# History of internet and Web Key Terminology

## Internet Protocol

The Internet Protocol (IP) is the method or protocol by which data is sent from one computer to another on the Internet. Each computer (known as a host) on the Internet has at least one IP address that uniquely identifies it from all other computers on the Internet.

Stands for "Internet Protocol." IP provides a standard set of rules for sending and receiving data over the Internet. It allows devices running on different platforms to communicate with each other as long as they are connected to the Internet.
In order for an Internet-connected host to be recognized by other devices, it must have an IP address.
The Internet Protocol also provides basic instructions for transferring packets between

devices.

## Client-Server Model

The client-server model describes how a server provides resources and services to one or more clients. Examples of servers include web servers, mail servers, and file servers. Each of these servers provide resources to client devices, such as desktop computers, laptops, tablets, and smartphones.

When a client requests a connection to a server, the server can either accept or reject the connection. If the connection is accepted, the server establishes and maintains a connection with the client over a specific protocol.

## DNS

Stands for "Domain Name System." Domain names serve as memorizable names for websites and other services on the Internet. However, computers access Internet devices by their IP addresses. DNS translates domain names into IP addresses, allowing you to access an Internet location by its domain name.

Thanks to DNS, you can visit a website by typing in the domain name rather than the IP address. For example, to visit the Tech Terms Computer Dictionary, you can simply type "techterms.com" in the address bar of your web browser rather than the IP address (67.43.14.98).

To understand how DNS works, you can think of it like the contacts app on your smartphone.

When you call a friend, you simply select his or her name from a list. The phone does not actually call the person by name, it calls the person's phone number. DNS works the same way by associating a unique IP address with each domain name.

## URL

Stands for "Uniform Resource Locator." A URL is the address of a specific webpage or file on the Internet. For example, the URL of the TechTerms website is "http://techterms.com." The address of this page is "http://techterms.com/definition/url" and includes the following elements:

http:// – the URL prefix, which specifies the protocol used to access the location

techterms.com – the server name or IP address of the server

/definition/url – the path to the directory or file

While all website URLs begin with "http," several other prefixes exist. Below is a list of various URL prefixes:

**http** : a webpage, website directory, or other file available over HTTP

**ftp** : a file or directory of files available to download from an FTP server

**telnet** : a Unix-based computer system that supports remote client

connections **mailto** : an email address (often used to redirect browsers to an

email client)

**file** : a file located on a local storage device (though not technically a URL because it does not refer to an Internet-based location)

You can manually enter a URL by typing it in the address bar of your web browser. For example, you might enter a website URL printed on a business card to visit the company's website. Most URLs, however appear automatically when you click on a link or open a bookmark. If the server name in the URL is not valid, your browser may display a "Server not found" error. If the path in the URL is incorrect, the server may respond with a 404 error.

NOTE: URLs use forward slashes to denote different directories and cannot contain spaces. Therefore, dashes and underscores are often used to separate words within a web address. If your browser produces an error when you visit a specific webpage, you can double-check the URL for typos or other errors. If you find an error, you can manually edit the URL and press Enter to see if it works.

## HTTP (Hypertext Transfer Protocol)

HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files, such as text, graphic images, sound, video, and other multimedia files, on the World Wide Web. As soon as a web user opens their web browser, the user is indirectly making use of HTTP. HTTP is an application protocol that runs on top of the TCP/IP suite of protocols (the foundation protocols for the Internet). The latest version of HTTP is HTTP/2, which was published in May 2015. It is an alternative to its predecessor, HTTP 1.1, but does not it make obsolete.

HTTP vs. HTTPS

HTTPS (HTTP over SSL or HTTP Secure) is the use of Secure Sockets Layer (SSL) or Transport Layer Security (TLS) as a sublayer under regular HTTP application layering. HTTPS encrypts and decrypts user HTTP page requests as well as the pages that are returned by the Web server. The use of HTTPS protects against eavesdropping and man-in-the-middle (MitM) attacks. HTTPS was developed by Netscape.

Migrating from HTTP to HTTPS is regarded as good for security.

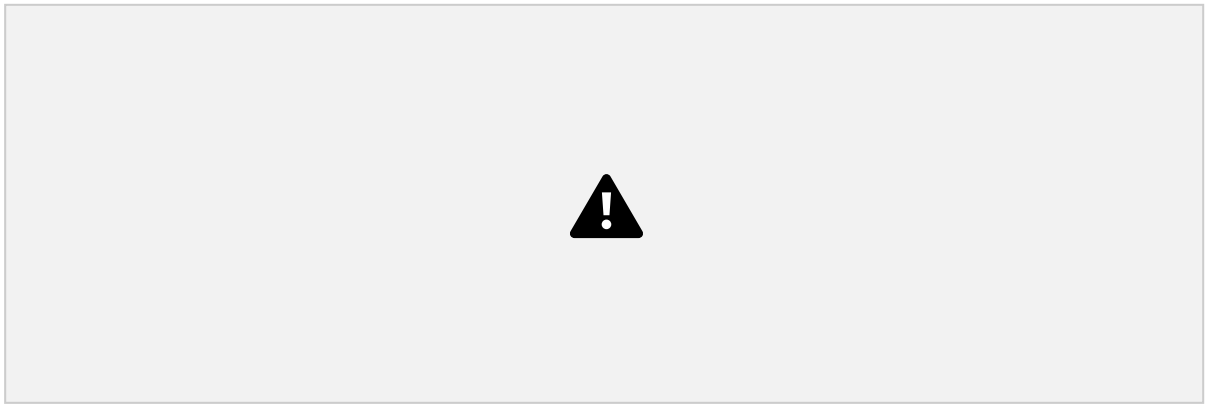
## web server

The term web server can refer to hardware or software, or both of them working together.

On the hardware side, a web server is a computer that stores web server software and a website's component files. (for example, HTML documents, images, CSS stylesheets, and JavaScript files) A web server connects to the Internet and supports physical data interchange with other devices connected to the web.

On the software side, a web server includes several parts that control how web users access hosted files. At a minimum, this is an HTTP server. An HTTP server is software that understands URLs (web addresses) and HTTP (the protocol your browser uses to view

WebPages). An HTTP server can be accessed through the domain names of the websites it stores, and it delivers the content of these hosted websites to the end user's device.



To publish a website, you need either a static or a dynamic web server.

A static web server, or stack, consists of a computer (hardware) with an HTTP server (software). We call it "static" because the server sends its hosted files as-is to your browser.

A dynamic web server consists of a static web server plus extra software, most commonly an application server and a database. We call it "dynamic" because the application server updates the hosted files before sending content to your browser via the HTTP server.

# Introduction to HTML5

HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web. HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, and Google Gears.

# New Features

HTML5 introduces a number of new elements and attributes that can help you in building modern websites. Here is a set of some of the most prominent features introduced in HTML5.

• New Semantic Elements : These are like <header>, <footer>, and <section>. • Forms 2.0 : Improvements to HTML web forms where new attributes have been introduced for <input> tag.

• Persistent Local Storage : To achieve without resorting to third-party plugins. • WebSocket : A next-generation bidirectional communication technology for web applications. • Server-Sent Events : HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).

• Canvas : This supports a two-dimensional drawing surface that you can program with JavaScript. • Audio & Video : You can embed audio or video on your WebPages without resorting to third party plugins.

• Geolocation : Now visitors can choose to share their physical location with your web application. • Microdata : This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.

## New structural elements of HTML5

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="UTF-8">
 <title></title>
```

```
</head>
<body>
<header> ... </header>
<nav> ... </nav>
<aside> ... </aside>
<article>
<section> ... </section>
</article>
<footer> ... </footer>
</body>
</html>
```

**The DOCTYPE**

DOCTYPEs in older versions of HTML were longer because the HTML language was SGML based and therefore required a reference to a DTD. HTML 5 authors would use simple syntax to specify DOCTYPE as follows:

```
<!DOCTYPE html>
```

**Character Encoding**

HTML 5 authors can use simple syntax to specify Character Encoding as follows –

```
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML, CSS, JavaScript">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

## Headers

The header element is used for introductory material that typically appears at the beginning of a web page or at the top of a section or article. There is no specified list of what a header must or should contain; anything that makes sense as the introduction to a page or section is acceptable. In the following example, the document header includes a logo image, the site title, and navigation.

```
<header>
<img src="/images/logo.png">
<hgroup>
<h1>Nuts about Web Fonts</h1>
<h2>News from the Web Typography Front</h2>
</hgroup>
<nav>
<ul>
```

```
 <li><a href="">Home</a></li>
 <li><a href="">Blog</a></li>
 <li><a href="">Shop</a></li>
 </ul>
 </nav>
</header>
```

When used in an individual article, the header might include the article title, author, and the publication date, as shown here:

```
<article>
 <header>
 <h1>More about WOFF</h1>
 <p>by Jennifer Robbins, <time datetime="11-11-2011"
pubdate>November 11, 2011</time></p>
 </header>
 <p>...article content starts here...</p>
</article>
```

## Footers

The footer element is used to indicate the type of information that typically comes at the end of a page or an article, such as its author, copyright information, related documents, or navigation. The footer element may apply to the entire document, or it could be associated with a particular section or article. If the footer is contained directly within the body element, either before or after all the other body content, then it applies to the entire page or application. If it is contained in a sectioning element (section, article, nav, or aside), it is parsed as the footer for just that section. Note that although it is called ―footer,‖ there is no requirement that it come last in the document or sectioning element. It could also appear at or near the beginning if it makes semantic sense. In this simple example we see the typical information listed at the bottom of an article or blog post marked up as a footer.

```
<footer>
 <p><small>Copyright &copy;2012 Jennifer Robbins.</small></p>
<nav>
 <ul>
 <li><a href="">Previous</a></li>
 <li><a href="">Next</a></li>
 </ul>
 </nav>
</footer>
```

## Navigation

The new nav element gives developers a semantic way to identify navigation for a site. Earlier in this chapter, we saw an unordered list that might be used as the top-level navigation for a font catalog site. Wrapping that list in a nav element makes its purpose explicitly clear.

```
<nav>
 <ul>
 <li><a href="">Serif</a></li>
 <li><a href="">Sans-serif</a></li>
 <li><a href="">Script</a></li>
 <li><a href="">Display</a></li>
 <li><a href="">Dingbats</a>/li>
 </ul>
</nav>
```

Not all lists of links should be wrapped in nav tags, however. The spec makes it clear that it should be used for links that provide primary navigation around a site or a lengthy section or article. The nav element may be especially helpful from an accessibility perspective. Once screen readers and other devices become HTML5-compatible, users can easily get to or skip navigation sections without a lot of hunting around.

