My HTML Notes

HTML form

[<form>](https://www.w3schools.com/tags/tag_form.asp) Defines an HTML form for user input

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname"><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname">  
</form

HTML Form Attributes

The Action Attribute

The action attribute defines the action to be performed when the form is submitted.

Usually, the form data is sent to a file on the server when the user clicks on the submit button.

## The Target Attribute

The target attribute specifies where to display the response that is received after submitting the form.

The target attribute can have one of the following values:

|  |  |
| --- | --- |
| **Value** | **Description** |
| \_blank | The response is displayed in a new window or tahgbhb |
| \_self | The response is displayed in the current window |
| \_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 |

The default value is \_self which means that the response will open in the current window.

## The Method Attribute

The method attribute specifies the HTTP method to be used when submitting the form data.

The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").

The default HTTP method when submitting form data is GET.

<form action="/action\_page.php" method="get">

**Notes on GET:**

* Appends the form data to the URL, in name/value pairs
* NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
* The length of a URL is limited (2048 characters)
* Useful for form submissions where a user wants to bookmark the result
* GET is good for non-secure data, like query strings in Google

**Notes on POST:**

* Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
* POST has no size limitations, and can be used to send large amounts of data.
* Form submissions with POST cannot be bookmarked

## The Autocomplete Attribute

The autocomplete attribute specifies whether a form should have autocomplete on or off.

When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

<form action="/action\_page.php" autocomplete="on">

## The Novalidate Attribute

The novalidate attribute is a boolean attribute.

When present, it specifies that the form-data (input) should not be validated when submitted.

<form action="/action\_page.php" novalidate>

## List of All <form> Attributes

|  |  |
| --- | --- |
| **,Attribute** | **Description** |
| [accept-charset](https://www.w3schools.com/tags/att_form_accept_charset.asp) | Specifies the character encodings used for form submission |
| [action](https://www.w3schools.com/tags/att_form_action.asp) | Specifies where to send the form-data when a form is submitted |
| [autocomplete](https://www.w3schools.com/tags/att_form_autocomplete.asp) | Specifies whether a form should have autocomplete on or off |
| [enctype](https://www.w3schools.com/tags/att_form_enctype.asp) | Specifies how the form-data should be encoded when submitting it to the server (only for method="post") |
| [method](https://www.w3schools.com/tags/att_form_method.asp) | Specifies the HTTP method to use when sending form-data |
| [name](https://www.w3schools.com/tags/att_form_name.asp) | Specifies the name of the form |
| [novalidate](https://www.w3schools.com/tags/att_form_novalidate.asp) | Specifies that the form should not be validated when submitted |
| [rel](https://www.w3schools.com/tags/att_form_rel.asp) | Specifies the relationship between a linked resource and the current document |
| [target](https://www.w3schools.com/tags/att_form_target.asp) | Specifies where to display the response that is received after submitting the form |

## The <input> Element

[<input>](https://www.w3schools.com/tags/tag_input.asp) Defines an input control

One of the most used form elements is the <input> element.

The <input> element can be displayed in several ways, depending on the type attribute.

## Input Type Text

Tip: The default value of the type attribute is "text".

<input type="text"> defines a single-line text input field:

## Input Type Password

<input type="password"> defines a password field: The characters in a password field are masked (shown as asterisks or circles).

## Input Type Submit

<input type="submit"> defines a button for submitting form data to a form-handler.

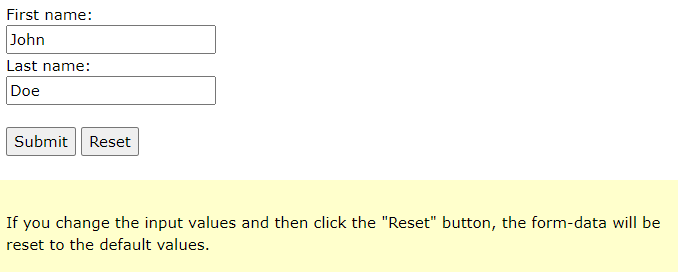
The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's action attribute:

If you omit the submit button's value attribute, the button will get a default text:

## Input Type Reset

<input type="reset"> defines a reset button that will reset all form values to their default values:



## Input Type Radio

<input type="radio"> defines a radio button.

