**Module 1: (August 16, 2022)**

WWW - uses the classical client/server architecture

* HTTP is text based request response protocol
* Through the http
* Web server software (IIS, Apache, etcn)
* We’re gonna use apache
* IIS pag visual studio

What is a web page

* Web pages are text files containing HTML
* Browsers only understand html
* HTML
  + A notation for describing
    - Formatting
* Creating a html pages
  + An html file must have an htm or gtml file extension
  + Html files can be created with text editors (is static) which can be dynamic if you use scripting
    - Notepad, notepad ++, psdpad
  + Or html editors (WYSIWYG Editors)
    - Microsoft front page
    - Macroedia dreamweaver
    - Netscape composer
    - Microsoft word
    - Visual studio
    - Eclipse is also capable raynao

HTML Basics

Html structure

* Is composed of elements and tags
  + Begins with html and ends with html
* Elements (tags) are nested one inside another
  + <html> <head></head> <body></body> </html>
* Tages have attributes
  + <img src=”logo,jpg”” alth + “logo”/>
* Html describes structure using two main section
  + <head> and <body>

HTML Code Formatting

* The HTML source code should be formatted to increase readability and facilitate debugging HWAHWAHWAHWA GUMAGAWA RIN AK
  + Every block element should start on a new line.
  + Every nested (block) element should be indented.
  + Browse ignore multiple whitespaeses

FIRST HTML PAGE

Test.html

<!DOCTYPE HTML>

<html>

<head>

<title>My First HTML Page</title>

</head>

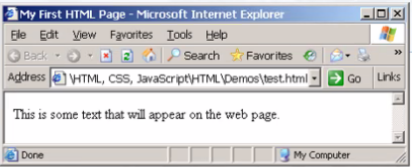
<body>

<p>This is some text that will appear on the web page</p>

</body>

</html>

Output:



Some Simple Tags

* Hyperlink Tags
  + <a href=”http://www.telerik.com/”

title

Some Smple TAgs – Example

Some-tags.html

<!DOCTYPE HTML>

<html>

<head>

<title>Simple Tags Demo</title>

</head>

<body>

<a href=”<http://ww.telerik.com/>” title=

“Telerik site”>This kjfsnjanwenkl

</body>

</html>

Output:

Tags Attributes

* Tags can have attributes
  + Attributes specify properties and behaviors
* Example:
  + <img id=”img1” src=”logo.gif” alt=”logo” />
* Few attributes can apply to every element:
  + Id, style, class, title
  + The id is unique in the document.
  + Content of title attribute is displayed as hint when the element is hovered with the mouse,
  + Some elements have obligatory (required) attributes.

Headings and Paragraphs

* <h1> Heading 1 </h1>
* <h2> Sub heading 2 </2>
* <h3>Sub heading 3</h3>

Paragraph Tags

* <p>This is my first paragraph</p>
* <p>This is my second paragraph</p>

Sections: div and span

* <div style+ “background: skyblue;”>
* This is a div</div>

headings. Html

<! DOCTYPE HTML>

<html>

<head><title>Headings and paragraphs

<body>

<h1>Heading 1</h1>

<h2>Sub heading 2</h2>

<h3>Sub heading 3</h3>

<p>This is my first paragraph </p>

<p>This is my second paragraph </p>

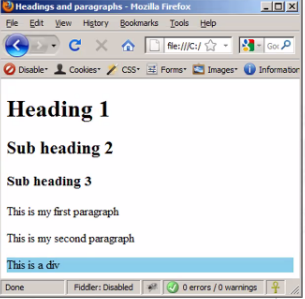
<div style=”background:skyblue”>

This is a div</div>

</body>

</html>

Output:



**INTRODUCTION TO HTML**

Preface

* It is important to have the correct vision and attitude towards HTML
  + HTML is only about structure, not appearance
  + Browser tolerate invalid HTML code and parse errors - you should not.\
* The <!DOCTYPE> Declaration
  + HTML documents must start with a document type definition (DTD)
    - It tells web browsers what type is the served code.
    - Possible versions HTML 4.01, XHTML 1.0 (Transitional or Strict), XHTML 1.1, HTML 5

* Example:
  + <!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN”

“<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>”>

* See <http://w3.org/QA/2002/04/valid-dtd-list.html> for a list of possible doctypes.

