|  |
| --- |
|  |
| HTML & CSS |
| Nagarjun Sanji |
|  |
|  |
|  |

|  |
| --- |
|  |

[1. HTM-5 - Hypertext Markup Language 2](#_Toc510346673)

[a. Elements & Semantic Elements 2](#_Toc510346674)

[b. Elements 2](#_Toc510346675)

[d. Iframe 4](#_Toc510346676)

[e. Forms 5](#_Toc510346677)

[f. Web Storage 5](#_Toc510346678)

[g. Assignment 6](#_Toc510346679)

[2. CSS-3 - Cascading Style Sheets 7](#_Toc510346680)

[a. 2D Transform 7](#_Toc510346681)

[b. 3D Transform 8](#_Toc510346682)

[c. Web Fonts 8](#_Toc510346683)

[d. Transitions 9](#_Toc510346684)

[e. Shadows 10](#_Toc510346685)

[f. Animation 11](#_Toc510346686)

[a. Keyframes 11](#_Toc510346687)

[b. Pseudo-classes 11](#_Toc510346688)

[c. Pseudo-elements 12](#_Toc510346689)

[d. Display properties 13](#_Toc510346690)

[e. Positions 13](#_Toc510346691)

[f. Tooltips 15](#_Toc510346692)

[g. Text Effects 16](#_Toc510346693)

# HTM-5 - Hypertext Markup Language

## Elements & Semantic Elements

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of non-semantic elements: <div> and <span> - Tells nothing about its content.

Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.

New Semantic Elements in HTML5

<article>

<aside>

<details>

<figcaption>

<figure>

<footer>

<header>

<main>

<mark>

<nav>

<section>

<summary>

<time>

## Elements

**Tag Description**

<!--...--> Defines a comment

<!DOCTYPE> Defines the document type

<a> Defines a hyperlink

<body> Defines the document's body

<br> Defines a single line break

<button> Defines a clickable button

<h1>-<h6> Defines HTML headings

<img> Defines an image

<input> Defines an input control

<label> Defines a label for an <input> element

<table> Defines a table

<tbody> Groups the body content in a table

<td> Defines a cell in a table

1. Attributes

**Attribute** **Description**

class Specifies one or more classnames for an element (refers to a class in a style sheet)

data-\* Used to store custom data private to the page or application

hidden Specifies that an element is not yet, or is no longer, relevant

id Specifies a unique id for an element

style Specifies an inline CSS style for an element

title Specifies extra information about an element

**Keyboard Events**

**Attribute** **Value** **Description**

onkeydown script Fires when a user is pressing a key

onkeypress script Fires when a user presses a key

onkeyup script Fires when a user releases a key

**Mouse Events**

**Attribute** **Value** **Description**

onclick script Fires on a mouse click on the element

ondblclick script Fires on a mouse double-click on the element

onmousedown script Fires when a mouse button is pressed down on an

element

onmousemove script Fires when the mouse pointer is moving

while it is over an element

onmouseout script Fires when the mouse pointer moves out of

an element

onmouseover script Fires when the mouse pointer moves over an

element

onmouseup script Fires when a mouse button is released over

an element

onmousewheel script Deprecated. Use the onwheel attribute

instead

onwheel script Fires when the mouse wheel rolls up or down

over an element

## Iframe

An iframe is used to display a web page within a web page.

Iframe Syntax

An HTML iframe is defined with the <iframe> tag:

<iframe src="*URL*"></iframe>

The src attribute specifies the URL (web address) of the inline frame page.

Iframe - Set Height and Width

Use the height and width attributes to specify the size of the iframe.

The attribute values are specified in pixels by default, but they can also be in percent (like "80%").

<iframe src="demo.htm" height="200" width="300"></iframe>

Iframe - Target for a Link

An iframe can be used as the target frame for a link.

The target attribute of the link must refer to the name attribute of the iframe:

<iframe src="demo.htm" name="iframe"></iframe>  
  
<p><a href="https://www.example.com" target="iframe">example.com</a></p>

## Forms

The <form> Element

The HTML <form> element defines a form that is used to collect user input:

Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

The <input> element is the most important form element.