Radio buttons let a user select ONLY ONE of a limited number of choices:



## Input Type Checkbox

<input type="checkbox"> defines a checkbox.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

## Input Type Button

<input type="button"> defines a button:

## Input Type Color

The <input type="color"> is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field.

## Input Type Date

The <input type="date"> is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

<form>  
  <label for="birthday">Birthday:</label>  
  <input type="date" id="birthday" name="birthday">  
</form>

You can also use the min and max attributes to add restrictions to dates:

## Input Type Email

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

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

## Input Type Image

The <input type="image"> defines an image as a submit button.

The path to the image is specified in the src attribute.

<form>  
<input type="image" src="img\_submit.gif" alt="Submit" width="48" height="48">  
</form>

## Input Type File

The <input type="file"> defines a file-select field and a "Browse" button for file uploads.

<form>  
  <label for="myfile">Select a file:</label>  
  <input type="file" id="myfile" name="myfile">  
</form>

## Input Type Hidden

The <input type="hidden"> defines a hidden input field (not visible to a user).

A hidden field lets web developers include data that cannot be seen or modified by users when a form is submitted.

A hidden field often stores what database record that needs to be updated when the form is submitted.

Note: While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

<form>  
  <label for="fname">First name:</label>  
  <input type="text" id="fname" name="fname"><br><br>  
  <input type="hidden" id="custId" name="custId" value="3487">  
  <input type="submit" value="Submit">  
</form>

## Input Type Month

The <input type="month"> allows the user to select a month and year.

Depending on browser support, a date picker can show up in the input field

<form>  
  <label for="bdaymonth">Birthday (month and year):</label>  
  <input type="month" id="bdaymonth" name="bdaymonth">  
</form>

## Input Type Number

The <input type="number"> defines a numeric input field.

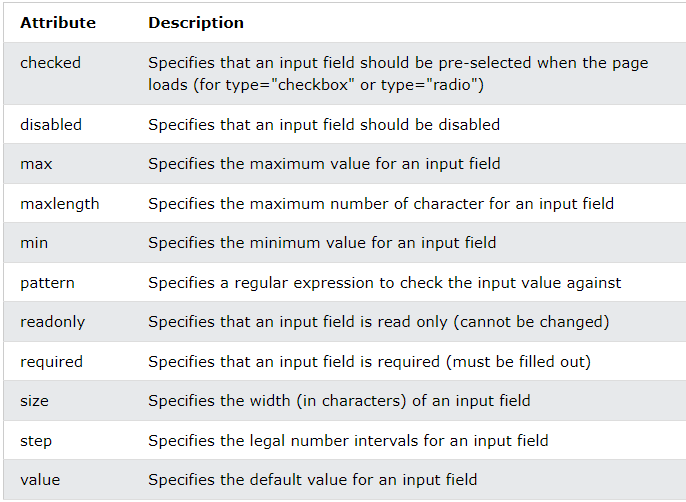
You can also set restrictions on what numbers are accepted.

The following example displays a numeric input field, where you can enter a value from 1 to 5:

<form>  
  <label for="quantity">Quantity (between 1 and 5):</label>  
  <input type="number" id="quantity" name="quantity" min="1" max="5">  
</form>

## Input Restrictions

Here is a list of some common input restrictions:



### **Example**

<form>  
  <label for="quantity">Quantity:</label>  
  <input type="number" id="quantity" name="quantity" min="0" max="100" step="10" value="30">  
</form>

## Input Type Range

The <input type="range"> defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the min, max, and step attributes:

<form>  
  <label for="vol">Volume (between 0 and 50):</label>  
  <input type="range" id="vol" name="vol" min="0" max="50">  
</form>

## Input Type Search

The <input type="search"> is used for search fields (a search field behaves like a regular text field).

