specified after the recipient's address are supported in all email clients.

Link that dials a number

If the value of the `href`-attribute begins with `tel:`, your device will dial the number when you click it. This works on mobile devices or on computers/tablets running software – like Skype or FaceTime – that can make phone calls.

```
<a href="tel:11234567890">Call us</a>
```

Most devices and programs will prompt the user in some way to confirm the number they are about to dial.

Read Anchors and Hyperlinks online: https://riptutorial.com/html/topic/254/anchors-and-hyperlinks

Chapter 3: ARIA

Syntax

- aria-live
- aria-relevant
- aria-autocomplete
- aria-checked
- aria-disabled
- aria-expanded
- aria-haspopup
- aria-hidden
- aria-invalid
- aria-label
- aria-level
- aria-multiline
- aria-multiselectable
- aria-orientation
- aria-pressed
- aria-readonly
- aria-required
- aria-selected
- aria-sort
- aria-valuemax
- aria-valuemin
- aria-valuenow
- aria-valuetext
- aria-atomic
- aria-busy
- aria-dropeffect
- aria-dragged
- aria-activedescendant
- aria-controls
- aria-describedby
- aria-flowto
- aria-labelledby
- aria-owns
- aria-posinset
- aria-setsize

Remarks

ARIA is a specification for semantically describing rich web applications. Following ARIA standards can increase accessibility for those using assistive technologies (such as a screen reader) to access your content.

Examples

role="alert"

A message with important, and usually time-sensitive, information.

```
<div role="alert" aria-live="assertive">Your session will expire in 60 seconds.</div>
```

> Note that I've included both `role="alert"` and `aria-live="assertive"` at the same time. These are synonymous attributes, but some screen readers only support one or the other. By using both simultaneously we therefore maximize the chances that the live region will function as expected.
>
> Source - Heydon Pickering 'Some practical ARIA examples'

role="alertdialog"

A type of dialog that contains an alert message, where initial focus goes to an element within the dialog.

```
<div role="alertdialog">
  <h1>Warning</h1>
  <div role="alert">Your session will expire in 60 seconds.</div>
</div>
```

role="application"

A region declared as a web application, as opposed to a web document. In this example, the application is a simple calculator that might add two numbers together.

```
<div role="application">
  <h1>Calculator</h1>
  <input id="num1" type="text"> + <input id="num2" type="text"> =
  <span id="result"></span>
</div>
```

role="article"

A section of a page that consists of a composition that forms an independent part of a document, page, or site.

> Setting an ARIA role and/or aria-* attribute that matches the default implicit ARIA semantics is unnecessary and is not recommended as these properties are already set by the browser.

```
<article>
  <h1>My first article</h1>
  <p>Lorem ipsum...</p>
</article>
```

You would use `role=article` on non-semantic elements (not recommended, invalid)

```
<div role="article">
  <h1>My first article</h1>
  <p>Lorem ipsum...</p>
</div>
```

W3C Entry for `role=article`

role="banner"

A region that contains mostly site-oriented content, rather than page-specific content.

```
<div role="banner">
  <h1>My Site</h1>

  <ul>
    <li><a href="/">Home</a></li>
    <li><a href="/about">About</a></li>
    <li><a href="/contact">Contact</a></li>
  </ul>
</div>
```

role="button"

An input that allows for user-triggered actions when clicked or pressed.

```
<button role="button">Add</button>
```

role="cell"

A cell in a tabular container.

```
<table>
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <td role="cell">95</td>
    <td role="cell">14</td>
    <td role="cell">25</td>
  </tbody>
</table>
```

role="checkbox"

A checkable input that has three possible values: true, false, or mixed.

```
<p>
  <input type="checkbox" role="checkbox" aria-checked="false">
  I agree to the terms
</p>
```

role="columnheader"

A cell containing header information for a column.

```
<table role="grid">
  <thead>
    <tr>
      <th role="columnheader">Day 1</th>
      <th role="columnheader">Day 2</th>
      <th role="columnheader">Day 3</th>
    </tr>
  </thead>
  <tbody>
    <!-- etc -->
  </tbody>
<table>
```

role="combobox"

