

HTML5

Presentation by Uplatz

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

Email: info@uplatz.com

Phone: +44 7836 212635

Table Of Contents:

- Input Control Elements
- Forms
- Div Element
- Sectioning Elements

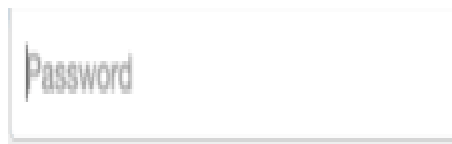
Password:

```
<input type="password" name="password">
```

- The input element with a type attribute whose value is password creates a single-line text field similar to the input type=text, except that text is not displayed as the user enters it.

```
<input type="password" name="password"  
placeholder="Password">
```

- Placeholder text is shown in plain text and is overwritten automatically when a user starts typing



Note: Some browsers and systems modify the default behavior of the password field to also display the most recently typed character for a short duration, like so:



File:

<input type="file" name="fileSubmission">

- File inputs allow users to select a file from their local filesystem for use with the current page.
- If used in conjunction with a form element, they can be used to allow users to upload files to a server (for more info see Uploading Files).

- The following example allows users to use the file input to select a file from their filesystem and upload that file to a script on the server named upload_file.php.

```
<form action="upload_file.php" method="post"  
enctype="multipart/form-data">
```

Select file to upload:

```
<input type="file" name="fileSubmission"  
id="fileSubmission">
```

```
<input type="submit" value="Upload your file"  
name="submit">
```

```
</form>
```

Multiple files:

- Adding the multiple attribute the user will be able to select more than one file:

**<input type="file" name="fileSubmission"
id="fileSubmission" multiple>**

Accept Files:

- Accept attribute specifies the types of files that user can select. E.g. .png, .gif, .jpeg.

**<input type="file" name="fileSubmission"
accept="image/x-png,image/gif,image/jpeg" />**

Button:

<input type="button" value="Button Text">

- Buttons can be used for triggering actions to occur on the page, without submitting the form.
- You can also use the <button> element if you require a button that can be more easily styled or contain other elements:

<button type="button">Button Text</button>

- Buttons are typically used with an "onclick" event:

**<input type="button" onclick="alert('hello world!')"
value="Click Me">**

or

**<button type="button" onclick="alert('hello
world!')">Click Me</button>**

Attributes:

[name]

- The name of the button, which is submitted with the form data.

[type]

- The type of the button.

Possible values are:

submit :

- The button submits the form data to the server.
- This is the default if the attribute is not specified, or if the attribute is dynamically changed to an empty or invalid value.

reset : The button resets all the controls to their initial values.

button : The button has no default behavior. It can have client-side scripts associated with the element's events, which are triggered when the events occur.

menu : The button opens a popup menu defined via its designated element.

[value]

The initial value of the button.

Extra Attributes for Submit Buttons

Attribute	Description
-----------	-------------

form	Specifies the ID of the form the button belongs to. If none is specified, it will belong to its ancestor form element (if one exists).
-------------	--

formaction	Specifies where to send the form-data when the form is submitted using this button.
-------------------	---

Formenctype	Specifies how the form-data should be encoded when submitting it to the server using this button. Can only be used with formmethod="post".
--------------------	--

formmethod	Specifies the HTTP method to use (POST or GET) when sending form-data using this button.
-------------------	--

formnovalidate	Specifies that the form-data should not be validated on submission.
-----------------------	---

formtarget Specifies where to display the response that is received after submitting the form using this button.

Submit:

<input type="submit" value="Submit">

- A submit input creates a button which submits the form it is inside when clicked.
- You can also use the <button> element if you require a submit button that can be more easily styled or contain other elements:

<button type="submit">

** Submit**

</button>

Reset:

<input type="reset" value="Reset">

- An input of type reset creates a button which, when clicked, resets all inputs in the form it is contained in to their default state.
- Text in an input field will be reset to blank or its default value (specified using the value attribute).
- Any option(s) in a selection menu will be deselected unless they have the selected attribute.
- All checkboxes and radio boxes will be deselected unless they have the checked attribute.

Note: A reset button must be inside or attached to (via the form attribute) a <form> element in order to have any effect.

The button will only reset the elements within this form.

Hidden:

<input type="hidden" name="inputName" value="inputValue">

- A hidden input won't be visible to the user, but its value will be sent to the server when the form is submitted nonetheless.

Tel:

<input type="tel" value="+8400000000">

- The input element with a type attribute whose value is tel represents a one-line plain-text edit control for entering a telephone number.

Email:

- The <input type="email"> is used for input fields that should contain an e-mail address.

<form>

<label>E-mail: <label>

<input type="email" name="email">

</form>

- E-mail address can be automatically validated when submitted depending on browser support.