<form>  
  <label for="gsearch">Search Google:</label>  
  <input type="search" id="gsearch" name="gsearch">  
</form>

## Input Type Tel

The <input type="tel"> is used for input fields that should contain a telephone number.

<form>  
  <label for="phone">Enter your phone number:</label>  
  <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">  
</form>

## Input Type Url

The <input type="url"> 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.

### **Example**

<form>  
  <label for="homepage">Add your homepage:</label>  
  <input type="url" id="homepage" name="homepage">  
</form>

# **HTML Input Attributes**

This chapter describes the different attributes for the HTML <input> element.

## The value Attribute

The input value attribute specifies an initial value for an input field:

Input fields with initial (default) values:

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" value="John"><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname" value="Doe">  
</form>

## The readonly Attribute

The input readonly attribute specifies that an input field is read-only.

A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it).

The value of a read-only input field will be sent when submitting the form!

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" value="John" readonly><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname" value="Doe">  
</form>

## The disabled Attribute

The input disabled attribute specifies that an input field should be disabled.

A disabled input field is unusable and un-clickable.

The value of a disabled input field will not be sent when submitting the form!

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" value="John" disabled><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname" value="Doe">  
</form>

## The size Attribute

The input size attribute specifies the visible width, in characters, of an input field.

The default value for size is 20.

Note: The size attribute works with the following input types: text, search, tel, url, email, and password.

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" size="50"><br>  
  <label for="pin">PIN:</label><br>  
  <input type="text" id="pin" name="pin" size="4">  
</form>

## The maxlength Attribute

The input maxlength attribute specifies the maximum number of characters allowed in an input field.

Note: When a maxlength is set, the input field will not accept more than the specified number of characters. However, this attribute does not provide any feedback. So, if you want to alert the user, you must write JavaScript code.

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" size="50"><br>  
  <label for="pin">PIN:</label><br>  
  <input type="text" id="pin" name="pin" maxlength="4" size="4">  
</form>

## The min and max Attributes

The input min and max attributes specify the minimum and maximum values for an input field.

The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

Tip: Use the max and min attributes together to create a range of legal values.

<form>  
  <label for="datemax">Enter a date before 1980-01-01:</label>  
  <input type="date" id="datemax" name="datemax" max="1979-12-31"><br><br>  
  
  <label for="datemin">Enter a date after 2000-01-01:</label>  
  <input type="date" id="datemin" name="datemin" min="2000-01-02"><br><br>  
  
  <label for="quantity">Quantity (between 1 and 5):</label>  
  <input type="number" id="quantity" name="quantity" min="1" max="5">  
</form>

## The multiple Attribute

The input multiple attribute specifies that the user is allowed to enter more than one value in an input field.

The multiple attribute works with the following input types: email, and file.

A file upload field that accepts multiple values:

<form>  
  <label for="files">Select files:</label>  
  <input type="file" id="files" name="files" multiple>  
</form>

## The pattern Attribute

The input pattern attribute specifies a regular expression that the input field's value is checked against, when the form is submitted.

The pattern attribute works with the following input types: text, date, search, url, tel, email, and password.

Tip: Use the global [title](https://www.w3schools.com/tags/att_global_title.asp) attribute to describe the pattern to help the user.

**Tip:** Learn more about [regular expressions](https://www.w3schools.com/js/js_regexp.asp) in our JavaScript tutorial.

An input field that can contain only three letters (no numbers or special characters):

<form>  
  <label for="country\_code">Country code:</label>  
  <input type="text" id="country\_code" name="country\_code"  
  pattern="[A-Za-z]{3}" title="Three letter country code">  
</form>

## The placeholder Attribute

The input placeholder attribute specifies a short hint that describes the expected value of an input field (a sample value or a short description of the expected format).