## Sections and articles

Long documents are easier to use when they are divided into smaller parts. For example, books are divided into chapters, and newspapers have sections for local news, sports, comics, and so on. To divide long web documents into thematic sections, use the aptly named section element. Sections typically have a heading (inside the section element) and any other content that has a meaningful reason to be grouped together.

The section element has a broad range of uses, from dividing a whole page into major sections or identifying thematic sections within a single article. In the following example, a document with information about typography resources has been divided into two sections based on resource type.

```
<section>
 <h2>Typography Books</h2>
 <ul>
 <li>...</li>
 </ul>
 </section>
```

```
<section>
<h2>Online Tutorials</h2>
<p>These are the best tutorials on the web.</p>
<ul>
<li>...</li>
</ul>
</section>
```

Use the article element for self-contained works that could stand alone or be reused in a different context (such as syndication). It is useful for magazine or newspaper articles, blog posts, comments, or other items that could be extracted for external use. You can think of it as a specialized section element that answers the question ―Could this appear on another site and make sense?‖ with ―yes.‖

To make things interesting, a long article could be broken into a number of sections, as shown here:

```
<article>
<h1>Get to Know Helvetica</h1>
<section>
<h2>History of Helvetica</h2>
<p>...</p>
</section>
<section>
<h2>Helvetica Today</h2>
<p>...</p>
</section>
</article>
```

Conversely, a section in a web document might be comprised of a number of articles.

```
<section id="essays">
<article>
<h1>A Fresh Look at Futura</h1>
<p>...</p>
</article>
<article>
<h1>Getting Personal with Humanist</h1>
<p>...</p>
</article>
</section>
```

## Aside (sidebars)

The aside element identifies content that is related but tangential to the surrounding content. In

print, its equivalent is a sidebar, but they couldn't call the element sidebar, because putting something on the —side‖ is a presentational description, not semantic. Nonetheless, a sidebar is a good mental model for using the aside element. aside can be used for pull quotes, background information, lists of links, callouts, or anything else that might be associated with (but not critical to) a document.

In this example, an aside element is used for a list of links related to the main article.

```
<p>We now have a number of methods for using beautiful fonts on
web pages...</p>
<aside>
 <h2>Web Font Resources</h2>
 <ul>
 <li>
 <a href="http://typekit.com/">Typekit</a>
 </li>
 <li>
 <a href="http://www.google.com/webfonts">Google Fonts</a> </li>
 </ul>
</aside>
```

## Heading groups

HTML5 includes the hgroup element for identifying a stack of headings as a group.* Browsers that support hgroup know to count only the highest-ranked heading in the outline and ignore the rest.

```
<hgroup>
 <h1>Creating a Simple Page</h1>
 <h2>(HTML Overview)</h2>
</hgroup>
```

## Figures

The figure element is used for content that illustrates or supports some point in the text. A figure may contain an image, a video, a code snippet, text, or even a table—pretty much anything that can go in the flow of web content—and should be treated and referenced as a self-contained unit. That means if a figure is removed from its original placement in the main flow (to a sidebar or appendix, for example), both the figure and the main flow should continue to make sense.

Although it is possible to simply drop an image into text, wrapping it in figure tags makes its purpose explicitly clear. It also allows you to apply special styles to figures but not to other images on the page.

```
<figure>
```

```
 <img src="piechart.png" alt="chart showing fonts on mobile devices">
<figcaption>
 This is Image.
 </figcaption>
</figure>
```

A caption can be attached to the figure using the optional figcaption element above or below the figure content.

## Addresses

Last, and well, least, is the address element that is used to create an area for contact information for the author or maintainer of the document. It is generally placed at the end of the document or in a section or article within a document. An address would be right at home in a footer element.

It is important to note that the address element should not be used for any old address on a page, such as mailing addresses. It is intended specifically for author contact information (although that could potentially be a mailing address). Following is an example of its intended use. The —a  href‖ parts are the markup for links

```
<address>
 <a href="../authors/robbins/">Jennifer Robbins</a>,
 <a href="http://www.oreilly.com/">O'Reilly Media</a>
</address>
```

## Highlighted text

The new mark element indicates a word that may be considered especially relevant to the reader. One might use it to call out a search term in a page of results, to manually call attention to a passage of text, indicate the current page in a series. Some designers (and browsers) give marked text a light colored background as though it was marked with a highlighter marker.

```
<p> ... PART I. ADMINISTRATION OF THE GOVERNMENT. TITLE
IX.  TAXATION. CHAPTER 65C. MASS.
 <mark>ESTATE TAX</mark>.
 <mark>estate tax</mark>.
</p>
```

# Styling HTML with CSS

## Creating selector using property and value

CSS selectors are used to "find" (or select) the HTML elements you want

to style. We can divide CSS selectors into five categories:

- The CSS element selector
- The CSS id selector
- The CSS class selector
- The CSS Universal Selector
- The CSS grouping selector

### The CSS id selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
#idselector {
 text-align: center;
 color: red;
}
```

### The CSS class selector

The class selector selects HTML elements with a specific class attribute. To select elements with a specific class, write a period (.) character, followed by the class name.

```
.container_main{
 width: 960px;
 margin: 0 auto;
}
```

You can also specify that only specific HTML elements should be affected by a class.

```
p.center {
 text-align: center;
 color: red;
}
```

HTML elements can also refer to more than one class.

```
<p class="center large">This paragraph refers to two classes.</p>
```

## The CSS Universal selector

The universal selector (*) selects all HTML elements on the page.

```
* {
 text-align: center;
 color: blue;
}
```

## The CSS grouping selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
 text-align: center;
 color: red;
}
h2 {
 text-align: center;
 color: red;
}
```

```
p {
 text-align: center;
 color: red;
}
```

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

```
h1, h2, p {
 text-align: center;
 color: red;
}
```

## Creative a Responsive Design with CSS3 media query

### CSS2 Introduced Media Types

The @media rule, introduced in CSS2, made it possible to define different style rules for different media types.

Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.

Unfortunately these media types never got a lot of support by devices, other than the print media type.

### CSS3 Introduced Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Media query is a CSS technique introduced in CSS3.

It uses the @media rule to include a block of CSS properties only if a certain condition is true.

## Media Query Syntax

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions) {
 CSS-Code;
}
```

The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.

Unless you use the not or only operators, the media type is optional and the all type will be implied.

## CSS3 Media Types

| Value | Description |
|---|---|
| all | Used for all media type devices |
| print | Used for printers |
| screen | Used for computer screens, tablets, smart-phones etc. |
| speech | Used for screenreaders that "reads" the page out loud |

## Media Queries Simple Examples

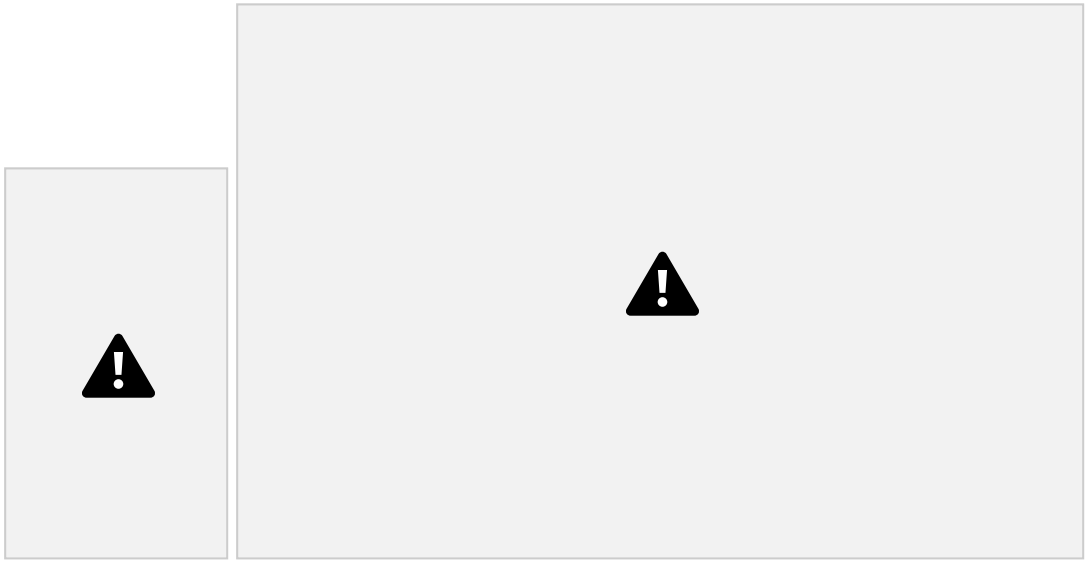
One way to use media queries is to have an alternate CSS section right inside your style sheet.

The following example changes the background-color to lightgreen if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the background-color will be pink):

```
@media screen and (min-width: 480px) {
 body {
 background-color: lightgreen;
 }
}
```

## Add a Breakpoint

We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.



