Web Design Course

WORKING WITH CSS AN EXAMPLE

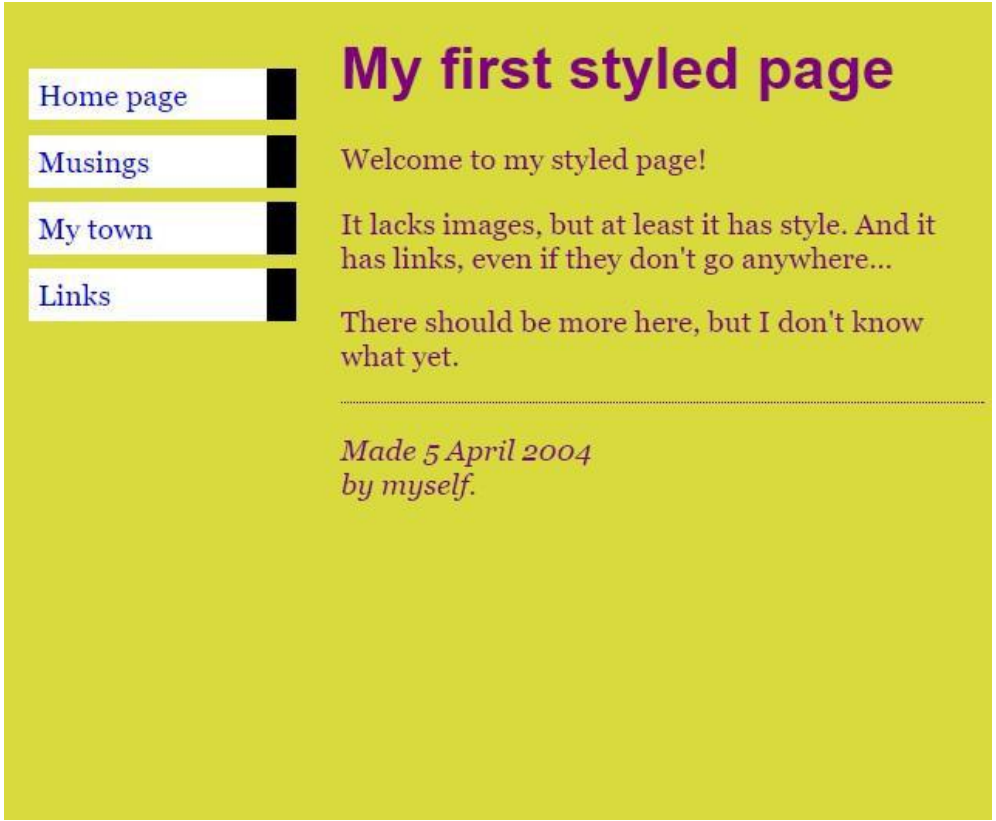
Lecture by Mis. Zon

IU #	IU Description	Required / Optional
01	Web Design Concepts	Required
02	HTML Basics	Required
03	Advanced HTML & Web Browsers	Required
04	Structuring & Styling with CSS	Required
05	**Working with CSS : An Example**	**Required**
06	Javascript Basics	Required
07	Advanced Javascript	Required

IU Contents

S. No.	Topic Description
01	CSS Case Study – From Scratch
02	Step 1 – Writing the HTML
03	Step 2 – Adding Some Colors
04	Step 3 – Adding Fonts
05	Step 4 – Adding a Navigation Bar
06	Step 5 - Styling the Links
07	Step 6 – Adding a Horizontal Line
08	Step 7 – Putting the Style Sheet in a Separate File
09	CSS Development Tool – RapidCSS
10	Screen Resolution & Responsive Web Design

- This case study explains how to create an HTML file, a CSS file and to make them work together.
- At the end of this Case Study, you will have made an HTML file that looks like this:

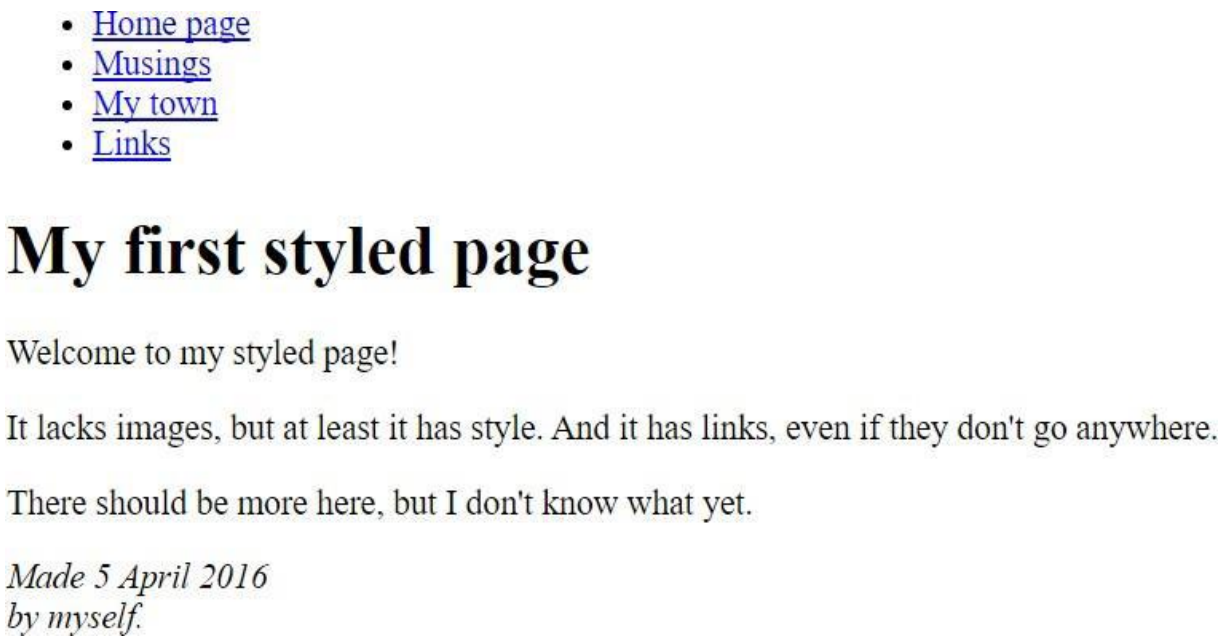


- Open your Dreamweaver, start with an empty window and type the following

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>My first styled page</title>
</head>
<body>
<!-- Site navigation menu -->
<ul class="navbar">
  <li><a href="index.html">Home page</a></li>
  <li><a href="musings.html">Musings</a></li>
  <li><a href="town.html">My town</a></li>
  <li><a href="links.html">Links</a></li>
</ul>
```

```
<!-- Main content -->
<h1>My first styled page</h1>
<p>Welcome to my styled page!</p>
<p>It lacks images, but at least it has style.
And it has links, even if they don't go
Anywhere. </p>
<p>There should be more here, but I don't know
what yet. </p>
<address>Made 5 April 2017<br>
  by myself.</address>
</body>
</html>
```

- The Page looks like below in the browser.
- The first line of the HTML tells the browser which type of HTML this is (DOCTYPE means Document TYPE). In this case, it is HTML version 4.01.

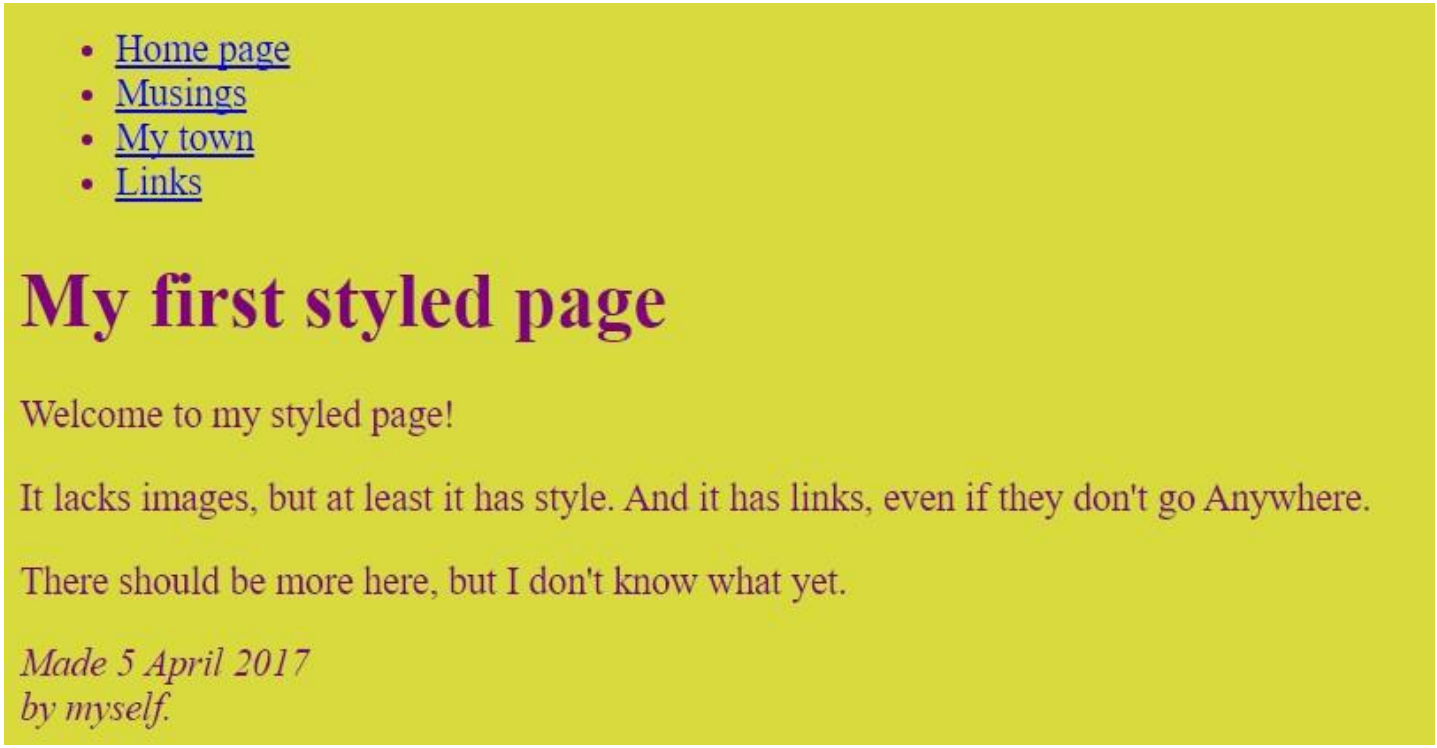


- One easy thing to do to make the page more aesthetic is to add some colors.
- We will start with a style sheet embedded inside the HTML file. Later, we will put the HTML and the CSS in separate files.
- Separate files is good, since it makes it easier to use the same style sheet for multiple HTML files.
- You only have to write the style sheet once. But for this step, we just keep everything in one file.

- We need to add a <style> element to the HTML file.
- The style sheet will be inside that element.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
 <title>My first styled page</title>
 <style type="text/css">
 body {
   color: purple;
   background-color: #d8da3d}
 </style>
</head>
<body>
… Remaining Code
```