HTML vs. XHTML

* XHTML is more strict than HTML
  + Tags and attribute names must be in lowercase
  + All tags must be closed (<br/>, <img/>) while HTML allows <br> and <img> and implies missing closing tags (<p> par1 <p> par2)
  + XHTML allows only one root <html> element (HTML allows more than one)
* Many element attributes are deprecated in HTML, most are moved to CSS
* Attribute minimization if forbidden, e.g.
  + <input type=”checkbox” checked>
  + <input type=”checkbox” checked=”checked” />

Note: Web browsers load XHTML faster than HTML and valid code faster than invalid!

**The <head> Section**

* Contains information that doesn't show directly on the viewable page.
* Starts after the <!doctype> declaration
* Begins with <head> and ends with </head>
* Contains mandatory single <title> tage
  + <meta
  + <script
  + <style>
  + <!-- comments -->

<head> Section: <title> tag

* Title should be placed between <head> and </head> tags\
* Used to specify a title in the window title bar
* Search engines and people rely on tiles.

**<head> Section: <meta>**

* Meta tags additionally describe the content contained within the page
  + <meta name = “description” content = “HTML tutorial” />
  + <meta name=”keywords” content=”html, web design, styles” />
  + <meta http-equiv=”refresh” content=”5; url=http://www.telerik.com”/>

**<head> Section: <script>**

* The <script> element is used to embed scripts into an HTML document
  + Script are executed in the client’s Web browser
  + Script can live in the <head> and in the <body> sections
* Supported client-side scripting languages:
  + JavaScript (not Java)
  + VBScript
  + JScript

**The <script> Tag - Example**

<!DOCTYPE HTML>

<html>

<head>

<title>JavaScript Example</title>

<script type=”text/javascript”>

function say Hello (){

document.write(“<p> Hello World! <\/p>”;

}

</script>

</head>

<body>

<script type=

“text/javascript”>

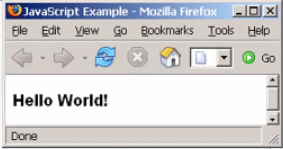
sayHello();

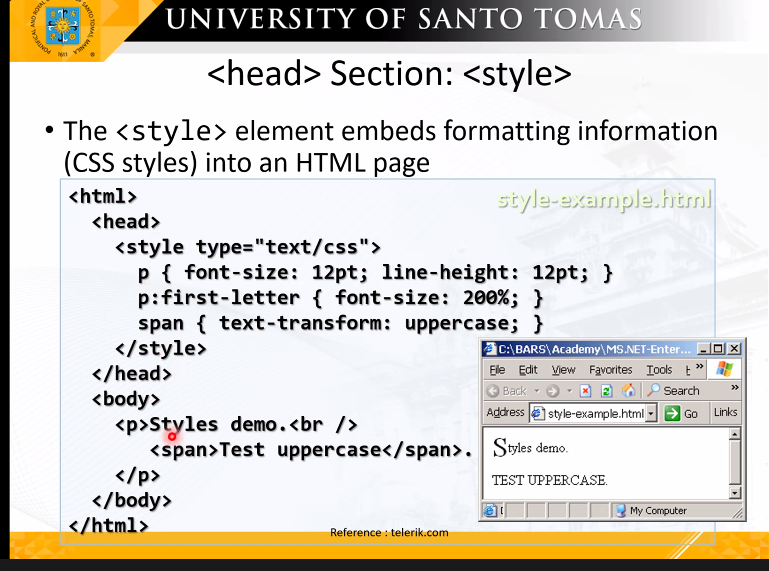
</script>

</body>

</html>

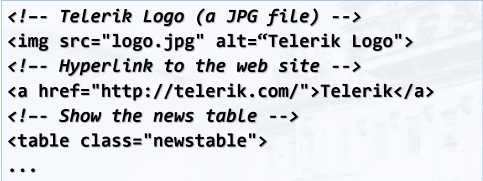
Output:





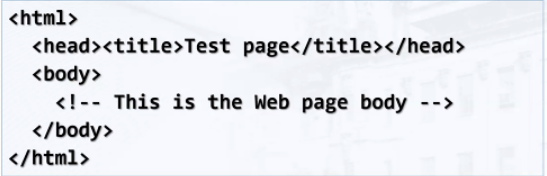
Comments: <! -- --> Tag

* Comments can exist anywhere between the <html> </html> tags
* Comments start with <!-- and end with →