The short hint is displayed in the input field before the user enters a value.

The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

An input field with a placeholder text:

<form>  
  <label for="phone">Enter a phone number:</label>  
  <input type="tel" id="phone" name="phone"  
  placeholder="123-45-678"  
  pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">  
</form>

## The required Attribute

The input required attribute specifies that an input field must be filled out before submitting the form.

The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

A required input field:

<form>  
  <label for="username">Username:</label>  
  <input type="text" id="username" name="username" required>  
</form>

## The step Attribute

The input step attribute specifies the legal number intervals for an input field.

Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

**Tip:** This attribute can be used together with the max and min attributes to create a range of legal values.

The step attribute works with the following input types: number, range, date, datetime-local, month, time and week.

An input field with a specified legal number intervals:

<form>  
  <label for="points">Points:</label>  
  <input type="number" id="points" name="points" step="3">  
</form>

Note: Input restrictions are not foolproof, and JavaScript provides many ways to add illegal input. To safely restrict input, it must also be checked by the receiver (the server)!

## The autofocus Attribute

The input autofocus attribute specifies that an input field should automatically get focus when the page loads.

Let the "First name" input field automatically get focus when the page loads:

<form>  
  <label for="fname">First name:</label><br>  
  <input type="text" id="fname" name="fname" autofocus><br>  
  <label for="lname">Last name:</label><br>  
  <input type="text" id="lname" name="lname">  
</form>

## The height and width Attributes

The input height and width attributes specify the height and width of an <input type="image"> element.

Tip: Always specify both the height and width attributes for images. If height and width are set, the space required for the image is reserved when the page is loaded. Without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load).

Define an image as the submit button, with height and width attributes:

<form>  
  <label for="fname">First name:</label>  
  <input type="text" id="fname" name="fname"><br><br>  
  <label for="lname">Last name:</label>  
  <input type="text" id="lname" name="lname"><br><br>  
  <input type="image" src="img\_submit.gif" alt="Submit" width="48" height="48">  
</form>

## The list Attribute

The input list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

An <input> element with pre-defined values in a <datalist>:

<form>  
  <input list="browsers">  
  <datalist id="browsers">  
    <option value="Edge">  
    <option value="Firefox">  
    <option value="Chrome">  
    <option value="Opera">  
    <option value="Safari">  
  </datalist>  
</form>

## The autocomplete Attribute

The input autocomplete attribute specifies whether a form or an input field should have autocomplete on or off.

Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.

The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

An HTML form with autocomplete on, and off for one input field:

<form action="/action\_page.php" autocomplete="on">  
  <label for="fname">First name:</label>  
  <input type="text" id="fname" name="fname"><br><br>  
  <label for="lname">Last name:</label>  
  <input type="text" id="lname" name="lname"><br><br>  
  <label for="email">Email:</label>  
  <input type="email" id="email" name="email" autocomplete="off"><br><br>  
  <input type="submit" value="Submit">  
</form>

Tip: In some browsers you may need to activate an autocomplete function for this to work (Look under "Preferences" in the browser's menu).

# **HTTP Request Methods**

## What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

## HTTP Methods

* **GET**
* **POST**
* **PUT**
* **HEAD**
* **DELETE**
* **PATCH nb**
* **OPTIONS**
* **CONNECT**
* **TRACE**

The two most common HTTP methods are: GET and POST.

## The GET Method

GET is used to request data from a specified resource.

Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo\_form.php?name1=value1&name2=value2

**Some notes on GET requests:**

* GET requests can be cached
* GET requests remain in the browser history
* GET requests can be bookmarked
* GET requests should never be used when dealing with sensitive data
* GET requests have length restrictions
* GET requests are only used to request data (not modify)

## The POST Method

POST is used to send data to a server to create/update a resource.