A presentation of a select; usually similar to a textbox where users can type ahead to select an option, or type to enter arbitrary text as a new item in the list.

```
<input type="text" role="combobox" aria-expanded="false">
```

Typically, you would use JavaScript to build the rest of the typeahead or list select functionality.

role="complementary"

A supporting section of the document, designed to be complementary to the main content at a similar level in the DOM hierarchy, but remains meaningful when separated from the main content.

```
<div role="complementary">
  <h2>More Articles</h2>

  <ul>
    <!-- etc -->
  </ul>
</div>
```

role="contentinfo"

A large perceivable region that contains information about the parent document.

```
<p role="contentinfo">
```

```
  Author: Albert Einstein<br>
  Published: August 15, 1940
</p>
```

role="definition"

A definition of a term or concept.

```
<span role="term" aria-labelledby="def1">Love</span>
<span id="def1" role="definition">an intense feeling of deep affection.</span>
```

role="dialog"

A dialog is an application window that is designed to interrupt the current processing of an application in order to prompt the user to enter information or require a response.

```
<div role="dialog">
  <p>Are you sure?</p>
  <button role="button">Yes</button>
  <button role="button">No</button>
</div>
```

role="directory"

A list of references to members of a group, such as a static table of contents.

```
<ul role="directory">
  <li><a href="/chapter-1">Chapter 1</a></li>
  <li><a href="/chapter-2">Chapter 2</a></li>
  <li><a href="/chapter-3">Chapter 3</a></li>
</ul>
```

role="document"

A region containing related information that is declared as document content, as opposed to a web application.

```
<div role="document">
  <h1>The Life of Albert Einstein</h1>
  <p>Lorem ipsum...</p>
</div>
```

role="form"

A landmark region that contains a collection of items and objects that, as a whole, combine to create a form.

Using the semantically correct HTML element `<form>` implies default ARIA semantics, meaning `role=form` is not required as you should not apply a contrasting role to an element that is already semantic, as adding a role overrides the native semantics of an element.

> Setting an ARIA role and/or aria-* attribute that matches the default implicit ARIA semantics is unnecessary and is not recommended as these properties are already set by the browser.

```
<form action="">
  <fieldset>
    <legend>Login form</legend>
    <div>
      <label for="username">Your username</label>
      <input type="text" id="username" aria-describedby="username-tip" required />
      <div role="tooltip" id="username-tip">Your username is your email address</div>
    </div>
    <div>
      <label for="password">Your password</label>
      <input type="text" id="password" aria-describedby="password-tip" required />
      <div role="tooltip" id="password-tip">Was emailed to you when you signed up</div>
    </div>
  </fieldset>
</form>
```

You would use `role=form` on non-semantic elements (not recommended, invalid)

```
<div role=form>
  <input type="email" placeholder="Your email address">
  <button>Sign up</button>
</div>
```

role="grid"

A grid is an interactive control which contains cells of tabular data arranged in rows and columns, like a table.

```
<table role="grid">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <!-- etc -->
  </tbody>
</table>
```

role="gridcell"

A cell in a grid or treegrid.

```
<table role="grid">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <tr>
      <td role="gridcell">17</td>
      <td role="gridcell">64</td>
      <td role="gridcell">18</td>
    </tr>
  </tbody>
<table>
```

role="group"

A set of user interface objects which are not intended to be included in a page summary or table of contents by assistive technologies.

```
<div role="group">
  <button role"button">Previous</button>
  <button role"button">Next</button>
</div>
```

role="heading"

A heading for a section of the page.

```
<h1 role="heading">Introduction</h1>
<p>Lorem ipsum...</p>
```

role="img"

A container for a collection of elements that form an image.

```
<figure role="img">
  <img alt="A cute cat." src="albert.jpg">
  <figcaption>This is my cat, Albert.</figcaption>
<figure>
```

role="link"

An interactive reference to an internal or external resource that, when activated, causes the user agent to navigate to that resource.