```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {
 /* For mobile phones: */
 [class*="col-"] {
 width: 100%;
 }
}
```

## Always Design for Mobile First

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices).

This means that we must make some changes in our CSS.

Instead of changing styles when the width gets *smaller* than 768px, we should change the design when the width gets *larger* than 768px. This will make our design Mobile First:

```
/* For mobile phones: */
[class*="col-"] {
 width: 100%;
}

@media only screen and (min-width: 768px) {
 /* For desktop: */
 .col-1 {width: 8.33%;}
 .col-2 {width: 16.66%;}
 .col-3 {width: 25%;}
 .col-4 {width: 33.33%;}
 .col-5 {width: 41.66%;}
 .col-6 {width: 50%;}
 .col-7 {width: 58.33%;}
 .col-8 {width: 66.66%;}
.col-9 {width: 75%;}
 .col-10 {width: 83.33%;}
 .col-11 {width: 91.66%;}
 .col-12 {width: 100%;}
}
```

## Another Breakpoint

You can add as many breakpoints as you like.

We will also insert a breakpoint between tablets and mobile phones.

We do this by adding one more media query (at 600px), and a set of new classes for devices larger than 600px (but smaller than 768px):

Note that the two sets of classes are almost identical, the only difference is the name (col and col-s-):

```
/* For mobile phones: */
[class*="col-"] {
 width: 100%;
}
```

```
@media only screen and (min-width: 600px) {
 /* For tablets: */
 .col-s-1 {width: 8.33%;}
 .col-s-2 {width: 16.66%;}
 .col-s-3 {width: 25%;}
 .col-s-4 {width: 33.33%;}
 .col-s-5 {width: 41.66%;}
 .col-s-6 {width: 50%;}
.col-s-7 {width: 58.33%;}
 .col-s-8 {width: 66.66%;}
 .col-s-9 {width: 75%;}
 .col-s-10 {width: 83.33%;}
 .col-s-11 {width: 91.66%;}
 .col-s-12 {width: 100%;}
}
@media only screen and (min-width: 768px) {
 /* For desktop: */
 .col-1 {width: 8.33%;}
 .col-2 {width: 16.66%;}
 .col-3 {width: 25%;}
 .col-4 {width: 33.33%;}
 .col-5 {width: 41.66%;}
 .col-6 {width: 50%;}
 .col-7 {width: 58.33%;}
 .col-8 {width: 66.66%;}
 .col-9 {width: 75%;}
 .col-10 {width: 83.33%;}
 .col-11 {width: 91.66%;}
 .col-12 {width: 100%;}
}
```

## Typical Device Breakpoints

There are tons of screens and devices with different heights and widths, so it is hard to create an  exact breakpoint for each device. To keep things simple you could target five groups:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and
up) */ @media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}
```

```
/* Large devices (laptops/desktops, 992px and up) */

@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and
up) */ @media only screen and (min-width: 1200px) {...}
```

## Apply border to box element

The CSS border properties allow you to specify the style, width, and color of an element's border.

### CSS Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value • ridge - Defines a 3D ridged border. The effect depends on the border-color value • inset - Defines a 3D inset border. The effect depends on the border-color value • outset - Defines a 3D outset border. The effect depends on the border-color value • none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

## CSS Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre defined values: thin, medium, or thick:

```
p.one {
 border-style: solid;
 border-width: 5px;
}
p.two {
 border-style: solid;
 border-width: medium;
}
p.three {
 border-style: dotted;
 border-width: 2px;
}
 p.four {
 border-style: dotted;
 border-width: thick;
}
```

```
p.one {
 border-style: solid;
 border-width: 5px 20px; /* 5px top and bottom, 20px on the sides
*/ }

p.two {
 border-style: solid;
 border-width: 20px 5px; /* 20px top and bottom, 5px on the sides
*/ } p.three {
 border-style: solid;
 border
width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and
35px le ft */
}
```

## CSS Border Color

The border-color property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

Note:If border-color is not set, it inherits the color of the element.

```
p.one {
 border-style: solid;
 border-color: red;
}
p.two {
 border-style: solid;
 border-color: green;
}
p.three {
 border-style: dotted;
 border-color: blue;
}
```

```
p.one {
 border-style: solid;
 border
color: red green blue yellow; /* red top, green right, blue bottom and
yel low left */

}
```

```
p.one {
 border-style: solid;
 border-color: #ff0000; /* red */
}
```

```
p.one {
 border-style: solid;
 border-color: rgb(255, 0, 0); /* red */
}
```

```
p.one {
 border-style: solid;
 border-color: rgba(255, 0, 0,0.5); /* red */
}
```

### CSS Border - Shorthand Property

Like you saw in the previous page, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

- border-width
- border-style (required)
- border-color

```
p {
 border: 5px solid red;
}
```

```
p {
 border-bottom: 6px solid red;

background-color: lightgrey;
}
```

### CSS Rounded Borders

The border-radius property is used to add rounded borders to an element:

```
p {
 border: 2px solid red;
 border-radius: 5px;
}
```

## CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

### Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- length - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Note: Negative values are not allowed.

```
div {
 padding-top: 50px;
 padding-right: 30px;
 padding-bottom: 50px;
 padding-left: 80px;
}
```

## Padding - Shorthand Property

**padding: 25px 50px 75px 100px;**

- top padding is 25px
- right padding is 50px
- bottom padding is 75px
- left padding is 100px

**padding: 25px 50px 75px;**

- top padding is 25px
- right and left paddings are 50px
- bottom padding is 75px

**padding: 25px 50px;**

- top and bottom paddings are 25px
- right and left paddings are 50px

**padding: 25px;**

- all four paddings are 25px

## Padding and Element Width

The CSS width property specifies the width of the element's content area. The content area is the  portion inside the padding, border, and margin of an element (the box model).

So, if an element has a specified width, the padding added to that element will be added

to the  total width of the element. This is often an undesirable result.

Example

Here, the  element is given a width of 300px. However, the actual width of the   element will be 350px (300px + 25px of left padding + 25px of right padding):

```
div {
 width: 300px;
 padding: 25px;
}
```

To keep the width at 300px, no matter the amount of padding, you can use the box sizing property. This causes the element to maintain its width; if you increase the padding, the  available content space will decrease.

```
div {
 width: 300px;
 padding: 25px;
 box-sizing: border-box;
}
```

## CSS Margins

The CSS margin properties are used to create space around elements, outside of any defined  borders.

With CSS, you have full control over the margins. There are properties for setting the margin for  each side of an element (top, right, bottom, and left).

### Margin - Individual Sides

```
p {
 margin-top: 100px;
 margin-bottom: 100px;
 margin-right: 150px;
 margin-left: 80px;
}
```

### Margin - Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one

property. The margin property is a shorthand property for the following individual

margin properties:

**margin: 25px 50px 75px 100px;**

- top margin is 25px
- right margin is 50px
- bottom margin is 75px
- left margin is 100px

**margin: 25px 50px 75px;**

- top margin is 25px
- right and left margins are 50px
- bottom margin is 75px

**margin: 25px 50px;**

- top and bottom margins are 25px
- right and left margins are 50px

**margin: 25px;**

- all four margins are 25px

## Apply Position to box

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

The position property specifies the type of positioning method used for an

element. There are five different position values:

- relative
- fixed
- absolute

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

### position: relative;

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

```
<!DOCTYPE html>
```

```
<html>
 <head>
 <style>
 div.relative {
 position: relative;
 left: 30px;
 border: 3px solid #73AD21;
 }
 </style>
 </head>
 <body>
 <h2>position: relative;</h2>
 <p>An element with position: relative; is positioned relative to its
n ormal position:</p>
 <div class="relative">
 This div element has position: relative;
 </div>
 </body>
</html>
```

## position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been

located. Notice the fixed element in the lower-right corner of the page. Here is the

CSS that is used:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div.fixed {

position: fixed;
 bottom: 0;
 right: 0;
 width: 300px;
 border: 3px solid #73AD21;
 }
 </style>
 </head>
 <body>
```

```
<h2>position: fixed;</h2>
<p>An element with position: fixed; is positioned relative to the view port, which means it always stays in the same place even if the page is sc rolled:</p>
<div class="fixed">
This div element has position: fixed;
</div>
</body>
</html>
```

## position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

**Note:** A "positioned" element is one whose position is anything except

static. Here is a simple example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
position: relative;
width: 400px;
height: 200px;

border: 3px solid #73AD21;
}

div.absolute {
position: absolute;
top: 80px;
right: 0;
width: 200px;
height: 100px;
border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: absolute;</h2>
```

```
 <p>An element with position: absolute; is positioned relative to the
n earest positioned ancestor (instead of positioned relative to the
viewport , like fixed):</p>
 <div class="relative">This div element has position: relative; <div
class="absolute">This div element has position: absolute;</div>
</div>
 </body>
</html>
```

# CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

## CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

### The translate() Method

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the  element 50 pixels to the right, and 100 pixels down from its current position:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
 -ms-transform: translate(50px,100px); /* IE 9 */
```

```
transform: translate(50px,100px); /* Standard syntax */ }
</style>
</head>
<body>
<h1>The translate() Method</h1>
<p>The translate() method moves an element from its current
position:< /p>
<div>
This div element is moved 50 pixels to the right, and 100 pixels
down from its current position.
</div>
</body>
</html>
```

## The rotate() Method

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the  element clockwise with 20 degrees:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
width: 300px;
height: 100px;
background-color: yellow;
border: 1px solid black;
}
div#myDiv {
-ms-transform: rotate(20deg); /* IE 9 */
transform: rotate(20deg); /* Standard syntax */ }
</style>
</head>
<body>
<h1>The rotate() Method</h1>
<p>The rotate() method rotates an element clockwise or counter
clockwise.</p>
<div>
This a normal div element.
</div>
<div id="myDiv">
This div element is rotated clockwise 20 degrees.
</div>
</body>
</html>
```

Using negative values will rotate the element counter-clockwise.

## The scale() Method

The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the  element to be two times of its original width, and three times of its original height:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
margin: 150px;
width: 200px;
height: 100px;
background-color: yellow;
border: 1px solid black;
-ms-transform: scale(2,3); /* IE 9 */
transform: scale(2,3); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The scale() Method</h1>
<p>The scale() method increases or decreases the size of an
element.</ p>
<div>
This div element is two times of its original width, and three times
o f its original height.
</div>
</body>
</html>
```

## The scaleX() Method

The scaleX() method increases or decreases the width of an element.

The following example increases the  element to be two times of its original width:

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
div {
margin: 150px;
width: 200px;
height: 100px;
background-color: yellow;
border: 1px solid black;
-ms-transform: scaleX(2); /* IE 9 */
transform: scaleX(2); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The scaleX() Method</h1>
<p>The scaleX() method increases or decreases the width of an
element. </p>
<div>
This div element is two times of its original
width. </div>
</body>
</html>
```

## The scaleY() Method

The scaleY() method increases or decreases the height of an element.