Number:

<input type="number" value="0" name="quantity">

- The Input element with a type attribute whose value is number represents a precise control for setting the element's value to a string representing a number.
- Please note that this field does not guarantee to have a correct number.

- It just allows all the symbols which could be used in any real number, for example user will be able to enter value like $e1e-0$.

Range:

<input type="range" min="" max="" step="" />

A control for entering a number whose exact value is not important.

Attribute	Description	Default value
min	Minimum value for range	0
max	Maximum value for range	100
step	Amount to increase by on each increment.	1

Search:

- Input type search is used for textual search.
- It will add magnifier symbol next to space for text on most browsers

<input type="search" name="googlesearch">

Image:

<input type="image" src="img.png" alt="image_name" height="50px" width="50px"/>

- An Image.
- You must use the src attribute to define the source of the image and the alt attribute to define alternative text.
- You can use the height and width attributes to define the size of the image in pixels.

Week:

<input type="week" />

- Dependent on browser support, a control will show for entering a week-year number and a week number with no time zone.

Url:

<input type="url" name="Homepage">

- This is used for input fields that should contain a URL address.
- Depending on browser support, the url field can be automatically validated when submitted.
- Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

DateTime-Local:

<input type="datetime-local" />

- Dependent on browser support, a date and time picker will pop up on screen for you to choose a date and time.

Month:

<input type="month" />

- Dependent on browser support, a control will show to pick the month

Time:

<input type="time" />

- The time input marks this element as accepting a string representing a time.
- The format is defined in RFC 3339 and should be a partial-time such as

19:04:39

08:20:39.04

- Currently, all versions of Edge, Chrome, Opera, and Chrome for Android support type="time".

- The newer versions of Android Browser, specifically 4.4 and up support it.
- Safari for iOS offers partial support, not supporting min, max, and step attributes.

DateTime (Global):

- The input element with a type attribute whose value is "datetime" represents a control for setting the element's value to a string representing a global date and time (with timezone information).

<fieldset>

<p><label>Meeting time: <input type=datetime name="meeting.start"></label>

</fieldset>

Permitted attributes:

- global attributes
- name
- disabled
- form
- type
- autocomplete
- autofocus
- list
- min & max
- step (float)
- readonly
- required value

Date:

<input type="date" />

- A date picker will pop up on screen for you to choose a date.
- This is not supported in Firefox or Internet Explorer.

Forms:

Attribute	Description
-----------	-------------

accept-charset	Specifies the character encodings that are to be used for the form submission.
-----------------------	--

action	Specifies where to send the form-data when a form is submitted.
---------------	---

autocomplete	Specifies whether a form should have autocomplete on or off.
---------------------	--

Enctype Specifies how the form-data should be encoded when submitting it to the server (only for method="post").

method Specifies the HTTP method to use when sending form-data (POST or GET).

name Specifies the name of a form.

novalidate Specifies that the form should not be validated when submitted.

target Specifies where to display the response that is received after submitting the form.

- In order to group input elements and submit data, HTML uses a form element to encapsulate input and submission elements.
- These forms handle sending the data in the specified method to a page handled by a server or handler.

This topic explains and demonstrates the usage of HTML forms in collecting and submitting input data

Submitting

The Action Attribute

- The action attribute defines the action to be performed when the form is submitted, which usually leads to a script that collects the information submitted and works with it.
- if you leave it blank, it will send it to the same file

<form action="action.php">

The Method Attribute

The method attribute is used to define the HTTP method of the form which is either GET or POST.

<form action="action.php" method="get">

<form action="action.php" method="post">

- The GET method is mostly used to get data, for example to receive a post by its ID or name, or to submit a search query.
- The GET method will append the form data to the URL specified in the action attribute.

www.example.com/action.php?firstname=Mickey&lastname=Mouse

- The POST method is used when submitting data to a script.
- The POST method does not append the form data to the action URL but sends using the request body.
- To submit the data from the form correctly, a name attribute name must be specified.

As an example let's send the value of the field and set its name to lastname:

<input type="text" name="lastname" value="Mouse">

More attributes:

```
<form action="action.php" method="post"  
target="_blank" accept-charset="UTF-8"  
enctype="application/x-www-form-urlencoded"  
autocomplete="off" novalidate>  
</form>
```

Target attribute in form tag:

- The target attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.
- The target attribute defines a name of, or keyword for, a browsing context (e.g. tab, window, or inline frame).

Form Tag with a target attribute:

<form target="_blank">

Attribute Values

Value	Description
-------	-------------

_blank	The response is displayed in a new window or tab
---------------	--

_self	The response is displayed in the same frame (this is default)
--------------	---

_parent	The response is displayed in the parent frame
----------------	---

_top	The response is displayed in the full body of the window
-------------	--

framename	The response is displayed in a named iframe
------------------	---

Note: The target attribute was deprecated in HTML 4.01. The target attribute is supported in HTML5.