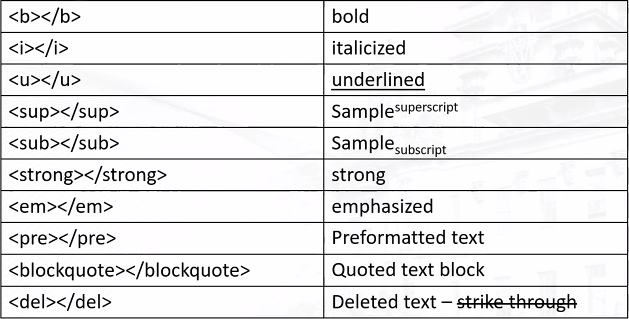
<body> Section: Introduction

* The <body> section describes the viewable portion of the page
* Starts after <head> </head> section
* Begins with <body> and ends with </body>

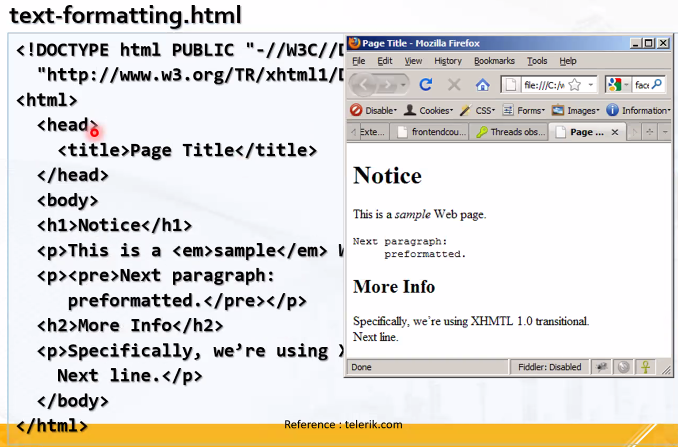


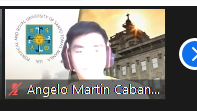
Text Formatting

* Text formatting tags modify the text between the opening tag and the closing tag
  + Ex.<b>Hello<b> makes “Hello” bold



Text Formatting — Example



parang kinukuha ni satanas

Natatawa ako tangina

Leche nakikita kita cgeh ayos bangs

PUTANGINA NAKAKAINIS HAHAHAHHAHAHA

**Hyperlinks: <a> Tag**

* Link to a document called form .html on the same server on the same directory:
  + <a href = “form.html”> Fill Our Form </a>
* Link to a documentary called parent.html on the same server in the parent diriecrotry:
  + <a href=”../parent.html”>Parent</a>
* Link to a document called cat.html on the same server in the subdirectory stuff:
  + <a href=”stuff/cat.html”>Catalog</a>
* Link to an external Web site
  + Always use a full URL, including “http://”, not just “www.somesite.com”
  + Using the target = “\_blank” attribute opens the link in a new window
* Link to an e-mail address:
  + <a href=”<mailto:bugs@example.com>?subject=Bug+Report”>
  + Please report bugs here (by e-mail only) </a>
* Link to a document called apple - now. html
  + On the same server, in same directory
  + Using an image as a link button:
  + <a href = “apply-now.html”
* Link to a document called index.html
  + On the same server, in the subdirectory english of the parent directory.
* Link to another location in the same document:
* <a href = “#section1”> Go to Introduction </a>

…

<h2 id = “section1”> Introduction </h2>

* Link to a specific location in another document.
  + <a href=”chapter3.html#section3//3/3”>Go to Section 3.1.1</a>

<! - - In chapter3.html – >

….