The following example increases the  element to be three times of its original height:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
margin: 150px;
width: 200px;
height: 100px;

background-color: yellow;
border: 1px solid black;
-ms-transform: scaleY(3); /* IE 9 */
transform: scaleY(3); /* Standard syntax */
}
</style>
</head>
<body>
<h1>The scaleY() Method</h1>
<p>The scaleY() method increases or decreases the height of an
element .</p>
```

```
 <div>
 This div element is three times of its original
height.  </div>
 </body>
</html>
```

## The skewX() Method

The skewX() method skews an element along the X-axis by the given angle. The following example skews the  element 20 degrees along the X-axis:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
 }

 div#myDiv {
 -ms-transform: skewX(20deg); /* IE 9 */
 transform: skewX(20deg); /* Standard syntax */
 }
 </style>

</head>
 <body>
 <h1>The skewX() Method</h1>
 <p>The skewX() method skews an element along the X
axis by the given angle.</p>
 <div>
 This a normal div element.
 </div>
 <div id="myDiv">
 This div element is skewed 20 degrees along the X-axis.
</div>
 </body>
</html>
```

## The skewY() Method

The skewY() method skews an element along the Y-axis by the given

angle. The following example skews the  element 20 degrees

along the Y-axis:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
 }

 div#myDiv {
 -ms-transform: skewY(20deg); /* IE 9 */
transform: skewY(20deg); /* Standard syntax */  }
 </style>
 </head>
 <body>

<h1>The skewY() Method</h1>
 <p>The skewY() method skews an element along the Y
axis by the given angle.</p>
 <div>
 This a normal div element.
 </div>
 <div id="myDiv">
 This div element is skewed 20 degrees along the Y-axis.
</div>
 </body>
</html>
```

## The skew() Method

The skew() method skews an element along the X and Y-axis by the given angles.

The following example skews the  element 20 degrees along the X-axis, and 10 degrees  along the Y-axis:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 300px;
 height: 100px;
 background-color: yellow;
 border: 1px solid black;
 }

 div#myDiv {
 -ms-transform: skew(20deg,10deg); /* IE 9 */
 transform: skew(20deg,10deg); /* Standard syntax */  }
 </style>
 </head>
 <body>
 <h1>The skew() Method</h1>

<p>The skew() method skews an element into a given angle.</p>
<div>
 This a normal div element.
 </div>
 <div id="myDiv">
 This div element is skewed 20 degrees along the X
axis, and 10 degrees along the Y-axis.
 </div>
 </body>
</html>
```

If the second parameter is not specified, it has a zero value. So, the following example skews the  element 20 degrees along the X-axis:

## The matrix() Method

The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow:
matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

```
<!DOCTYPE html>
<html>
 <head>
```

```
<style>
div {
width: 300px;
height: 100px;
background-color: yellow;
border: 1px solid black;
}
div#myDiv1 {
-ms-transform: matrix(1, -0.3, 0, 1, 0, 0); /* IE 9 */
transform: matrix(1, -0.3, 0, 1, 0, 0); /* Standard syntax */ }
div#myDiv2 {
-ms-transform: matrix(1, 0, 0.5, 1, 150, 0); /* IE 9 */

transform: matrix(1, 0, 0.5, 1, 150, 0); /* Standard syntax */ }
</style>
</head>
<body>
<h1>The matrix() Method</h1>
<p>The matrix() method combines all the 2D transform methods into
one. </p>
<div>
This a normal div element.
</div>
<div id="myDiv1">
Using the matrix() method.
</div>
<div id="myDiv2">
Another use of the matrix() method.
</div>
</body>
</html>
```

# CSS **Transitions**

## CSS Transitions

CSS transitions allows you to change property values smoothly, over a given

duration. you will learn about the following properties:

- transition
- transition-delay
- transition-duration

### How to Use CSS Transitions?

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Note:If the duration part is not specified, the transition will have no effect, because the default value is 0.

The following example shows a 100px * 100px red  element. The  element has also specified a transition effect for the width property, with a duration of 2 seconds:

```
div {
 width: 100px;
 height: 100px;
 background: red;
 transition: width 2s;
 }
```

The transition effect will start when the specified CSS property (width) changes value.

Now, let us specify a new value for the width property when a user mouses over the  element:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 100px;
 height: 100px;
 background: red;
 transition: width 2s;
 }
 div:hover {
 width: 300px;
 }
 </style>
 </head>
 <body>
 <h1>The transition Property</h1>
 <p>Hover over the div element below, to see the transition
effect:</p>  <div></div>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
and earlier versions.</p>

</body>
</html>
```

## Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 100px;
 height: 100px;
 background: red;
 transition: width 2s, height 4s;
 }
 div:hover {
 width: 300px;
 height: 300px;
 }
 </style>
 </head>
 <body>
 <h1>The transition Property</h1>
 <p>Hover over the div element below, to see the transition
effect:</p>  <div></div>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
and  earlier versions.</p>
 </body>
</html>
```

## Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition

effect. The following example has a 1 second delay before starting:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 100px;
 height: 100px;
 background: red;
```

```
 transition: width 3s;
 transition-delay: 1s;
 }
 div:hover {
 width: 300px;
 }
 </style>
 </head>
 <body>
 <h1>The transition-delay Property</h1>
 <p>Hover over the div element below, to see the transition
effect:</p>  <div></div>
 <p><b>Note:</b> The transition effect has a 1 second delay before
star ting.</p>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
and earlier versions.</p>
 </body>
</html>
```

## Transition + Transformation

The following example adds a transition effect to the transformation:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 100px;
 height: 100px;
 background: red;

transition: width 2s, height 2s, transform 2s;
 }
 div:hover {
 width: 300px;
 height: 300px;
 transform: rotate(180deg);
 }
 </style>
 </head>
 <body>
 <h1>Transition + Transform</h1>
 <p>Hover over the div element below:</p>
 <div></div>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
```

```
and earlier versions.</p>
 </body>
</html>
```

## creating animation with CSS

CSS allows animation of HTML elements without using JavaScript or

Flash! In this chapter you will learn about the following properties:

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

### What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times you

want. To use CSS animation, you must first specify some keyframes for

the animation. Keyframes hold what styles the element will have at

certain times.

### The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the  element. The animation will last for 4 seconds, and it will gradually change the background-color of the  element from "red" to "yellow":

```
<!DOCTYPE html>
<html>
 <head>
```

```
<style>
div {
width: 100px;
height: 100px;
background-color: red;
animation-name: example;
animation-duration: 4s;
}
@keyframes example {
from {background-color: red;}
to {background-color: yellow;}
}
</style>
</head>
<body>
<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>
<div></div>
<p><b>Note:</b> When an animation is finished, it changes back to its original style.</p>
</body>

</html>
```

Note: The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the  element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
width: 100px;
height: 100px;
background-color: red;
animation-name: example;
animation-duration: 4s;
}
```

```
@keyframes example {
 0% {background-color: red;}
 25% {background-color: yellow;}
 50% {background-color: blue;}
 100% {background-color: green;}
 }
 </style>
 </head>
 <body>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
and earlier versions.</p>
 <div></div>
 </body>
</html>
```

The following example will change both the background-color and the position of the  element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 100px;
 height: 100px;
 background-color: red;
 position: relative;
 animation-name: example;
 animation-duration: 4s;
 }
 @keyframes example {
 0% {background-color:red; left:0px; top:0px;} 25%
{background-color:yellow; left:200px; top:0px;} 50%
{background-color:blue; left:200px; top:200px;} 75%
{background-color:green; left:0px; top:200px;} 100%
{background-color:red; left:0px; top:0px;} }
 </style>
 </head>
 <body>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
and earlier versions.</p>
 <div></div>
 </body>
</html>
```

## Delay an Animation

The animation-delay property specifies a delay for the start of an animation. The following example has a 2 seconds delay before starting the animation:

```
<!DOCTYPE html>
<html>

<head>
 <style>
 div {
 width: 100px;
 height: 100px;
 background-color: red;
 position: relative;
 animation-name: example;
 animation-duration: 4s;
 animation-delay: 2s;
 }
 @keyframes example {
 0% {background-color:red; left:0px; top:0px;}
 25% {background-color:yellow; left:200px; top:0px;}
50% {background-color:blue; left:200px; top:200px;}  75%
{background-color:green; left:0px; top:200px;}  100%
{background-color:red; left:0px; top:0px;}
 }
 </style>
 </head>
 <body>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
and earlier versions.</p>
 <div></div>
 </body>
</html>
```

Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for *N* seconds.

In the following example, the animation will start as if it had already been playing for 2 seconds:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 100px;
```

```
 height: 100px;
 background-color: red;

position: relative;
 animation-name: example;
 animation-duration: 4s;
 animation-delay: -2s;
 }
 @keyframes example {
 0% {background-color:red; left:0px; top:0px;}  25%
{background-color:yellow; left:200px; top:0px;}  50%
{background-color:blue; left:200px; top:200px;}  75%
{background-color:green; left:0px; top:200px;}  100%
{background-color:red; left:0px; top:0px;}  }
 </style>
 </head>
 <body>
 <p>Using negative values: Here, the animation will start as if it
had  already been playing for 2 seconds:</p>
 <div></div>
 <p><b>Note:</b> This example does not work in Internet Explorer 9
and  earlier versions.</p>
 </body>
</html>
```

## Set How Many Times an Animation Should Run

The animation-iteration-count property specifies the number of times an animation should run. The following example will run the animation 3 times before it stops:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 div {
 width: 100px;
 height: 100px;
 background-color: red;
 position: relative;
 animation-name: example;

animation-duration: 4s;
 animation-iteration-count: 3;
 }
 @keyframes example {
 0% {background-color:red; left:0px; top:0px;}  25%
```

```
{background-color:yellow; left:200px; top:0px;} 50%
{background-color:blue; left:200px; top:200px;} 75%
{background-color:green; left:0px; top:200px;} 100%
{background-color:red; left:0px; top:0px;} }
</style>
</head>
<body>
<p><b>Note:</b> This example does not work in Internet Explorer 9
and earlier versions.</p>
<div></div>
</body>
</html>
```

The following example uses the value "infinite" to make the animation continue forever:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
width: 100px;
height: 100px;
background-color: red;
position: relative;
animation-name: example;
animation-duration: 4s;
animation-iteration-count: infinite;
}
@keyframes example {
0% {background-color:red; left:0px; top:0px;} 25%
{background-color:yellow; left:200px; top:0px;} 50%
{background-color:blue; left:200px; top:200px;} 75%
{background-color:green; left:0px; top:200px;} 100%
{background-color:red; left:0px; top:0px;} }

</style>
</head>
<body>
<p><b>Note:</b> This example does not work in Internet Explorer 9
and earlier versions.</p>
<div></div>
</body>
</html>
```

# HTML5 & CSS Web Forms

## Styling Input Fields

Use the **width** property to determine the width of the input field:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 input {
 width: 100%;
 }
 </style>
 </head>
 <body>
 <p>A full-width input field:</p>
 <form>
 <label for="fname">First Name</label>
 <input type="text" id="fname" name="fname">
 </form>
 </body>
</html>
```

The example above applies to all <input> elements. If you only want to style a specific input type, you can use attribute selectors:

- input[type=text] - will only select text fields
- input[type=password] - will only select password fields
- input[type=number] - will only select number fields

## Padded Inputs

Use the padding property to add space inside the text field.