The data sent to the server with POST is stored in the request body of the HTTP request:

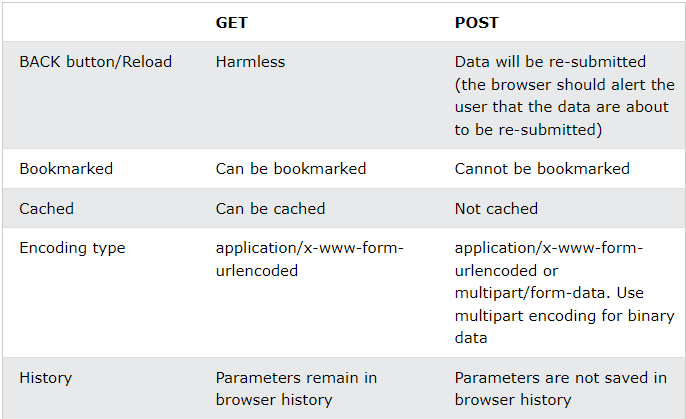
POST /test/demo\_form.php HTTP/1.1  
Host: w3schools.com  
  
name1=value1&name2=value2

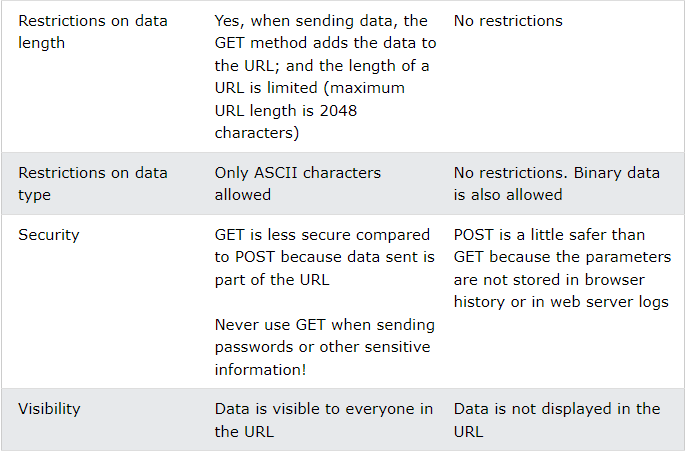
**Some notes on POST requests:**

* POST requests are never cached
* POST requests do not remain in the browser history
* POST requests cannot be bookmarked
* POST requests have no restrictions on data length

## Compare GET vs. POST

The following table compares the two HTTP methods: GET and POST.





## The PUT Method

PUT is used to send data to a server to create/update a resource.

The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

## The HEAD Method

HEAD is almost identical to GET, but without the response body.

In other words, if GET /users returns a list of users, then HEAD /users will make the same request but will not return the list of users.

A HEAD request is useful for checking what a GET request will return before actually making a GET request - a HEAD request can read the Content-Length header to check the size of the file, without actually downloading the file.

## The DELETE Method

The DELETE method deletes the specified resource.

## The PATCH Method

The PATCH method is used to apply partial modifications to a resource.

## The OPTIONS Method

The OPTIONS method describes the communication options for the target resource.

## The CONNECT Method

The CONNECT method is used to start a two-way communications (a tunnel) with the requested resource.

## The TRACE Method

The TRACE method is used to perform a message loop-back test that tests the path for the target resource (useful for debugging purposes).

# **HTML Div Element**

The <div> element is used as a container for other HTML elements.

## The <div> Element

The <div> element is by default a block element, meaning that it takes all available width, and comes with line breaks before and after.

A <div> element takes up all available width:

Lorem Ipsum <div>I am a div</div> dolor sit amet.

### **Result**

Lorem Ipsum

I am a div

dolor sit amet.

The <div> element has no required attributes, but style, class and id are common.

## <div> as a container

The <div> element is often used to group sections of a web page together.

### **Example**

A <div> element with HTML elements:

<div>  
  <h2>London</h2>  
  <p>London is the capital city of England.</p>  
  <p>London has over 13 million inhabitants.</p>  
</div>

### **Result**

## London

London is the capital city of England.