➤ Frames and framesets are not supported in HTML5, so the _parent, _top and framename values are now mostly used with iframes.

Uploading Files:

- Images and files can be uploaded/submitted to server by setting enctype attribute of form tag to multipart/formdata.
- enctype specifies how form data would be encoded while submitting to the server.

Example

```
<form method="post" enctype="multipart/form-data"
action="upload.php">
  <input type="file" name="pic" />
  <input type="submit" value="Upload" />
</form>
```

Grouping a few input fields:

- While designing a form, you might like to group a few input fields into a group to help organise the form layout.

- This can be done by using the tag . Here is an example for using it.
- For each fieldset, you can set a legend for the set using the tag LEGEND TEXT

Example

<form>

<fieldset>

<legend>1st field set:</legend>

**Field one:
**

**<input type="text">
**

**Field two:
**

**<input type="text">
**

**</fieldset>
**

<fieldset>

<legend>2nd field set:</legend>

**Field three:
**

**<input type="text">
**

**Field four:
**

**<input type="text">
**

**</fieldset>
**

<input type="submit" value="Submit">

</form>

Div Element:

- The div element in HTML is a container element that encapsulates other elements and can be used to group and separate parts of a webpage.
- A div by itself does not inherently represent anything but is a powerful tool in web design.

Basic usage:

- The `<div>` element usually has no specific semantic meaning by itself, simply representing a division, and is typically used for grouping and encapsulating other elements within an HTML document and separating those from other groups of content.
- As such, each `<div>` is best described by its contents.

`<div>`

`<p>Hello! This is a paragraph.</p>`

`</div>`

- The `div` element is typically a block-level element, meaning that it separates a block of an HTML document and occupying the maximum width of the page.
- Browsers typically have the following default CSS rule:

```
div {  
  display: block;  
}
```

- It's strongly encouraged by the The World Wide Web Consortium (W3C) to view the div element as an
- element of last resort, for when no other element is suitable.
- The use of more appropriate elements instead of the div element leads to better accessibility for readers and easier maintainability for authors.
- For example, a blog post would be marked up using <article>, a chapter using <section>, a page's navigation aids using <nav>, and a group of form controls using <fieldset>.

- div elements can be useful for stylistic purposes or to wrap multiple paragraphs within a section that are all to be annotated in a similar way.

Nesting

- It is a common practice to place multiple `<div>` inside another `<div>`.
- This is usually referred to as "nesting" elements and allows for further dividing elements into subsections or aid developers with CSS styling.
- The `<div class="outer-div">` is used to group together two `<div class="inner-div">` elements; each containing a `<p>` element.

```
<div class="outer-div">
```

```
  <div class="inner-div">
```

```
    <p>This is a paragraph</p>
```

```
  </div>
```

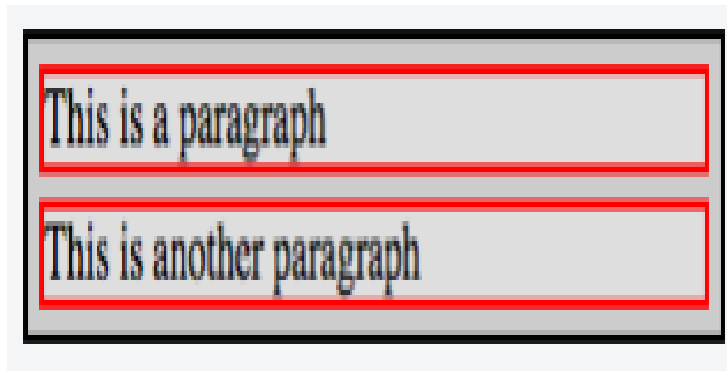
```
<div class="inner-div">
```

```
<p>This is another paragraph</p>
```

```
</div>
```

```
</div>
```

This will yield the following result (CSS styles applied for clarity):



- Nesting inline and block elements While nesting elements you should keep in mind, that there are inline and block elements.

- while block elements "add a line break in the background", what means, other nested elements are shown in the next line automatically, inline elements can be positioned next to each other by default Avoid deep nesting A deep and oftenly used nested container layouts shows a bad coding style.
- Rounded corners or some similar functions often create such an HTML code.
- For most of the last generation browsers there are CSS3 counterparts.
- Try to use as little as possible HTML elements to increase the content to tag ratio and reduce page load, resulting in a better ranking in search engines.
- div section Element should be not nested deeper than 6 layers.

Sectioning Elements:

Nav Element:

- The <nav> element is primarily intended to be used for sections that contain main navigation blocks for the website, this can include links to other parts of the web page (e.g. anchors for a table of contents) or other pages entirely.