**Tip:** When you have many inputs after each other, you might also want to add some margin, to add more space outside of them:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 input[type=text] {
 width: 100%;
 padding: 12px 20px;
 margin: 8px 0;
 box-sizing: border-box;
 }
 </style>
```

```
</head>
<body>
<p>Padded text fields:</p>
<form>
<label for="fname">First Name</label>
<input type="text" id="fname" name="fname">
<label for="lname">Last Name</label>
<input type="text" id="lname" name="lname">
</form>
</body>
</html>
```

## Bordered Inputs

Use the **border** property to change the border size and color, and use the **border-radius** property to add rounded corners:

```
<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
width: 100%;
padding: 12px 20px;

margin: 8px 0;
box-sizing: border-box;
border: 2px solid red;
border-radius: 4px;
}
</style>
</head>
<body>
<p>Text fields with borders:</p>
<form>
<label for="fname">First Name</label>
<input type="text" id="fname" name="fname">
<label for="lname">Last Name</label> <input
type="text" id="lname" name="lname"> </form>
</body>
</html>
```

If you only want a bottom border, use the **border-bottom** property:

```
<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
width: 100%;
```

```
 padding: 12px 20px;
 margin: 8px 0;
 box-sizing: border-box;
 border: none;
 border-bottom: 2px solid red;
 }
 </style>
 </head>
 <body>
 <p>Text fields with only a bottom
border:</p>  <form>
 <label for="fname">First Name</label>
<input type="text" id="fname" name="fname">
<label for="lname">Last Name</label>  <input
type="text" id="lname" name="lname">

</form>
 </body>
</html>
```

## Colored Inputs

Use the background-color property to add a background color to the input, and the color property  to change the text color:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 input[type=text] {
 width: 100%;
 padding: 12px 20px;
 margin: 8px 0;
 box-sizing: border-box;
 border: none;
 background-color: #3CBC8D;
 color: white;
 }
 </style>
 </head>
 <body>
 <p>Colored text fields:</p>
 <form>
 <label for="fname">First Name</label>
 <input type="text" id="fname" name="fname" value="John">
<label for="lname">Last Name</label>
 <input type="text" id="lname" name="lname" value="Doe">
</form>
 </body>
```

```
</html>
```

## Focused Inputs

By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input.

Use the :focus selector to do something with the input field when it gets focus:

```
<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
width: 100%;
padding: 12px 20px;
margin: 8px 0;
box-sizing: border-box;
border: 1px solid #555;
outline: none;
}
input[type=text]:focus {
background-color: lightblue;
}
</style>
</head>
<body>
<p>In this example, we use the :focus selector to add a background color to the text field when it gets focused (clicked on):</p> <form>
<label for="fname">First Name</label>
<input type="text" id="fname" name="fname" value="John"> <label for="lname">Last Name</label>
<input type="text" id="lname" name="lname" value="Doe"> </form>
</body>
</html>
<!DOCTYPE html>
<html>
<head>
<style>
input[type=text] {
width: 100%;
padding: 12px 20px;
margin: 8px 0;
box-sizing: border-box;
border: 3px solid #ccc;
-webkit-transition: 0.5s;
```

```
transition: 0.5s;
 outline: none;
 }
 input[type=text]:focus {
 border: 3px solid #555;
 }
 </style>
 </head>
 <body>
 <p>In this example, we use the :focus selector to add a black border
c olor to the text field when it gets focused (clicked on):</p>
<p>Note that we have added the CSS transition property to animate the
border color (takes 0.5 seconds to change the color on focus).</p>
<form>
 <label for="fname">First Name</label>
 <input type="text" id="fname" name="fname" value="John">
<label for="lname">Last Name</label>
 <input type="text" id="lname" name="lname" value="Doe">
</form>
 </body>
</html>
```

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 input[type=text] {
 width: 100%;
 padding: 12px 20px;
 margin: 8px 0;
 box-sizing: border-box;
 border: 3px solid #ccc;
 -webkit-transition: 0.5s;
 transition: 0.5s;
 outline: none;
 }
 input[type=text]:focus {
 border: 3px solid #555;

}
 </style>
 </head>
 <body>
 <p>In this example, we use the :focus selector to add a black border
```

```
c olor to the text field when it gets focused (clicked on):</p>
<p>Note that we have added the CSS transition property to animate the
border color (takes 0.5 seconds to change the color on focus).</p>
<form>
 <label for="fname">First Name</label>
 <input type="text" id="fname" name="fname" value="John">
<label for="lname">Last Name</label>
 <input type="text" id="lname" name="lname" value="Doe">
</form>
 </body>
</html>
```

## Input with icon/image

If you want an icon inside the input, use the **background-image** property and position it with the **background-position** property. Also notice that we add a large left padding to reserve the space of the icon:

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 input[type=text] {
 width: 100%;
 box-sizing: border-box;
 border: 2px solid #ccc;
 border-radius: 4px;
 font-size: 16px;
 background-color: white;
 background-image: url('searchicon.png');
 background-position: 10px 10px;
 background-repeat: no-repeat;
 padding: 12px 20px 12px 40px;
 }
 </style>

</head>
 <body>
 <p>Input with icon:</p>
 <form>
 <input type="text" name="search" placeholder="Search..">
</form>
 </body>
</html>
```

## Animated Search Input

In this example we use the CSS **transition** property to animate the width of the search input when it gets focus.

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 input[type=text] {
 width: 130px;
 box-sizing: border-box;
 border: 2px solid #ccc;
 border-radius: 4px;
 font-size: 16px;
 background-color: white;
 background-image: url('searchicon.png');
 background-position: 10px 10px;
 background-repeat: no-repeat;
 padding: 12px 20px 12px 40px;
 transition: width 0.4s ease-in-out;
 }
 input[type=text]:focus {
 width: 100%;
 }
 </style>
 </head>
 <body>
 <p>Animated search input:</p>
 <form>
 <input type="text" name="search" placeholder="Search..">
</form>

</body>
</html>
```

## Styling Textareas

**Tip:** Use the **resize** property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 textarea {
 width: 100%;
 height: 150px;
 padding: 12px 20px;
 box-sizing: border-box;
 border: 2px solid #ccc;
 border-radius: 4px;
 background-color: #f8f8f8;
```

```
 font-size: 16px;
 resize: none;
 }
 </style>
 </head>
 <body>
 <p><strong>Tip:</strong> Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):</p>  <form>
 <textarea>Some text...</textarea>
 </form>
 </body>
</html>
```

## Styling Select Menus

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 select {
 width: 100%;

padding: 16px 20px;
 border: none;
 border-radius: 4px;
 background-color: #f1f1f1;
 }
 </style>
 </head>
 <body>
 <p>A styled select menu.</p>
 <form>
 <select id="country" name="country">
 <option value="au">Australia</option>
 <option value="ca">Canada</option>
 <option value="usa">USA</option>
 </select>
 </form>
 </body>
</html>
```

## Styling Input Buttons

```
<!DOCTYPE html>
<html>
 <head>
 <style>
```

```
 input[type=button], input[type=submit], input[type=reset] {
background-color: #4CAF50;
 border: none;
 color: white;
 padding: 16px 32px;
 text-decoration: none;
 margin: 4px 2px;
 cursor: pointer;
 }
 </style>
 </head>
 <body>
 <p>Styled input buttons.</p>
 <input type="button" value="Button">
 <input type="reset" value="Reset">

<input type="submit" value="Submit">
 </body>
</html>
```

## Responsive Form

Resize the browser window to see the effect. When the screen is less than 600px wide, make the  two columns stack on top of each other instead of next to each other.

```
<!DOCTYPE html>
<html>
 <head>
 <style>
 * {
 box-sizing: border-box;
 }
 input[type=text], select, textarea {
 width: 100%;
 padding: 12px;
 border: 1px solid #ccc;
 border-radius: 4px;
 resize: vertical;
 }
 label {
 padding: 12px 12px 12px 0;
 display: inline-block;
 }
 input[type=submit] {
 background-color: #4CAF50;
 color: white;
 padding: 12px 20px;
 border: none;
 border-radius: 4px;
```

```
 cursor: pointer;
 float: right;
 }
 input[type=submit]:hover {
 background-color: #45a049;
 }
 .container {
 border-radius: 5px;

background-color: #f2f2f2;
 padding: 20px;
 }
 .col-25 {
 float: left;
 width: 25%;
 margin-top: 6px;
 }
 .col-75 {
 float: left;
 width: 75%;
 margin-top: 6px;
 }
 /* Clear floats after the columns */
 .row:after {
 content: "";
 display: table;
 clear: both;
 }
 /* Responsive layout -
  when the screen is less than 600px wide, make the two columns stack
on to p of each other instead of next to each other */
 @media screen and (max-width: 600px) {
 .col-25, .col-75, input[type=submit] {
 width: 100%;
 margin-top: 0;
 }
 }
 </style>
 </head>
 <body>

 <h2>Responsive Form</h2>
 <p>Resize the browser window to see the effect. When the screen is
les s than 600px wide, make the two columns stack on top of each other
instead of next to each other.</p>
 <div class="container">
 <form action="/action_page.php">
```

```
 <div class="row">

<div class="col-25">
 <label for="fname">First Name</label>
 </div>
 <div class="col-75">
 <input type="text" id="fname" name="firstname" placeholder="Your
  name..">
 </div>
 </div>
 <div class="row">
 <div class="col-25">
 <label for="lname">Last Name</label>
 </div>
 <div class="col-75">
 <input type="text" id="lname" name="lastname" placeholder="Your last
name..">
 </div>
 </div>
 <div class="row">
 <div class="col-25">
 <label for="country">Country</label>
 </div>
 <div class="col-75">
 <select id="country" name="country">
 <option value="australia">Australia</option>
<option value="canada">Canada</option>
 <option value="usa">USA</option>
 </select>
 </div>
 </div>
 <div class="row">
 <div class="col-25">
 <label for="subject">Subject</label>
 </div>
 <div class="col-75">
 <textarea id="subject" name="subject" placeholder="Write somethi
ng.." style="height:200px"></textarea>
 </div>
 </div>
 <div class="row">