<div id=”section3.1.1”>

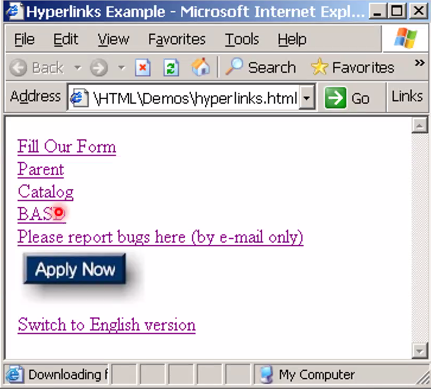
<h3> 3.1.1 Technical Bacground<h/3>

</div>

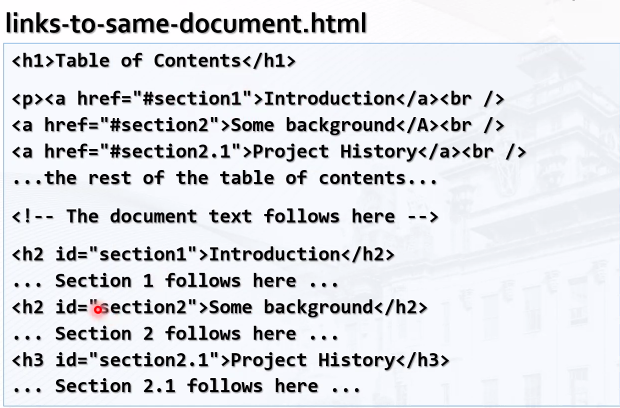
**Hyperlinks example**



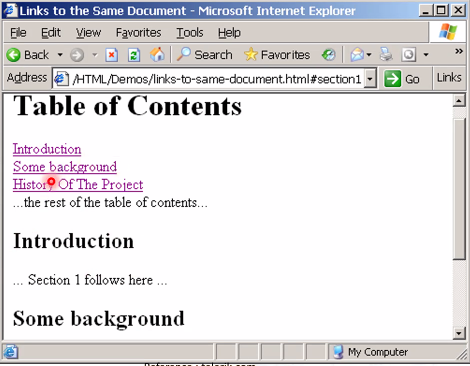
Output:



**Links to the Same Document**

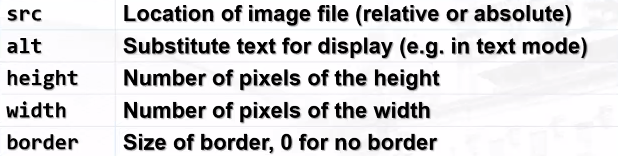


Output:



**Images: <img>tag**

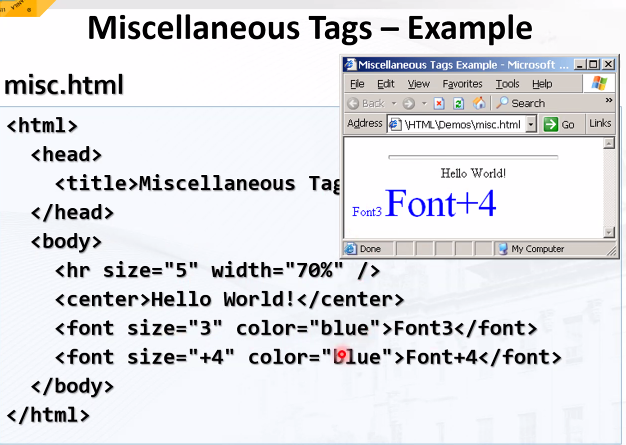
* Inserting an image with <img> tag:
  + img src=”/img/basd-logo.png”>
* Image attributes:



* Example:
  + <img src=”./php.png” alt=”PHP Logo” />

Miscellaneous Tags (can be used or not depende sayo)

* <hr /dikonakuha: Draws a horizontal line
* <center></center>:Deprecated!