> In the majority of cases setting an ARIA role and/or aria-* attribute that matches the default implicit ARIA semantics is unnecessary and not recommended as these properties are already set by the browser.
>
> Source - https://www.w3.org/TR/html5/dom.html#aria-usage-note

role="list"

A group of non-interactive list items.

```
<ul role="list">
  <li role="listitem">One</li>
  <li role="listitem">Two</li>
  <li role="listitem">Three</li>
</ul>
```

role="listbox"

A widget that allows the user to select one or more items from a list of choices.

```
<ul role="listbox">
  <li>One</li>
  <li>Two</li>
  <li>Three</li>
</ul>
```

Typically, you would use JavaScript to build the multiple-selection functionality.

role="listitem"

A single item in a list or directory.

```
<ul role="list">
  <li role="listitem">One</li>
  <li role="listitem">Two</li>
  <li role="listitem">Three</li>
</ul>
```

role="log"

A type of live region where new information is added in meaningful order and old information may disappear.

```
<ul role="log">
  <li>User 1 logged in.</li>
  <li>User 2 logged in.</li>
  <li>User 1 logged out.</li>
</ul>
```

role="main"

The main content of a document.

```
<!-- header & nav here -->
<div role="main">
  <p>Lorem ipsum...</p>
</div>
<!-- footer here -->
```

role="marquee"

A type of live region where non-essential information changes frequently.

```
<ul role="marquee">
  <li>Dow +0.26%</li>
  <li>Nasdaq +0.54%</li>
  <li>S&amp;P +0.44%</li>
</ul>
```

role="math"

Content that represents a mathematical expression.

```
<img role="math" alt="y=mx+b" src="slope.png">
```

role="menu"

A type of widget that offers a list of choices to the user.

```
<ul role="menu">
  <li role="menuitem">New</li>
  <li role="menuitem">Open</li>
  <li role="menuitem">Save</li>
  <li role="menuitem">Close</li>
</ul>
```

role="menubar"

A presentation of menu that usually remains visible and is usually presented horizontally.

```
<ul role="menubar">
  <li role="menuitem">File</li>
  <li role="menuitem">Edit</li>
  <li role="menuitem">View</li>
  <li role="menuitem">Help</li>
</ul>
```

role="menuitem"

An option in a group of choices contained by a menu or menubar.

```
<ul role="menubar">
  <li role="menuitem">File</li>
  <li role="menuitem">Edit</li>
  <li role="menuitem">View</li>
  <li role="menuitem">Help</li>
</ul>
```

role="menuitemcheckbox"

A checkable menuitem that has three possible values: true, false, or mixed.

```
<ul role="menu">
```

```
  <li role="menuitem">Console</li>
  <li role="menuitem">Layout</li>
  <li role="menuitemcheckbox" aria-checked="true">Word wrap</li>
</ul>
```

role="menuitemradio"

A checkable menuitem in a group of menuitemradio roles, only one of which can be checked at a time.

```
<ul role="menu">
  <li role="menuitemradio" aria-checked="true">Left</li>
  <li role="menuitemradio" aria-checked="false">Center</li>
  <li role="menuitemradio" aria-checked="false">Right</li>
</ul>
```

role="navigation"

A collection of navigational elements (usually links) for navigating the document or related documents.

```
<ul role="navigation">
  <li><a href="/">Home</a></li>
  <li><a href="/about">About</a></li>
  <li><a href="/contact">Contact</a></li>
</ul>
```

role="note"

A section whose content is parenthetic or ancillary to the main content of the resource.

```
<p>Lorem ipsum...</p>
<p>Lorem ipsum...</p>
<p role="note">Lorem ipsum...</p>
```

role="option"

A selectable item in a select list.

```
<ul role="listbox">
  <li role="option">Option 1</li>
  <li role="option">Option 2</li>
  <li role="option">Option 3</li>
</ul>
```

role="presentation"

An element whose implicit native role semantics will not be mapped to the accessibility API.

```
<div style="float:left;">Some content on the left.</div>
<div style="float:right;">Some content on the right</div>
<div role="presentation" style="clear:both;"></div> <!-- Only used to clear floats -->
```

role="progressbar"