<input type="submit" value="Submit">
</div>
 </form>
 </div>
 </body>
```

```
</html>
```

# Concept of Bootstrap Layout and Media object

## What is Bootstrap ?

- Bootstrap is a free front-end framework for faster and easier web development.
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins.
- Bootstrap also gives you the ability to easily create responsive designs.

## What is Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

## Where to Get Bootstrap?

There are two ways to start using Bootstrap on your own web site.

You can:

- Download Bootstrap from getbootstrap.com
- Include Bootstrap from a CDN

## Downloading Bootstrap

If you want to download and host Bootstrap yourself, go to getbootstrap.com, and follow the instructions there.

## Bootstrap CDN

If you don't want to download and host Bootstrap yourself, you can include it from a CDN (Content Delivery Network).

MaxCDN provides CDN support for Bootstrap's CSS and JavaScript. You must also include jQuery:



```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/
3.4.1/css/bootstrap.min.css" integrity="sha384-
HSMxcRTRxnN+Bdg0JdbxYKrThecOKuH5zCYotlSAcp1+c8xmyTe9GYg1l9a69psu"
crossori gin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/
3.4.1/css/bootstrap-theme.min.css" integrity="sha384-
6pzBo3FDv/PJ8r2KRkGHifhEocL+1X2rVCTTkUfGk7/0pbek5mMa1upzvWbrUbOZ"
crossori gin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/js/bootstr
ap.min.js" integrity="sha384-
aJ21OjlMXNL5UyIl/XNwTMqvzeRMZH2w8c5cRVpzpU8Y5bApTppSuUkhZXN0VxHd"
crossori gin="anonymous"></script>
```

## Basic template

Start with this basic HTML template, or modify these examples. We hope you'll customize our templates and examples, adapting them to suit your needs.

```
<!doctype html>
<html lang="en">
 <head>
 <meta charset="utf-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge"> <meta
name="viewport" content="width=device-width, initial-scale=1"> <!--
 The above 3 meta tags *must* come first in the head; any other head
conte nt must come *after* these tags -->
 <title>Bootstrap 101 Template</title>

 <!-- Bootstrap -->
```

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootst
rap/3.4.1/css/bootstrap.min.css" integrity="sha384-
HSMxcRTRxnN+Bdg0JdbxYKrThecOKuH5zCYotlSAcp1+c8xmyTe9GYg1l9a69psu"
crossori gin="anonymous">

 <!--
  HTML5 shim and Respond.js for IE8 support of HTML5 elements and
media que ries -->
 <!--
  WARNING: Respond.js doesn't work if you view the page via file://
-->  <!--[if lt IE 9]>
 <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js
"></script>
 <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></
script>
 <![endif]-->
 </head>
 <body>
 <h1>Hello, world!</h1>

 <!-- jQuery (necessary for Bootstrap's JavaScript plugins)
-->  <script src="https://code.jquery.com/jquery
1.12.4.min.js" integrity="sha384-
nvAa0+6Qg9clwYCGGPpDQLVpLNn0fRaROjHqs13t4Ggj3Ez50XnGQqc/r8MhnRDZ"
crossori gin="anonymous"></script>
 <!--
  Include all compiled plugins (below), or include individual files as
need ed -->
 <script
src="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/js/boo
tstrap.min.js" integrity="sha384-
aJ21OjlMXNL5UyIl/XNwTMqvzeRMZH2w8c5cRVpzpU8Y5bApTppSuUkhZXN0VxHd"
crossori gin="anonymous"></script>
 </body>
</html>
```

## HTML5 doctype

Bootstrap makes use of certain HTML elements and CSS properties that require the use of the  HTML5 doctype. Include it at the beginning of all your projects.

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

```
  ...
</html>
```

## Mobile first

With Bootstrap 2, we added optional mobile friendly styles for key aspects of the framework. With Bootstrap 3, we've rewritten the project to be mobile friendly from the start. Instead of adding on optional mobile styles, they're baked right into the core. In fact, **Bootstrap is mobile first**. Mobile first styles can be found throughout the entire library instead of in separate files.

To ensure proper rendering and touch zooming, **add the viewport meta tag** to your <head>. `<meta name="viewport" content="width=device-width, initial-scale=1">`

You can disable zooming capabilities on mobile devices by adding user-scalable=no to the viewport meta tag. This disables zooming, meaning users are only able to scroll, and results in your site feeling a bit more like a native application. Overall, we don't recommend this on every site, so use caution!

```
<meta name="viewport" content="width=device-width,
initial scale=1, maximum-scale=1, user-scalable=no">
```

## Containers

Bootstrap requires a containing element to wrap site contents and house our grid system. You may choose one of two containers to use in your projects. Note that, due to padding and more, neither container is nestable.

Use .container for a responsive fixed width container.

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

Use .container-fluid for a full width container, spanning the entire width of your viewport.

```
<div class="container-fluid">
  ...
</div>
```

## Grid system

Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

Grid systems are used for creating page layouts through a series of rows and columns that house your content. Here's how the Bootstrap grid system works:

- Rows must be placed within a .container (fixed-width) or .container-fluid (full-width)

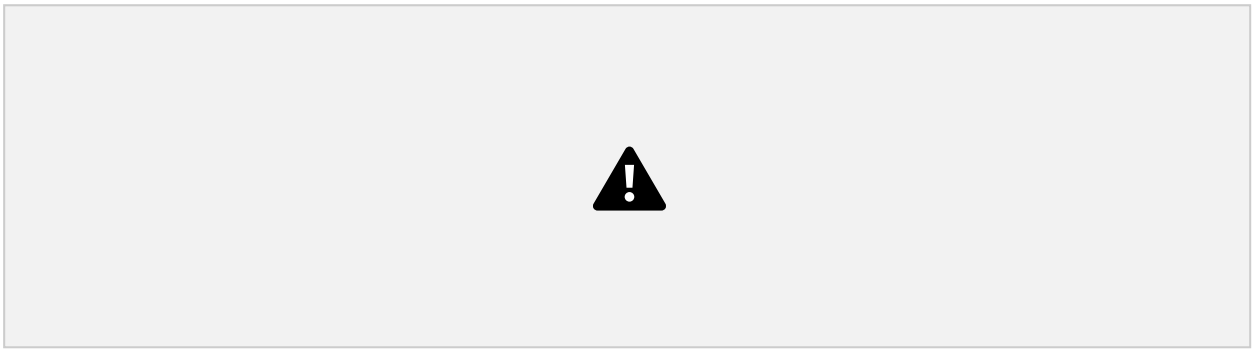
for proper alignment and padding.

- Use rows to create horizontal groups of columns.
- Content should be placed within columns, and only columns may be immediate children of rows.
- Predefined grid classes like .row and .col-xs-4 are available for quickly making grid layouts. Less mixins can also be used for more semantic layouts.
- Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and last column via negative margin on .rows.
- The negative margin is why the examples below are outdented. It's so that content within grid columns is lined up with non-grid content.
- Grid columns are created by specifying the number of twelve available columns you wish to span. For example, three equal columns would use three .col-xs-4.
- If more than 12 columns are placed within a single row, each group of extra columns will, as one unit, wrap onto a new line.
- Grid classes apply to devices with screen widths greater than or equal to the breakpoint sizes, and override grid classes targeted at smaller devices. Therefore, e.g. applying any .col-md-* class to an element will not only affect its styling on medium devices but also on large devices if a .col-lg-* class is not present.

## Grid options

| | **Extra small devices Phones (<768px)** | **Small devices Tablets (≥768px)** | **Medium devices Desktops (≥992px)** | **Large devices Desktops (≥1200px)** |
|---|---|---|---|---|
| **Grid behavior** | Horizontal at all times | Collapsed to start, horizontal above breakpoints | | |
| **Container width** | None (auto) | 750px | 970px | 1170px |
| **Class prefix** | .col-xs- | .col-sm- | .col-md- | .col-lg- |
| **# of columns** | 12 | | | |
| **Column width** | Auto | ~62px | ~81px | ~97px |
| **Gutter width** | 30px (15px on each side of a column) | | | |
| **Nestable** | Yes | | | |
| **Offsets** | Yes | | | |
| **Column ordering** | Yes | | | |

## Example: Stacked-to-horizontal



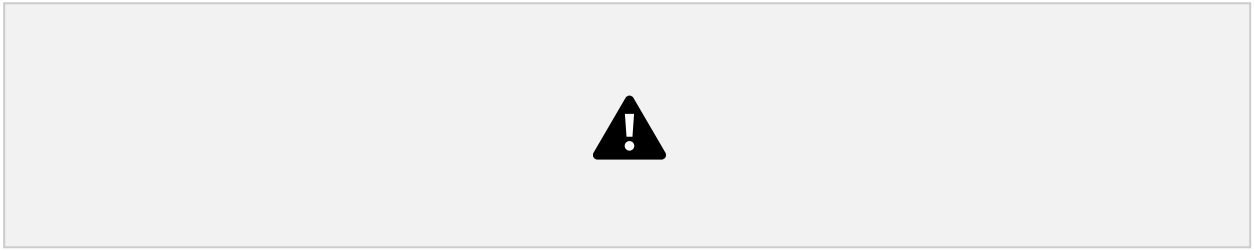
```
<div class="row">
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
 <div class="col-md-1">.col-md-1</div>
</div>
<div class="row">
 <div class="col-md-8">.col-md-8</div>
 <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
 <div class="col-md-4">.col-md-4</div>
 <div class="col-md-4">.col-md-4</div>
 <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
 <div class="col-md-6">.col-md-6</div>
 <div class="col-md-6">.col-md-6</div>
</div>
```

## Example: Fluid container

Turn any fixed-width grid layout into a full-width layout by changing your outermost .container to .container-fluid.

```
<div class="container-fluid">
 <div class="row">
 ...
 </div>
</div>
```

## Example: Mobile and desktop



```
<!-- Stack the columns on mobile by making one full
width and the other half-width -->
<div class="row">
 <div class="col-xs-12 col-md-8">.col-xs-12
.col-md-8</div>  <div class="col-xs-6
col-md-4">.col-xs-6 .col-md-4</div> </div>

<!--
  Columns start at 50% wide on mobile and bump up to 33.3% wide on
desktop  -->
<div class="row">
     <div      class="col-xs-6      col-md-4">.col-xs-6
.col-md-4</div>        <div       class="col-xs-6
col-md-4">.col-xs-6      .col-md-4</div>      <div
class="col-xs-6   col-md-4">.col-xs-6   .col-md-4</div>
</div>

<!-- Columns are always 50% wide, on mobile and desktop
-->  <div class="row">

<div class="col-xs-6">.col-xs-6</div>
 <div class="col-xs-6">.col-xs-6</div>
</div>
```

## Example: Mobile, tablet, desktop

```
<div class="row">
 <div class="col-xs-12 col-sm-6 col-md-8">.col-xs-12 .col-sm-6
.col-md 8</div>
```

```
 <div class="col-xs-6 col-md-4">.col-xs-6
.col-md-4</div> </div>
<div class="row">
     <div      class="col-xs-6      col-sm-4">.col-xs-6
.col-sm-4</div>        <div       class="col-xs-6
col-sm-4">.col-xs-6 .col-sm-4</div>  <!--
  Optional: clear the XS cols if their content doesn't match in height
-->  <div class="clearfix visible-xs-block"></div>
 <div class="col-xs-6 col-sm-4">.col-xs-6
.col-sm-4</div> </div>
```

## Offsetting columns

Move columns to the right using .col-md-offset-* classes. These classes increase the left margin  of a column by * columns. For example, .col-md-offset-4 moves .col-md-4 over four columns.