**Ordered Lists: <ol> Tag**

* Create an Ordered List using <ol> </ol>:
  + ol type=”1”>

<li>Apple</li>

<li>Orange</li>

<li>Grapefruit</li>

</ol>

* Attribute values for type are 1, A, a, I, or i

**Unordered Lists <ul> Tag**

* Create an Unordered List using <ul></ul>
  + <ul type “disk”>

<li>Apple</li>

<li>Orange</li>

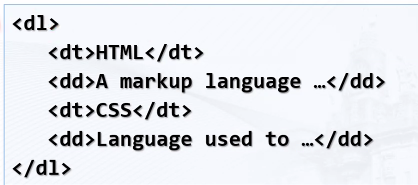
<li>Grapefruit</li>

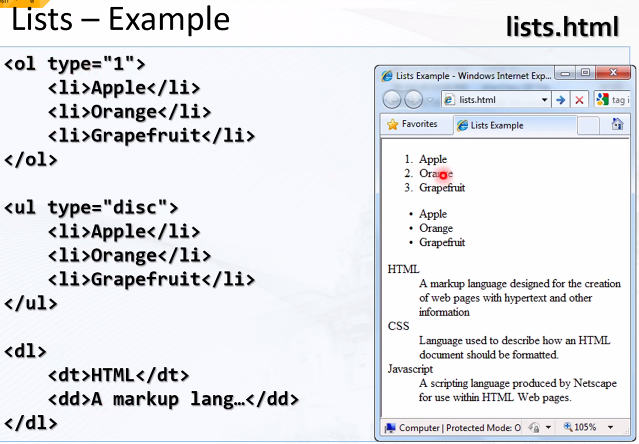
</ul>

* Attribute values for type are:
  + Disc, circle or square

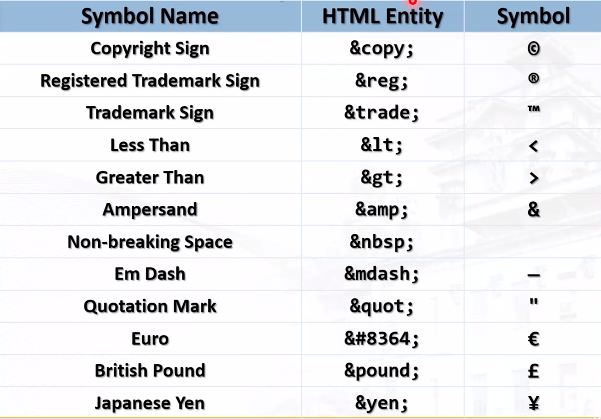
**Definition lists: <dl> tag**

* Create definition lists using <dl>
  + Pairs of text and associated definition; text is in <

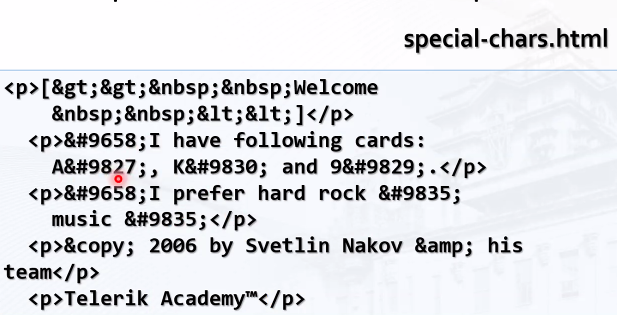


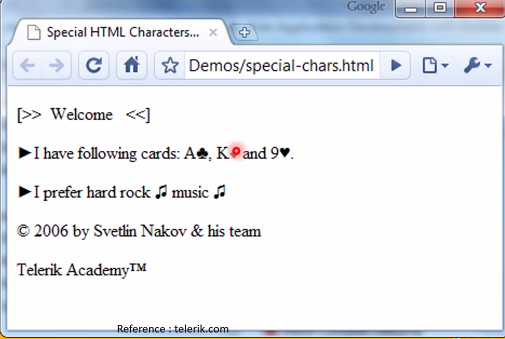


**HTML Special Characters**



Special Characters – Example



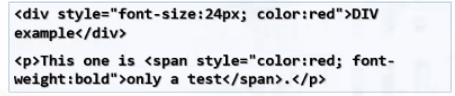


**Block and Inline Elements**

* Block elements add a line break before and after them
  + <div> is a block element
  + Other block elements are <table>, <hr> headings, lists, <p>, and etc.
* Inline elements don’t break the text before and after them
  + <span> is an inline element
  + Most HTML elements are inline, e.g. <a>

**The <div> Tag**

* <div> creates ivisions within a page
* Block style element
* Used with CSS
* Example:



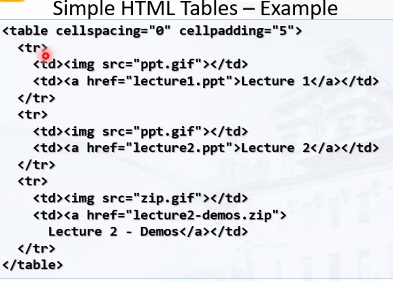


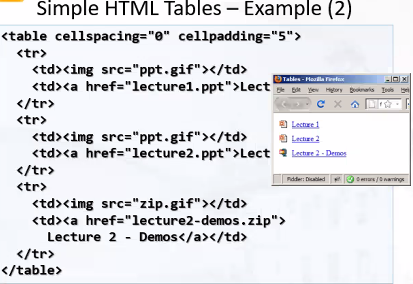
**The <span> Tag**

* Inline style element
* ulol

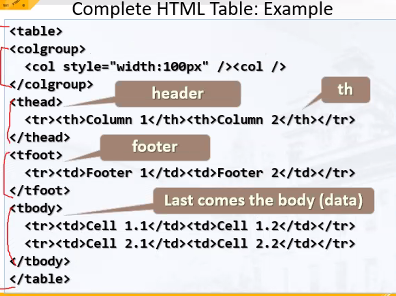
**HTML Tables**

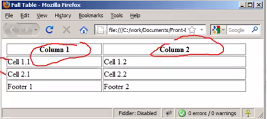
* Tables represent tabular data
  + A table consists of one or several rows
  + Each rows has one or more columns
* Tables comprised of several core tags: <table></table>: begin / end the table
  + <tr></tr>: create table row
  + <td></td>: create tabular data (cell)
* Tables should not be used for layout. Use CSS floats and positioning styles instead.

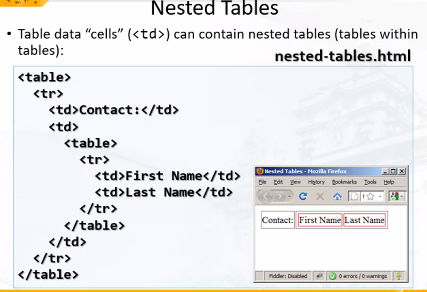


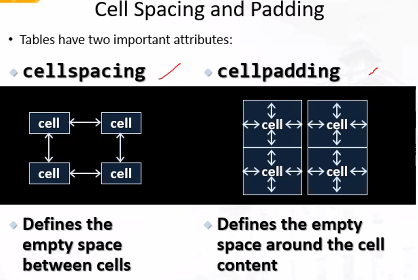


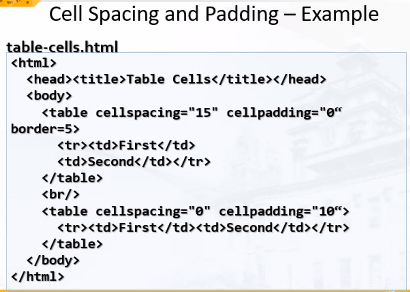


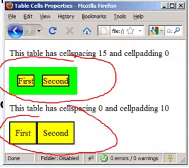












**Form Fields**

* Single-line text input fields:
  + <input type=”text” name=”FirstName” value=”This is a text field” />
* Multi-line text area fields:
  + <textarea name=”Commnets”>This is a multi-line text field</textarea>
* Hidden fields contain data not shown to the user:
  + <input type=”hidden” name=”Account” value=”This is a hidden text field” />
    - Often used by JavaScript code

**Fieldsets**

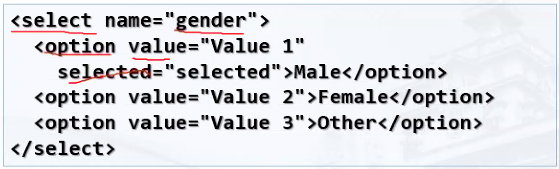
* Fieldsets are used to enclose a group of related form fields:
  + 

**Form Input Controls**

* Checkboxes:
  + <input type=”checkbox” name=”fruit” value=”apple” />
* Radio buttons:
  + <input type=”radio” name=”title” value=”Mr.” />
* Radio buttons can be grouped, allowing only one to be selected from a group”
  + <input type=”radio” name=”city” value=”Lom” />
  + <input type=”radio” name=”city” value=”Ruse” />

**Other Form Controls**

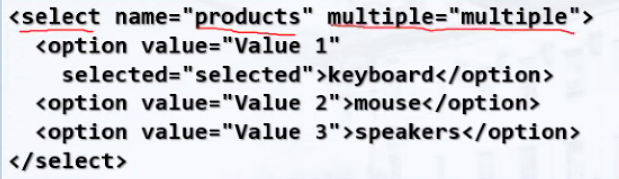
* Dropdown menus:



* Submit button:



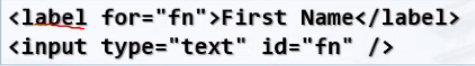
* Password input – a text field which makes masks the entered text with \* signs
  + <input type=”password” name=”pass” />
* Multiple select field – displays the list of items in multiple lines, instead of one



* File input – a field used for uploading files

**Labels**

* Form labels are used to associate an explanatory text to a form field using the field’s ID