An element that displays the progress status for tasks that take a long time.

```
<progress role="progressbar" value="25" max="100">25%</progress>
```

role="radio"

A checkable input in a group of radio roles, only one of which can be checked at a time.

```
<div role="radiogroup">
  <input role="radio" type="radio" aria-checked="true"> One<br>
  <input role="radio" type="radio" aria-checked="false"> Two<br>
  <input role="radio" type="radio" aria-checked="false"> Three
</div>
```

role="region"

A large perceivable section of a web page or document, that the author feels is important enough to be included in a page summary or table of contents, for example, an area of the page containing live sporting event statistics.

```
<div role="region">
  Home team: 4<br>
  Away team: 2
</div>
```

role="radiogroup"

A group of radio buttons.

```
<div role="radiogroup">
  <input role="radio" type="radio" aria-checked="true"> One<br>
  <input role="radio" type="radio" aria-checked="false"> Two<br>
  <input role="radio" type="radio" aria-checked="false"> Three
</div>
```

role="row"

A row of cells in a tabular container.

```
<table>
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <tr role="row">
      <!-- etc -->
    </tr>
  </tbody>
</table>
```

role="rowgroup"

A group containing one or more row elements in a grid.

```
<table>
  <thead role="rowgroup">
    <!-- etc -->
  </thead>
  <tbody role="rowgroup">
    <!-- etc -->
  </tbody>
</table>
```

role="rowheader"

A cell containing header information for a row in a grid.

```
<table role="grid">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <tr>
      <th role="rowheader">Day 1</th>
      <td>65</td>
    </tr>
    <tr>
      <th role="rowheader">Day 2</th>
      <td>74</td>
    </tr>
  </tbody>
</table>
```

role="scrollbar"

A graphical object that controls the scrolling of content within a viewing area, regardless of whether the content is fully displayed within the viewing area.

```
<div id="content1">Lorem ipsum...</div>
<div
  role="scrollbar"
  aria-controls="content1"
  aria-orientation="vertical"
  aria-valuemax="100"
  aria-valuemin="0"
  aria-valuenow="25">
    <div class="scrollhandle"></div>
```

```
</div>
```

role="search"

A landmark region that contains a collection of items and objects that, as a whole, combine to create a search facility.

```
<div role="search">
  <input role="searchbox" type="text">
  <button role="button">Search</button>
</div>
```

role="searchbox"

A type of textbox intended for specifying search criteria.

```
<div role="search">
  <input role="searchbox" type="text">
  <button role="button">Search</button>
</div>
```

role="separator"

A divider that separates and distinguishes sections of content or groups of menuitems.

```
<p>Lorem ipsum...</p>
<hr role="separator">
<p>Lorem ipsum...</p>
```

role="slider"

A user input where the user selects a value from within a given range.

```
<div
  role="slider"
  aria-valuemax="100"
  aria-valuemin="0"
  aria-valuenow="25">
    <div class="sliderhandle"></div>
</div>
```

role="spinbutton"

A form of range that expects the user to select from among discrete choices.

```
<input
  role="spinbutton"
  aria-valuemax="100"
  aria-valuemin="0"
  aria-valuenow="25"
  type="number"
  value="25">
```

role="status"

A container whose content is advisory information for the user but is not important enough to justify an alert, often but not necessarily presented as a status bar.

```
<div role="status">Online</div>
```

role="switch"

A type of checkbox that represents on/off values, as opposed to checked/unchecked values.

```
<select role="switch" aria-checked="false">
  <option>On</option>
  <option selected>Off</option>
</select>
```

role="tab"

A grouping label providing a mechanism for selecting the tab content that is to be rendered to the user.

```
<ul role="tablist">
  <li role="tab">Introduction</li>
  <li role="tab">Chapter 1</li>
  <li role="tab">Chapter 2</li>
</ul>
```

role="table"

A section containing data arranged in rows and columns. The table role is intended for tabular containers which are not interactive.

```
<table role="table">
  <thead>
    <!-- etc -->
  </thead>
  <tbody>
    <!-- etc -->
  </tbody>
</table>
```

role="tablist"