Inline items:

The following will display an inline set of hyperlinks.

```
<nav>
```

```
<a href="https://google.com">Google</a>
```

```
<a href="https://www.yahoo.com">Yahoo!</a>
```

```
<a href="https://www.bing.com">Bing</a>
```

```
</nav>
```

Use list items when needed

- If the content represents a list of items, use a list item to show this and enhance the user experience.
- Note the role="navigation", more on this below.

<nav role="navigation">

Google

Yahoo!

Bing

</nav>.

Avoid unnecessary usage:

- <footer> elements may have a list of links to other parts of the site (FAQ, T&C, etc.).
- The footer element alone is sufficient in this case, you don't need to further wrap your links with a <nav> element in the <footer>.

<!-- the <nav> is not required in the <footer> -->

<footer>

<nav>

...

</nav>

</footer>

<!-- The footer alone is sufficient -->

<footer>

...

</footer>

Notes:

- <main> element descendants are not allowed within a <nav>

- Adding a role="navigation" ARIA role to the <nav> element is advised to aid user agents that don't support HTML5 and also to provide more context for those that do

<nav role="navigation"><!-- ... --></nav>

Screen Readers:

- (software that allows blind or visually impaired users to navigate the site)
- User agents like screen readers will interpret the <nav> element differently depending on their requirements.
- It could give the <nav> element a higher priority when rendering the page
- It could delay the rendering of the element
- It could adapt the page in a specific way to tailor for the user's needs

- example: make the text links within the <nav> elements larger for someone who's visually impaired.

Article Element:

- The <article> element contains self-contained content like articles, blog posts, user comments or an interactive widget that could be distributed outside the context of the page, for example by RSS.
- When article elements are nested, the contents of the inner article node should be related to the outer article element.
- A blog (section) with multiple posts (article), and comments (article) might look something like this.

<section>

<!-- Each individual blog post is an <article> -->

<article>

<header>

<h1>Blog Post</h1>

**<time datetime="2016-03-13">13th March
2016</time>**

</header>

**<p>The article element represents a self contained
article or document.</p>**

**<p>The section element represents a grouping of
content.</p>**

<section>

**<h2>Comments <small>relating to "Blog
Post"</small></h2>**

**<!-- Related comment is also a self-contained article
-->**

```
<article id="user-comment-1">
  <p>Excellent!</p>
  <footer><p>...</p><time>...</time></footer>
</article>
</section>
</article>
</section>
<!-- Content unrelated to the blog or posts should
be outside the section. -->
<footer>
  <p>This content should be unrelated to the
blog.</p>
</footer>
```


- When the main content of the page (excluding headers, footers, navigation bars, etc.) is simply one group of elements.
- You can omit the `<article>` in favour of the `<main>` element.

`<article>`

**`<p>This doesn't make sense, this article has no real
`context`.</p>`**

`</article>`

- Instead, replace the article with a `<main>` element to indicate this is the main content for this page.

`<main>`

**`<p>I'm the main content, I don't need to belong to an
article.</p>`**

`</main>`

- If you use another element, ensure you specify the `<main>` ARIA role for correct interpretation and rendering across multiple devices and non HTML5 browsers.

`<section role="main">`

`<p>This section is the main content of this page.</p>`

`</section>`

Main Element:

- The `<main>` element contains the main content for your web page.
- This content is unique to the individual page, and should not appear elsewhere on the site.
- Repeating content like headers, footers, navigation, logos, etc., is placed outside the element.

- The `<main>` element should only ever be used at most once on a single page.
- The `<main>` element must not be included as a descendant of an article, aside, footer, header or nav element.
- In the following example, we're displaying a single blog post (and related information like references and comments).

`<body>`

`<header>`

`<nav>...</nav>`

`</header>`

`<main>`

`<h1>Individual Blog Post</h1>`

`<p>An introduction for the post.</p>`

`<article>`

- If you use another element, ensure you specify the `<main>` ARIA role for correct interpretation and rendering across multiple devices and non HTML5 browsers.

`<section role="main">`

`<p>This section is the main content of this page.</p>`

`</section>`

Main Element:

- The `<main>` element contains the main content for your web page.
- This content is unique to the individual page, and should not appear elsewhere on the site.
- Repeating content like headers, footers, navigation, logos, etc., is placed outside the element.

- In the following example, we're displaying a single blog post (and related information like references and comments).

<body>

<header>

<nav>...</nav>

</header>

<main>

<h1>Individual Blog Post</h1>

<p>An introduction for the post.</p>

<article>

<h2>References</h2>

<p>...</p>

</article>

<article>

<h2>Comments</h2> ...

</article>

</main>

<footer>...</footer>

</body>

- The blog post is contained within the <main> element to indicate this is the main content for this page (and therefore, unique across the website).
- The <header> and <footer> tags are siblings to the <main> element.

Notes:

- The HTML5 specification recognizes the <main> element as a grouping element, and not a sectioning element.

Summary:



Input Control Elements



Forms



Div Element



Sectioning Elements

Thank You.....

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