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

**Type** **Description**

<input type="text"> Defines a one-line text input field

<input type="radio"> Defines a radio button (for selecting one of

many choices)

<input type="submit"> Defines a submit button (for submitting the

form)

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

<input name="firstname" type="text">

The <select> element defines a drop-down list:

<select name="cars">  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="fiat">Fiat</option>  
  <option value="audi">Audi</option>  
</select>

The <option> elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

## Web Storage

With web storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

**The localStorage Object**

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

// Store  
localStorage.setItem("lastname", "Smith");  
// Retrieve  
document.getElementById("result").innerHTML = localStorage.getItem("lastname");

**The sessionStorage Object**

The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

The following example counts the number of times a user has clicked a button, in the current session:

if (sessionStorage.clickcount) {  
    sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;  
} else {  
    sessionStorage.clickcount = 1;  
}  
document.getElementById("result").innerHTML = "You have clicked the button " +  
sessionStorage.clickcount + " time(s) in this session.";

## Assignment

Creating a web page with header, footer, left navigation, body contains article, section and uses localstorage and session storage to save user details on click of save button through user form.

# CSS-3 - Cascading Style Sheets

CSS stands for Cascading Style Sheets

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

CSS saves a lot of work. It can control the layout of multiple web pages all at once

External stylesheets are stored in CSS files

body {  
    background-color: lightblue;  
}  
  
h1 {  
    color: white;  
    text-align: center;  
}  
  
p {  
    font-family: verdana;  
    font-size: 20px;  
}

## 2D Transform

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

A transformation is an effect that lets an element change shape, size and position.

**2D Transform Methods**

**Function** **Description**

matrix(n,n,n,n,n,n) Defines a 2D transformation, using a matrix of

six values

translate(x,y) Defines a 2D translation, moving the element

along the X- and the Y-axis

translateX(n) Defines a 2D translation, moving the element

along the X-axis

translateY(n) Defines a 2D translation, moving the element

along the Y-axis

scale(x,y) Defines a 2D scale transformation, changing the

elements width and height

scaleX(n) Defines a 2D scale transformation, changing the

element's width

scaleY(n) Defines a 2D scale transformation, changing the

element's height

rotate(angle) Defines a 2D rotation, the angle is specified in the parameter

## 3D Transform

CSS allows you to format your elements using 3D transformations.

**Function** **Description**

translate3d(x,y,z) Defines a 3D translation

translateX(x) Defines a 3D translation, using only the value for the X-axis

translateY(y) Defines a 3D translation, using only the

value for the Y-axis

translateZ(z) Defines a 3D translation, using only the

value for the Z-axis

scale3d(x,y,z) Defines a 3D scale transformation

scaleX(x) Defines a 3D scale transformation by giving

a value for the X-axis

scaleY(y) Defines a 3D scale transformation by giving

a value for the Y-axis

scaleZ(z) Defines a 3D scale transformation by giving

a value for the Z-axis

rotate3d(x,y,z,angle) Defines a 3D rotation

rotateX(angle) Defines a 3D rotation along the X-axis

rotateY(angle) Defines a 3D rotation along the Y-axis

rotateZ(angle) Defines a 3D rotation along the Z-axis

## Web Fonts

Web fonts allow Web designers to use fonts that are not installed on the user's computer.

When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed.

Your "own" fonts are defined within the CSS @font-face rule.

**Different Font Formats**

**TrueType Fonts (TTF)**

TrueType is a font standard developed in the late 1980s, by Apple and Microsoft. TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.

**OpenType Fonts (OTF)**

OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft. OpenType fonts are used commonly today on the major computer platforms.

**The Web Open Font Format (WOFF)**

WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata. The goal is to support font distribution from a server to a client over a network with bandwidth constraints.

**The Web Open Font Format (WOFF 2.0)**

TrueType/OpenType font that provides better compression than WOFF 1.0.

**SVG Fonts/Shapes**

SVG fonts allow SVG to be used as glyphs when displaying text. The SVG 1.1 specification define a font module that allows the creation of fonts within an SVG document. You can also apply CSS to SVG documents, and the @font-face rule can be applied to text in SVG documents.

**Embedded OpenType Fonts (EOT)**

EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

@font-face {  
    font-family: myFirstFont;  
    src: url(sansation\_light.woff);  
}  
  
div {  
    font-family: myFirstFont;  
}

## Transitions

CSS transitions allows you to change property values smoothly (from one value to another), over a given duration.

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 <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

div {  
    width: 100px;  
    height: 100px;  
    background: red;  
    -webkit-transition: width 2s; /\* Safari \*/  
    transition: width 2s;  
}

**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:

div {  
    -webkit-transition: width 2s, height 4s; /\* Safari \*/  
    transition: width 2s, height 4s;  
}

**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:

div {  
    -webkit-transition-delay: 1s; /\* Safari \*/  
    transition-delay: 1s;  
}

## Shadows

With CSS you can add shadow to text and to elements.