London has over 13 million inhabitants.

## Center align a <div> element

If you have a <div> element that is not 100% wide, and you want to center-align it, set the CSS margin property to auto.

### **Example**

<style>  
div {  
  width:300px;  
  margin:auto;  
}  
</style>

### **Result**

## London

London is the capital city of England.

London has over 13 million inhabitants.

## Multiple <div> elements

You can have many <div> containers on the same page.

### **Example**

<div>  
  <h2>London</h2>  
  <p>London is the capital city of England.</p>  
  <p>London has over 13 million inhabitants.</p>  
</div>  
  
<div>  
  <h2>Oslo</h2>  
  <p>Oslo is the capital city of Norway.</p>  
  <p>Oslo has over 600.000 inhabitants.</p>  
</div>  
  
<div>  
  <h2>Rome</h2>  
  <p>Rome is the capital city of Italy.</p>  
  <p>Rome has almost 3 million inhabitants.</p>  
</div>

### **Result**

## //’\London

London is the capital city of England.

London has over 13 million inhabitants.

## Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

## Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

## Aligning <div> elements side by side

When building web pages, you often want to have two or more <div> elements side by side, like this:

## London

London is the capital city of England.

London has over 13 million inhabitants.

## Oslo

/Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

## Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

There are different methods for aligning elements side by side, all include some CSS styling. We will look at the most common methods:

## Float

The CSS float property was not originally meant to align <div> elements side-by-side, but has been used for this purpose for many years.

The CSS float property is used for positioning and formatting content and allow elements float next to each other instead of on top of each other.

### **Example**

How to use float to align div elements side by side:

<style>  
.mycontainer {  
  width:100%;  
  overflow:auto;  
}  
.mycontainer div {  
  width:33%;  
  float:left;  
}  
</style>

### **Result**

## London

London is the capital city of England.

London has over 13 million inhabitants.

## Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

## Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

Learn more about float in our [CSS float tutorial](https://www.w3schools.com/css/css_float.asp).

## Inline-block

If you change the <div> element's display property from block to inline-block, the <div> elements will no longer add a line break before and after, and will be displayed side by side instead of on top of each other.

### **Example**

How to use display: inline-block to align div elements side by side:

<style>  
div {  
  width: 30%;  
  display: inline-block;  
}  
</style>

### **Result**

## London

London is the capital city of England.

London has over 13 million inhabitants.

## Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

## Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

## Flex

The CSS Flexbox Layout Module was introduced to make it easier to design flexible responsive layout structure without using float or positioning.

To make the CSS flex method work, surround the <div> elements with another <div> element and give it the status as a flex container.

### **Example**

How to use flex to align div elements side by side:

<style>  
.mycontainer {  
  display: flex;  
}  
.mycontainer > div {  
  width:33%;  
}  
</style>

### **Result**

## London

London is the capital city of England.

London has over 13 million inhabitants.

## Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

## Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

Learn more about flex in our [CSS flexbox tutorial](https://www.w3schools.com/css/css3_flexbox.asp).

## Grid

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Sounds almost the same as flex, but has the ability to define more than one row and position each row individually.

The CSS grid method requires that you surround the <div> elements with another <div> element and give the status as a grid container, and you must specify the width of each column.

### **Example**

How to use grid to align <div> elements side by side:

<style>  
.grid-container {  
  display: grid;  
  grid-template-columns: 33% 33% 33%;  
}  
</style>

### **Result**

## London

London is the capital city of England.

London has over 13 million inhabitants.

## Oslo

Oslo is the capital city of Norway.

Oslo has over 600.000 inhabitants.

## Rome

Rome is the capital city of Italy.

Rome has almost 3 million inhabitants.

Learn more about grid in our [CSS grid tutorial](https://www.w3schools.com/css/css_grid.asp).

## HTML Tags

[<div>](https://www.w3schools.com/tags/tag_div.asp) Defines a section in a document (block-level)