A list of tab elements, which are references to tabpanel elements.

```
<ul role="tablist">
```

```
    <li role="tab">Introduction</li>
    <li role="tab">Chapter 1</li>
    <li role="tab">Chapter 2</li>
</ul>
```

role="tabpanel"

A container for the resources associated with a tab, where each tab is contained in a tablist.

```
<ul role="tablist">
  <li role="tab">Introduction</li>
  <li role="tab">Chapter 1</li>
  <li role="tab">Chapter 2</li>
</ul>
<div role="tabpanel">
  <!-- etc -->
</div>
```

role="textbox"

Input that allows free-form text as its value.

```
<textarea role="textbox"></textarea>
```

role="timer"

A type of live region containing a numerical counter which indicates an amount of elapsed time from a start point, or the time remaining until an end point.

```
<p>
  <span role="timer">60</span> seconds remaining.
</p>
```

role="toolbar"

A collection of commonly used function buttons represented in compact visual form.

```
<ul role="toolbar">
  <li><img alt="New" src="new.png"></li>
  <li><img alt="Open" src="open.png"></li>
  <li><img alt="Save" src="save.png"></li>
  <li><img alt="Close" src="close.png"></li>
</ul>
```

role="tooltip"

A contextual popup that displays a description for an element.

```
<span aria-describedby="slopedesc">Slope</span>
<div role="tooltip" id="slopedesc">y=mx+b</div>
```

Typically, the tooltip would be hidden. Using JavaScript, the tooltip would be displayed after a delay when the user hovers over the element that it describes.

role="tree"

A type of list that may contain sub-level nested groups that can be collapsed and expanded.

```
<ul role="tree">
  <li role="treeitem">
    Part 1
    <ul>
      <li role="treeitem">Chapter 1</li>
      <li role="treeitem">Chapter 2</li>
      <li role="treeitem">Chapter 3</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 2
    <ul>
      <li role="treeitem">Chapter 4</li>
      <li role="treeitem">Chapter 5</li>
      <li role="treeitem">Chapter 6</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 3
    <ul>
      <li role="treeitem">Chapter 7</li>
      <li role="treeitem">Chapter 8</li>
      <li role="treeitem">Chapter 9</li>
    </ul>
  </li>
</ul>
```

role="treegrid"

A grid whose rows can be expanded and collapsed in the same manner as for a tree.

role="treeitem"

An option item of a tree. This is an element within a tree that may be expanded or collapsed if it contains a sub-level group of treeitems.

```
<ul role="tree">
  <li role="treeitem">
    Part 1
    <ul>
      <li role="treeitem">Chapter 1</li>
      <li role="treeitem">Chapter 2</li>
      <li role="treeitem">Chapter 3</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 2
    <ul>
```

```
      <li role="treeitem">Chapter 4</li>
      <li role="treeitem">Chapter 5</li>
      <li role="treeitem">Chapter 6</li>
    </ul>
  </li>
  <li role="treeitem">
    Part 3
    <ul>
      <li role="treeitem">Chapter 7</li>
      <li role="treeitem">Chapter 8</li>
      <li role="treeitem">Chapter 9</li>
    </ul>
  </li>
</ul>
```

Read ARIA online: https://riptutorial.com/html/topic/2734/aria

Chapter 4: Canvas

Parameters

Attribute	Description
height	Specifies the canvas height
width	Specifies the canvas width

Remarks

- This tag is not compatible with versions of Internet Explorer less than 9. Check caniuse.com for browser compatibility.
- `canvas` is only a container for graphics, and the actual drawing of graphics is done by JavaScript.

Examples

Basic Example

The `canvas` element was introduced in HTML5 for drawing graphics.

```
<canvas id="myCanvas">
   Cannot display graphic. Canvas is not supported by your browser (IE<9)
</canvas>
```

The above will create a transparent HTML`<canvas>` element of 300×150 px in size.