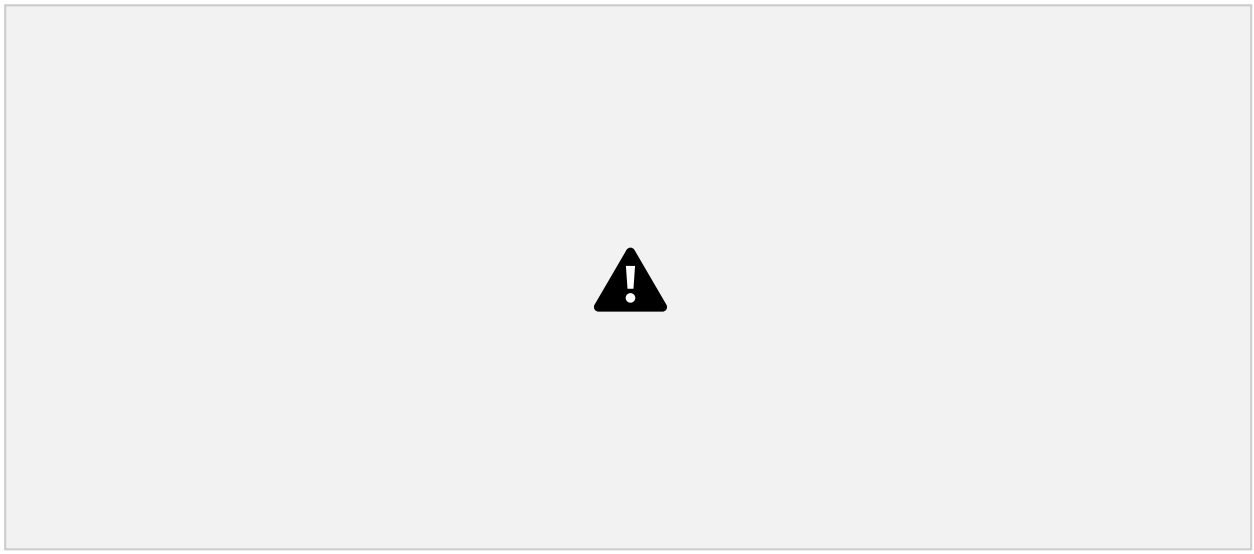
```
<div class="row">
 <div class="col-md-4">.col-md-4</div>
 <div class="col-md-4 col-md-offset-4">.col-md-4
.col-md-offset-4</div>  </div>

<div class="row">
        <div        class="col-md-3        col-md-offset-3">.col-md-3
.col-md-offset-3</div>          <div          class="col-md-3
col-md-offset-3">.col-md-3 .col-md-offset-3</div>  </div>
 <div class="row">
 <div class="col-md-6 col-md-offset-3">.col-md-6
.col-md-offset-3</div>  </div>
```

## Typography

## Headings

All HTML headings, <h1> through <h6>, are available. .h1 through .h6 classes are also available, for when you want to match the font styling of a heading but still want your text to be  displayed inline.



```
<h1>h1. Bootstrap heading</h1>
<h2>h2. Bootstrap heading</h2>
<h3>h3. Bootstrap heading</h3>
<h4>h4. Bootstrap heading</h4>
<h5>h5. Bootstrap heading</h5>
<h6>h6. Bootstrap heading</h6>
```

## Body copy

Bootstrap's global default font-size is **14px**, with a line-height of **1.428**. This is applied to the <body> and all paragraphs. In addition, <p> (paragraphs) receive a bottom margin of half their computed line-height (10px by default).

```
<p>...</p>
```

## Marked text

For highlighting a run of text due to its relevance in another context, use the <mark> tag.

```
You can use the mark tag to <mark>highlight</mark>
text.
```

## Deleted text

For indicating blocks of text that have been deleted use the <del> tag.

```
<del>This line of text is meant to be treated as deleted
text.</del>
```

## Strikethrough text

For indicating blocks of text that are no longer relevant use the <s> tag.

```
<s>This line of text is meant to be treated as no longer
accurate.</s>
```

## Inserted text

For indicating additions to the document use the <ins> tag.

```
<ins>This line of text is meant to be treated as an addition to the
document.</ins>
```

## Alignment classes

Easily realign text to components with text alignment classes.

```
<p class="text-left">Left aligned text.</p>
<p class="text-center">Center aligned text.</p>
<p class="text-right">Right aligned text.</p>
<p class="text-justify">Justified text.</p>
<p class="text-nowrap">No wrap text.</p>
```

## Transformation classes

Transform text in components with text capitalization classes.

```
<p class="text-lowercase">Lowercased text.</p>
```

```
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">Capitalized text.</p>
```

## Addresses

Present contact information for the nearest ancestor or the entire body of work.
Preserve  formatting by ending all lines with <br>.

```
<address>
 <strong>Twitter, Inc.</strong><br>
 1355 Market Street, Suite 900<br>
 San Francisco, CA 94103<br>
 <abbr title="Phone">P:</abbr> (123) 456-7890
 </address>
```

```
 <address>
 <strong>Full Name</strong><br>
 <a href="mailto:#">first.last@example.com</a>
 </address>
```

## Blockquotes

Wrap <blockquote> around any HTML as the quote. For straight quotes, we recommend a <p>.

```
<blockquote>
 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer
posuere erat a ante.</p>
</blockquote>
```

## Lists

A list of items in which the order does *not* explicitly matter.

```
<ul>
 <li>...</li>
</ul>
```

## Tables

For basic styling—light padding and only horizontal dividers—add the base class .table to any <table>. It may seem super redundant, but given the widespread use of tables for other plugins like calendars and date pickers, we've opted to isolate our custom table styles.

```
<table class="table">
  ...
</table>
```

## Striped rows

Use .table-striped to add zebra-striping to any table row within the <tbody>.

```
<table class="table table-striped">
  ...
</table>
```

## Bordered table

Add .table-bordered for borders on all sides of the table and cells.

```
<table class="table table-bordered">
  ...
</table>
```

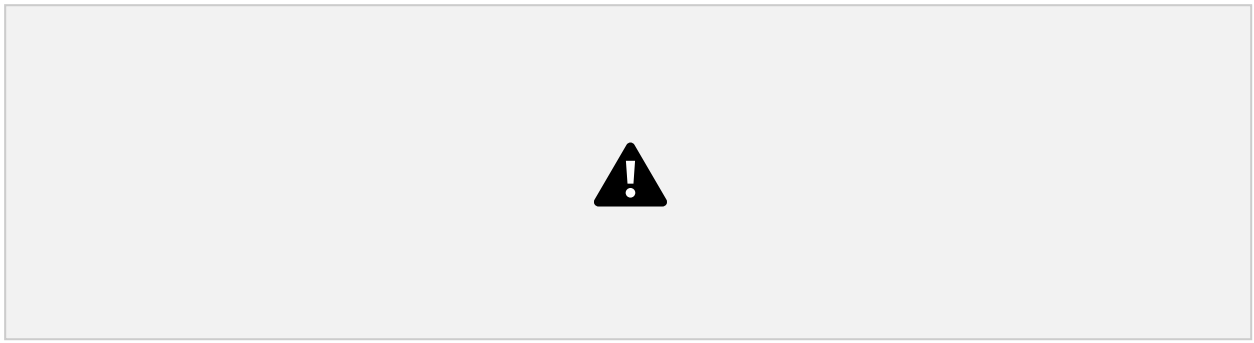
## Hover rows

Add .table-hover to enable a hover state on table rows within a <tbody>.

```
<table class="table table-hover">
  ...
</table>
```

## Contextual classes

Use contextual classes to color table rows or individual cells.



```
<!-- On rows -->
<tr class="active">...</tr>
<tr class="success">...</tr>
```

```
<tr class="warning">...</tr>
<tr class="danger">...</tr>
<tr class="info">...</tr>
<!-- On cells (`td` or `th`) -->
<tr>
 <td class="active">...</td>
 <td class="success">...</td>
 <td class="warning">...</td>
 <td class="danger">...</td>
 <td class="info">...</td>
</tr>
```

## Responsive tables

Create responsive tables by wrapping any .table in .table-responsive to make them scroll horizontally on small devices (under 768px). When viewing on anything larger than 768px wide,  you will not see any difference in these tables.

```
<div class="table-responsive">
 <table class="table">
 ...
 </table>
</div>
```

## Images

## Responsive images

Images in Bootstrap 3 can be made responsive-friendly via the addition of the .img responsive class. This applies max-width: 100%;, height: auto; and display: block; to the image  so that it scales nicely to the parent element.