* Clicking on a label focuses its associated field (checkboxes are toggled, radio buttons are checked)
* Labels are both a usability and accessibility feature and are required in order to pass accessibility validation/

Hello gud am

Inaantok ako

nagmemessenger

Hhaah kachat bff padua

**jaahahahahHAAHAHAHAHAHA OO NGA EH RINIG NA RINIG HAHAHAHAHAHAHHAHAHAHAHAHAHAHHAHAHAHAHAHAHAHHAHAHA AYAW KO NA**

**HAHAHAHAHAHHAHAHAHHAHAHAHHAHAHAHA GAGO GAGAGAGGAHAHAHAHAHAHAHHAHAHAHAHAHAHHAHAAHAHAHAAHHAAHAHAHHAAHHAAHAHAHAHA PUTANGINA KASI AHHAHAHAHAHAHAHAHAHAAHAHAHAH HAHAHAHAHHHAHAHAHAHHAHAHHAHAHAHAHAHAHHA AYOKO NA**

**hahahhHAHAHAHAHHAHAA NASA RECORDING YAN**

**CSS**

A New Philosophy

* Separate content from presentation

CSS Introduction

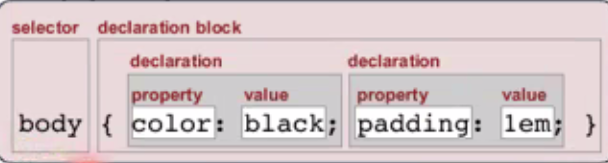
* Cascading Style Sheets (CSS)
  + Used to describe the presentation of documents.
  + Define sizes, spacing, fonts, colors, layout, etc.
  + Improve content accessibility.
  + Improve flexibility.
* Designed to separate presentation from content
* Due to CSS, all HTML presentation tags and attributes are deprecated, e.g. font, center,...
* CSS can be applied to any XML document
  + Not just to HTML /XHTML
* CSS can specify different styles for different media
  + On-screen
  + In print
  + Handheld, projection, etc
  + …even by voice or Braille-based reader

Why “Cascading”?

* Priority scheme determining which style rules apply to elements
  + **Cascade priorities or specify (weight)** are calculated and assigned to the rules.
  + Child elements in the HTML DOM tree inherit styles from their parent.
    - Can override them
    - Control via !important rule
* Some CSS styles are inherited and some are not
  + Text-trelated and list-related properties punyeta
  + Box-related and positioning styles are not inherited - width, height, border, margin, padding, position, float etc.

Style Sheets Syntax

* Stylesheets consists of rules, selectors, declarations, properties, and values
* Selectors are separated by commas
* Declarations are separated by semicolons
* Properties and values are separated by colons
  + h1,h2,h3 { color: green; font-weight: bold; }



Selectors

* Selectors determine which element the rule applies to:
  + All elements of specific type (tag)
  + Those that match a specific attribute (id, class)
  + Screenshot gaming tong c perol
* Examples:
  + Headeejfnrbskgjkbjfbnjkfdfdfnjkdfgjkdfgjkfbkgbkfdbg GRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
* Three primary kinds of selectors:
  + By tag (type selector):
    - h1 { font-family: verdana,sans-serif; }
  + By element id:
    - #element\_id { color: #ff0000; }
  + By element class name (only for HTML)
    - myClass {border: 1px solid red}
  + Selectors can be combined with commas:
    - h1, .link, #top-link {font-weight: bold}
      * This will match <h1> tags, elements with class link, and element with id top-link
* Pseudo-classes define state
  + :hover, :visited,:active, :lang
* Pseudo-elements define element “parts” or are used to generate content
  + :first-line, :before, :after
* Match relative to element placement
  + P a {text-declaration: underline}
  + This will math all <a> tags that are inside of <p>
* \* - universal selector (avoid or use with care!):
  + p \* {color: black}
  + This will match all descendants of <p> element
* + selector - used to match “next sibling”:
  + Img + .link {float:right}
  + THis will match all siblings with class name link that appear immediately after <img> tag.
* + selector - matches direct child nodes:
  + p> .error {font-size: 8px}
  + This will match all elements with class error, direct children of <p> tah
* [ ] - matches tag attributes by regular expression:
  + img [alt = logo] {border: none}
  + This will match all