In this chapter you will learn about the following properties:

* text-shadow
* box-shadow

**CSS Text Shadow**

The CSS text-shadow property applies shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

h1 {  
    text-shadow: 2px 2px;  
}

**CSS box-shadow Property**

The CSS box-shadow property applies shadow to elements.

In its simplest use, you only specify the horizontal shadow and the vertical shadow:

div {  
    box-shadow: 10px 10px;  
}

## Animation

CSS animations allows animation of most HTML elements without using JavaScript or Flash!

**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.

## Keyframes

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

/\* The animation code \*/  
@keyframes example {  
    from {background-color: red;}  
    to {background-color: yellow;}  
}  
  
/\* The element to apply the animation to \*/  
div {  
    width: 100px;  
    height: 100px;  
    background-color: red;  
    animation-name: example;  
    animation-duration: 4s;  
}

## Pseudo-classes

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

* Style an element when a user mouses over it
* Style visited and unvisited links differently
* Style an element when it gets focus

/\* unvisited link \*/  
a:link {  
    color: #FF0000;  
}  
  
/\* visited link \*/  
a:visited {  
    color: #00FF00;  
}  
  
/\* mouse over link \*/  
a:hover {  
    color: #FF00FF;  
}  
  
/\* selected link \*/  
a:active {  
    color: #0000FF;  
}

The :first-child pseudo-class matches a specified element that is the first child of another element.

Match the first <p> element, In the following example, the selector matches any <p> element that is the first child of any element:

p:first-child {  
    color: blue;  
}

## Pseudo-elements

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

* Style the first letter, or line, of an element
* Insert content before, or after, the content of an element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all <p> elements:

#### p::first-line {     color: #ff0000;     font-variant: small-caps; }

The ::before pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each <h1> element:

h1::before {  
    content: url(smiley.gif);  
}

## Display properties

The display property is the most important CSS property for controlling layout.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Display: none

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The following example displays <span> elements as block elements:

span {  
    display: block;  
}

The following example displays <a> elements as block elements:

a {  
    display: block;  
}

## Positions

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

There are five different position values:

* Static

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

div.static {  
    position: static;  
    border: 3px solid #73AD21;  
}

* 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.

div.relative {  
    position: relative;  
    left: 30px;  
    border: 3px solid #73AD21;  
}

* 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:

div.fixed {  
    position: fixed;  
    bottom: 0;  
    right: 0;  
    width: 300px;  
    border: 3px solid #73AD21;  
}

* 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.

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;  
}

## Tooltips

A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element:

Create a tooltip that appears when the user moves the mouse over an element:

<style>  
/\* Tooltip container \*/  
.tooltip {  
    position: relative;  
    display: inline-block;  
    border-bottom: 1px dotted black; /\* If you want dots under the hoverable text \*/  
}  
  
/\* Tooltip text \*/  
.tooltip .tooltiptext {  
    visibility: hidden;  
    width: 120px;  
    background-color: black;  
    color: #fff;  
    text-align: center;  
    padding: 5px 0;  
    border-radius: 6px;  
   
    /\* Position the tooltip text - see examples below! \*/  
    position: absolute;  
    z-index: 1;  
}  
  
/\* Show the tooltip text when you mouse over the tooltip container \*/  
.tooltip:hover .tooltiptext {  
    visibility: visible;  
}  
</style>  
  
<div class="tooltip">Hover over me  
  <span class="tooltiptext">Tooltip text</span>  
</div>

## Text Effects

* + text-overflow
  + word-wrap
  + word-break

Text Overflow

The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.

p.test1 {  
    white-space: nowrap;   
    width: 200px;   
    border: 1px solid #000000;  
    overflow: hidden;  
    text-overflow: clip;   
}  
  
p.test2 {  
    white-space: nowrap;   
    width: 200px;   
    border: 1px solid #000000;  
    overflow: hidden;  
    text-overflow: ellipsis;   
}

Word Wrapping

The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.

p {  
    word-wrap: break-word;  
}

**Word Breaking**

The CSS word-break property specifies line breaking rules.

p.test1 {  
    word-break: keep-all;  
}  
  
p.test2 {  
    word-break: break-all;  
}