To center images which use the .img-responsive class, use .center-block instead of .text-center.

```
<img src="..." class="img-responsive" alt="Responsive
image">
```

## Image shapes

Add classes to an <img> element to easily style images in any project.

```
<img src="..." alt="..." class="img-rounded">
<img src="..." alt="..." class="img-circle">
<img src="..." alt="..." class="img-thumbnail">
```

# Advanced Bootstrap Components like: Badges, Buttons, Cards.

## alert messages.

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Wrap any text and an optional dismiss button in .alert and one of the four contextual classes (e.g., .alert-success) for basic alert messages.

```
<div class="alert alert-success" role="alert">...</div>
<div class="alert alert-info" role="alert">...</div>
<div class="alert alert-warning" role="alert">...</div>
<div class="alert alert-danger" role="alert">...</div>
```

### Links in alerts

Use the .alert-link utility class to quickly provide matching colored links within any alert.

```
<div class="alert alert-success" role="alert">
 <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-info" role="alert">
 <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-warning" role="alert">
 <a href="#" class="alert-link">...</a>
</div>
<div class="alert alert-danger" role="alert">
 <a href="#" class="alert-link">...</a>
</div>
```

## Badges

Easily highlight new or unread items by adding a  to links, Bootstrap navs, and more.

```
<a href="#">Inbox <span class="badge">42</span></a>
```

```
<button class="btn btn-primary" type="button">
 Messages <span class="badge">4</span>
</button>
```

### Adapts to active nav states

Built-in styles are included for placing badges in active states in pill navigations.

```
<ul class="nav nav-pills" role="tablist">
 <li role="presentation" class="active">
 <a href="#">Home <span class="badge">42</span>
 </a>
 </li>
 <li role="presentation"><a href="#">Profile</a></li>
 <li role="presentation"><a href="#">Messages <span class="badge">3</span
></a></li>
</ul>
```

## Buttons

### Button tags

Use the button classes on an <a>, <button>, or <input> element.

```
<a class="btn btn-default" href="#" role="button">Link</a>
<button class="btn btn-default"
type="submit">Button</button> <input class="btn btn-default"
type="button" value="Input"> <input class="btn btn-default"
type="submit" value="Submit">
```

### Options

Use any of the available button classes to quickly create a styled button.

```
<!-- Standard button -->
<button type="button" class="btn
btn-default">Default</button> <!--
  Provides extra visual weight and identifies the primary action in a set
o f buttons -->
<button type="button" class="btn
btn-primary">Primary</button> <!-- Indicates a successful or
positive action -->

<button type="button" class="btn
btn-success">Success</button> <!-- Contextual button for
informational alert messages --> <button type="button"
class="btn btn-info">Info</button>
<!-- Indicates caution should be taken with this action -->
```

```
<button type="button" class="btn
btn-warning">Warning</button> <!-- Indicates a dangerous or
potentially negative action --> <button type="button"
class="btn btn-danger">Danger</button> <!--
 Deemphasize a button by making it look like a link while maintaining
butt on behavior -->
<button type="button" class="btn btn-link">Link</button>
```

## Sizes

Fancy larger or smaller buttons? Add .btn-lg, .btn-sm, or .btn-xs for additional sizes.

```
<p>
 <button type="button" class="btn btn-primary btn
lg">Large button</button>
 <button type="button" class="btn btn-default btn
lg">Large button</button>
</p>
<p>
 <button type="button" class="btn btn-primary">Default button</button>
<button type="button" class="btn btn-default">Default button</button>
</p>
<p>
 <button type="button" class="btn btn-primary btn
sm">Small button</button>
 <button type="button" class="btn btn-default btn
sm">Small button</button>
</p>
<p>
 <button type="button" class="btn btn-primary btn
xs">Extra small button</button>
 <button type="button" class="btn btn-default btn
xs">Extra small button</button>
</p>
```

## Active state

Buttons will appear pressed (with a darker background, darker border, and inset shadow) when active. For <button> elements, this is done via :active. For <a> elements, it's done with .active. However, you may use .active on <button>s (and include the aria-pressed="true" attribute) should you need to replicate the active state programmatically.

```
<button type="button" class="btn btn-primary btn
lg active">Primary button</button>
<button type="button" class="btn btn-default btn
lg active">Button</button>
```

## Disabled state

Make buttons look unclickable by fading them back with opacity.

```
<button type="button" class="btn btn-lg btn
```

```
primary" disabled="disabled">Primary button</button>
<button type="button" class="btn btn-default btn
lg" disabled="disabled">Button</button>
```

### Thumbnails

Extend Bootstrap's grid system with the thumbnail component to easily display grids of images, videos, text, and more.

```
<div class="row">
 <div class="col-sm-6 col-md-4">
 <div class="thumbnail">
 <img src="..." alt="...">
 <div class="caption">
 <h3>Thumbnail label</h3>
 <p>...</p>
 <p><a href="#" class="btn btn-primary" role="button">Button</a>  <a
href="#" class="btn btn-default" role="button">Button</a>  </p>
 </div>
 </div>
 </div>
</div>
```

## Advanced Bootstrap Components like: Carousal, Form Controls,  Navigation bar, Progress bar

### Carousel

A slideshow component for cycling through elements, like a carousel. **Nested carousels are not supported.**

```
<div id="carousel-example-generic" class="carousel slide" data
ride="carousel">
 <!-- Indicators -->
 <ol class="carousel-indicators">
 <li data-target="#carousel-example-generic" data-slide
to="0" class="active"></li>
  <li  data-target="#carousel-example-generic"  data-slide-to="1"></li>
<li   data-target="#carousel-example-generic"   data-slide-to="2"></li>
</ol>

 <!-- Wrapper for slides -->
 <div class="carousel-inner" role="listbox">
```

```
 <div class="item active">
 <img src="..." alt="...">
 <div class="carousel-caption">
 ...
 </div>
 </div>
 <div class="item">
 <img src="..." alt="...">
 <div class="carousel-caption">
 ...
 </div>
 </div>
 ...
 </div>

 <!-- Controls -->


 <a class="left carousel-control" href="#carousel-example
generic" role="button" data-slide="prev">
 <span class="glyphicon glyphicon-chevron-left" aria
hidden="true"></span>
 <span class="sr-only">Previous</span>
 </a>
 <a class="right carousel-control" href="#carousel-example
generic" role="button" data-slide="next">
 <span class="glyphicon glyphicon-chevron-right" aria
hidden="true"></span>
 <span class="sr-only">Next</span>
 </a>
</div>
```

## Forms

Individual form controls automatically receive some global styling. All textual <input>, <textarea>, and <select> elements with .form-control are set to width: 100%; by default. Wrap labels and controls in .form-group for optimum spacing.

```
<form>
 <div class="form-group">
 <label for="exampleInputEmail1">Email address</label>
 <input type="email" class="form
control" id="exampleInputEmail1" placeholder="Email">
 </div>
 <div class="form-group">
 <label for="exampleInputPassword1">Password</label>
```

```
 <input type="password" class="form
 control" id="exampleInputPassword1"
placeholder="Password">  </div>
 <div class="form-group">
 <label for="exampleInputFile">File input</label>
 <input type="file" id="exampleInputFile">
 <p class="help-block">Example block-level help text here.</p>
</div>
 <div class="checkbox">
 <label>

 <input type="checkbox"> Check me out
 </label>
 </div>
 <button type="submit" class="btn btn-default">Submit</button>
</form>
```

## Navbar

Navbars are responsive meta components that serve as navigation headers for your application or site. They begin collapsed (and are toggleable) in mobile views and become horizontal as the available viewport width increases.

```
<nav class="navbar navbar-default">
 <div class="container-fluid">
 <!-- Brand and toggle get grouped for better mobile display -->
<div class="navbar-header">
 <button type="button" class="navbar-toggle collapsed" data
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria
expanded="false">
 <span class="sr-only">Toggle navigation</span>
 <span class="icon-bar"></span>
 <span class="icon-bar"></span>
 <span class="icon-bar"></span>
 </button>
 <a class="navbar-brand" href="#">Brand</a>
 </div>
 <!-- Collect the nav links, forms, and other content for toggling -->
<div class="collapse navbar-collapse" id="bs-example-navbar-collapse 1">
 <ul class="nav navbar-nav navbar-right">
 <li><a href="#">Link</a></li>
 </ul>
 </div><!-- /.navbar-collapse -->
 </div><!-- /.container-fluid -->
</nav>
```

### Brand image

Replace the navbar brand with your own image by swapping the text for an <img>. Since the .navbar-brand has its own padding and height, you may need to override some CSS depending on your image.

```
<nav class="navbar navbar-default">
 <div class="container-fluid">
 <div class="navbar-header">
 <a class="navbar-brand" href="#">
 <img alt="Brand" src="...">
 </a>
 </div>
 </div>
</nav>
```

## Progress bars

Provide up-to-date feedback on the progress of a workflow or action with simple yet flexible progress bars.

```
<div class="progress">
 <div class="progress-bar" role="progressbar" aria-valuenow="60" aria
valuemin="0" aria-valuemax="100" style="width: 60%;">
 <span class="sr-only">60% Complete</span>
 </div>
</div>
```

### With label

Remove the  with .sr-only class from within the progress bar to show a visible percentage.

```
<div class="progress">
 <div class="progress-bar" role="progressbar" aria-valuenow="60" aria
valuemin="0" aria-valuemax="100" style="width: 60%;">
 60%
 </div>
</div>
```

### Contextual alternatives

Progress bars use some of the same button and alert classes for consistent styles.

```
<div class="progress">
 <div class="progress-bar progress-bar-success" role="progressbar" aria
valuenow="40" aria-valuemin="0" aria-valuemax="100" style="width: 40%">
<span class="sr-only">40% Complete (success)</span>
 </div>
```

```
</div>
<div class="progress">
 <div class="progress-bar progress-bar-info" role="progressbar" aria
valuenow="20" aria-valuemin="0" aria-valuemax="100" style="width:
20%">  <span class="sr-only">20% Complete</span>
 </div>
</div>
<div class="progress">
 <div class="progress-bar progress-bar-warning" role="progressbar" aria
valuenow="60" aria-valuemin="0" aria-valuemax="100" style="width: 60%">
<span class="sr-only">60% Complete (warning)</span>
 </div>
</div>
<div class="progress">
 <div class="progress-bar progress-bar-danger" role="progressbar" aria
valuenow="80" aria-valuemin="0" aria-valuemax="100" style="width: 80%">
<span class="sr-only">80% Complete (danger)</span>
 </div>
</div>
```

## Animated

Add .active to .progress-bar-striped to animate the stripes right to left. Not available in IE9 and below.

```
<div class="progress">
 <div class="progress-bar progress-bar
striped active" role="progressbar" aria-valuenow="45" aria
valuemin="0" aria-valuemax="100" style="width: 45%">
 <span class="sr-only">45% Complete</span>
 </div>
</div>
```