Values in the CSS Rules

* Colors are set in RGB format (decimal or hex):
  + Example: #a0a6aa = rgb(160, 166, 170)
  + Predefined color aliases exist
* Numeric values are specified in:
  + Pixels
  + Points, inches, centimeters, millimeters
  + Percentages
  + Zero can be used with no unit

Default Browser Styles

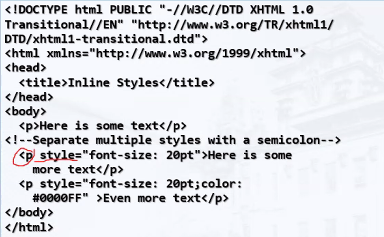
* Browsers have default CSS styles
  + Used when there is no CSS information or any other style information in the document
* Caution: default styles differ in browsers
  + Margins, paddings and font sizes differ most often and usually developers reset them

Linking HTML and CSS

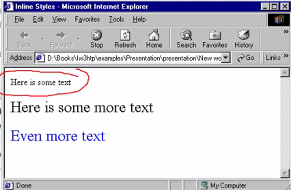
* HTML (content) and CSS (presentation acan be linked in three ways:
  + Inline: the CSS rules in the style attribute
    - No selectors are needed
  + Embedded: in the <head> in a <style> tag
  + External: CSS rules in separate file (best)
    - Usually a file with .css extension
    - Linked via <link rel=”stylesheet” href=..>tag or @import directive in embedded CSS block
  + Using external files is highly recommended
    - Simplifies the HTML document
    - Improves page load speed as the CSS file is cached

**Inline Styles: Example**

**Inline-style.html**



**Output:**



**CSS Cascade (precedence)**

* There are browser, user, and author stylesheets with “normal” and “importatn declarations
  + Browser styles (least priority)
  + Normal user styles
  + Normal author styles (external, in head, inline)
  + Important author styles
  + importangina

CSS Specificity

* CSS specify is used to determine the precedence of style declarations with the same origins

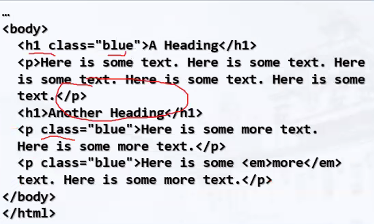
Embedded Styles

* Embedded in the HTML in the <style> tag
  + <style type=”text/css”>
* The <style>tag is placed in the <head> section of the document
* Type attribute specifies in the MIME type
  + MIME describes the format of the content
  + Other MIME types include text/html, image/gif, text/javascript
* Used for document-specific styles

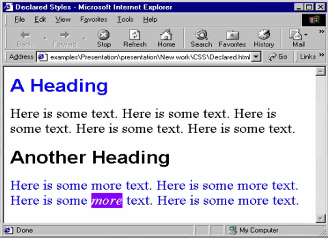
**Embedded Styles: Example**

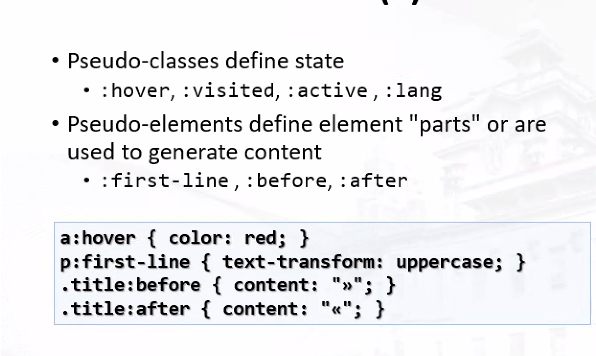
**embedded-stylesheets.html**

****

****

**Output:**

****



External CSS Styles

Text-related CSS Properties

* Color - specifies the color of the text
* Font size - size of font: xx-small, x-small, small, medium, large, xx-large, smaller, larger or numeric values.
* Font family (comma separated font names)
* Font weight (can be normal, bold, bolder, lighter or number ranging from 100-900)
* Font style
* Text decoration
* Text align (alignment of text or other text)

Backgrounds

* background-image
* background-color
* Background-repeat
* background-attachment
  + fixed/scroll
* Background-position

**Margin and Padding**

* Margin and padding define the spacing around the elements
  + Numerical value, e.g. 10px or - 5px
  + Can be defined for each of the four sides separately - marin-top,padding-left,...
  + Margin is the spacing outside of the border
  + Padding is the spacing between the border and the content
  + What are collapsing margins?