You can use the **canvas** element to draw amazing stuff like shapes, graphs, manipulate images, create engaging games etc. with **JavaScript**.
The `canvas`'s 2D *drawable layer* surface Object is referred to as `CanvasRenderingContext2D`; or from a `HTMLCanvasElement` using the `.getContext("2d")` method:

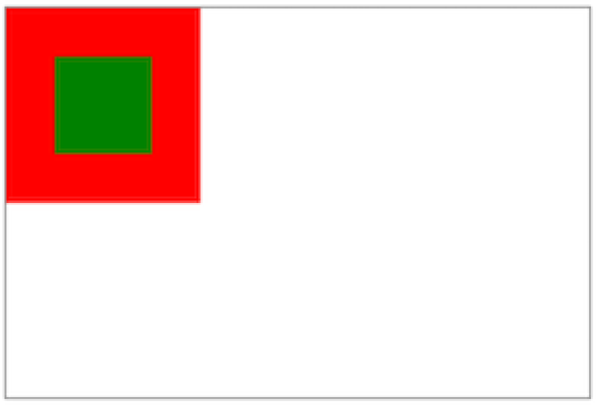
```
var ctx = document.getElementById("myCanvas").getContext("2d");
// now we can refer to the canvas's 2D layer context using `ctx`

ctx.fillStyle = "#f00";
ctx.fillRect(0, 0, ctx.canvas.width, ctx.canvas.height); // x, y, width, height

ctx.fillStyle = "#000";
ctx.fillText("My red canvas with some black text", 24, 32); // text, x, y
```

jsFiddle example

Drawing two rectangles on a

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Draw two rectangles on the canvas</title>
    <style>
      canvas{
          border:1px solid gray;
      }
    </style>
    <script async>
      window.onload = init; // call init() once the window is completely loaded
      function init(){
        // #1 - get reference to <canvas> element
        var canvas = document.querySelector('canvas');

        // #2 - get reference to the drawing context and drawing API
        var ctx = canvas.getContext('2d');

        // #3 - all fill operations are now in red
        ctx.fillStyle = 'red';

        // #4 - fill a 100x100 rectangle at x=0,y=0
        ctx.fillRect(0,0,100,100);

        // #5 - all fill operations are now in green
        ctx.fillStyle = 'green';

        // #6 - fill a 50x50 rectangle at x=25,y=25
        ctx.fillRect(25,25,50,50);

      }
      </script>
</head>
<body>
  <canvas width=300 height=200>Your browser does not support canvas.</canvas>
</body>
</html>
```

This example looks like this:



Read Canvas online: https://riptutorial.com/html/topic/1162/canvas

Chapter 5: Character Entities

Examples

Common Special Characters

Some character may be reserved for HTML and cannot be used directly as it may obstruct the actual HTML codes. For example, trying to display the left and right angle brackets (`<>`) in the source code may cause unexpected results in the output. Similarly, white spaces as written in the source code may not display as expected in the output HTML. Some, like ☎, are not available in the ASCII character set.

For this purpose, character entities are created. These are of the form `&entity_name;` or `&entity_number;`. The following are some of the available HTML entities.

Character	Description	Entity Name	Entity Number
“ ”	non-breaking space	`&nbsp;`	`&#160;`
“<”	less than	`&lt;`	`&#60;`
“>”	greater than	`&gt;`	`&#62;`
“&”	ampersand	`&amp;`	`&#38;`
“—”	em dash	`&mdash;`	`&#8212;`
“–”	en dash	`&ndash;`	`&#8211;`
“©”	copyright	`&copy;`	`&#169;`
“®”	registered trademark	`&reg;`	`&#174;`
“™”	trademark	`&trade;`	`&#8482;`
“☎”	phone	`&phone;`	`&#9742;`

Thus, to write

© 2016 Stack Exchange Inc.

the following HTML code is used:

```
<b>&copy; 2016 Stack Exchange Inc.</b>
```

Character Entities in HTML

Many symbols and special characters are required while developing a web page in html, but as we know that sometimes the use of characters directly may interfere with the actual html code which have certain characters reserved and also certain characters being not available on keyboard. Thus, to avoid the conflict and at same time to be able to use different symbols in our code w3 org provides us with 'Character Entities'.

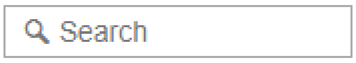
Character Entities are predefined with 'Entity Name' - &entity_name; and 'Entity Number' - &entity_number; so we need to use either of the two for the required symbol to be rendered on our page.

The list of few Character Entities can be found at https://dev.w3.org/html5/html-author/charref

A simple example with the use of character entity for 'magnifying glass' :

```
<input type="text" placeholder="  &#128269; Search"/>
```

which renders as



Read Character Entities online: https://riptutorial.com/html/topic/5229/character-entities

Chapter 6: Classes and IDs

Introduction

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.

Syntax

- class="class1 class2 class3"
- id="uniqueid"

Parameters

Parameter	Details
class	Indicates the Class of the element (non-unique)
id	Indicates the ID of the element (unique in the same context)

Remarks

- Both `class` and `id` are global attributes, and may therefore be assigned to any HTML element.
- Class names must begin with a letter (A-Z or a-z) and can be followed by letters, digits , hyphens and underscores.
- In `HTML5`, the class and id attributes can be used on any element. In HTML 4.0.1, they were off-limits to the `<base>`, `<head>`, `<html>`, `<meta>`, `<param>`, `<script>`, `<style>` and `<title>` tags.
- An element can have one or more classes. Classes are separated by spaces and cannot contain spaces themselves.
- An element can have only one ID and it must be unique within its context (i.e. a webpage). IDs also cannot contain spaces themselves.

Examples

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.

```
<div class="example-class"></div>
```

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

```
<div class="class1 class2"></div>
```

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:

```
<span></span>
<span class="special"></span>
```

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`:

```
<div class="highlight">Lorem ipsum</div>
<span class="highlight">Lorem ipsum</span>
```

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).

```
<div id="example-id"></div>
```

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:

```
<div id="example-id"></div>
<span id="example-id"></span>
```

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
```

This feature is supported in most browsers and does not require additional JavaScript or CSS to work.

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:

```
<div id="aDiv">a</div>
<div id="aDiv">b</div>
```

CSS selectors still work

```
#aDiv {
    color: red;
}
```

But JavaScript fails to handle both elements:

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

In this case`html` variable bears only the first `div` content `("a")`.

Acceptable Values

For an ID

5

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:

```
<div id="container"> ... </div>
<div id="999"> ... </div>
<div id="#%LV-||"> ... </div>
<div id="____V"> ... </div>
<div id="□□"> ... </div>
<div id="♥"> ... </div>
<div id="{}"> ... </div>
<div id="©"> ... </div>
<div id="♤₩¤☆€~¥"> ... </div>
```

This is invalid:

```
<div id=" "> ... </div>
```

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

```
<div id="results"> ... </div>
<div id="results"> ... </div>
```

4.01

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:

```
<div id="container"> ... </div>
```

These are also valid:

```
<div id="sampletext"> ... </div>
<div id="sample-text"> ... </div>
<div id="sample_text"> ... </div>
<div id="sample:text"> ... </div>
<div id="sample.text"> ... </div>
```

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"> ... </div>
<div id="results"> ... </div>
```

This is perfectly okay:

```
<div class="results"> ... </div>
<div class="results"> ... </div>
```

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:

```
<div id="9lions"> ... </div>
```

... it is invalid in CSS:

> **4.1.3 Characters and case**
>
> In CSS, *identifiers* (including element names, classes, and IDs in selectors) can

> contain only the characters [a-zA-Z0-9] and ISO 10646 characters U+00A0 and higher, plus the hyphen (-) and the underscore (_); ***they cannot start with a digit, two hyphens, or a hyphen followed by a digit***. (emphasis added)

In most cases you may be able to escape characters in contexts where they have restrictions or special meaning.

W3C References

- 3.2.5.1 The `id` attribute
- 3.2.5.7 The `class` attribute
- 6.2 SGML basic types

Read Classes and IDs online: https://riptutorial.com/html/topic/586/classes-and-ids