- The first line says that this is a style sheet and that it is written in CSS (“text/css”).
- The second line says that we add style to the “body” element.
- The third line sets the color of the text to purple and the next line sets the background to greenish yellow.
- It looks like below in browser



- Style sheets in CSS are made up of *rules*. Each rule has three parts:
  - the *selector* (in the example: “body”), which tells the browser which part of the document is affected by the rule;
  - the *property* (in the example, 'color' and 'background-color' are both properties), which specifies what aspect of the layout is being set;
  - and the *value* ('purple' and '#d8da3d'), which gives the value for the style property.
- The example shows that rules can be combined. We have set two properties, so we could have made two separate rules

```
body { color: purple }
```

```
body { background-color: #d8da3d }
```

- But since both rules affect the body, we only wrote “body” once and put the properties and values together.

- The background of the body element will also be the background of the whole document.
- We haven't given any of the other elements (p, li, address…) any explicit background, so by default they will have none (or: will be transparent).
- The 'color' property sets the color of the text for the body element, but all other elements inside the body inherit that color, unless explicitly overridden.
- Now save this file (use “Save” from the File menu) and go back to the browser window and refresh to see the page with colours.

- Can make some distinction in the fonts for the various elements of the page.
- Set the text in the “Georgia” font, except for the **h1** heading, which we'll give “Helvetica.”
- On the Web, you can never be sure what fonts your readers have on their computers, so we add some alternatives as well
- if Georgia is not available, Times New Roman or Times are also fine, and if all else fails, the browser may use any other font with serifs.
- If Helvetica is absent, Geneva, Arial and SunSans-Regular are quite similar in shape, and if none of these work, the browser can choose any other font that is serif-less.

- Insert the below code inside style element

```
<style type="text/css">
 body {
   font-family: Georgia, "Times New Roman",
        Times, serif;
   color: purple;
   background-color: #d8da3d }
 h1 {
   font-family: Helvetica, Geneva, Arial,
        SunSans-Regular, sans-serif }
</style>
```

- If you save the file again and press “Reload” in the browser, there should now be different fonts for the heading and the other text.

- The list at the top of the HTML page is meant to become a navigation menu.
- Many Web sites have some sort of menu along the top or on the side of the page.
- Create it on the left side.
- The menu is already in the HTML page. It is the <ul> list at the top.
- So we need to move the list to the left and move the rest of the text a little to the right, to make room for it.
- The CSS properties we use for that are 'padding-left' (to move the body text) and 'position', 'left' and 'top' (to move the menu).

```
body {
  padding-left: 11em;
  font-family: Georgia, "Times New Roman",
       Times, serif;
  color: purple;
  background-color: #d8da3d }
 ul.navbar {
  position: absolute;
  top: 2em;
  left: 1em;
  width: 9em }
 h1 {
  font-family: Helvetica, Geneva, Arial,
       SunSans-Regular, sans-serif }
```

- Save the file again and reload it in the browser, you should now have the list of links to the left of the main text.
- Now the Page look like below

- The 'position: absolute' says that the ul element is positioned independently of any text that comes before or after it in the document
- 'left' and 'top' indicate what that position is. In this case, 2em from the top and 1em from the left side of the window.
- '2em' means 2 times the size of the current font. E.g., if the menu is displayed with a font of 12 points, then '2em' is 24 points.
- The 'em' is a very useful unit in CSS, since it can adapt automatically to the font that the reader happens to use.
- Most browsers have a menu for increasing or decreasing the font size.
- Try it and see that the menu increases in size as the font increases, which would not have been the case, if we had used a size in pixels instead.

- The navigation menu still looks like a list, instead of a menu.
- Let's add some style to it. We'll remove the list bullet and move the items to the left, to where the bullet was.
- We'll also give each item its own white background and a black square.
- We also haven't said what the colors of the links should be, so let's add that as well: blue for links that the user hasn't seen yet and purple for links already visited.

```
ul.navbar {
    list-style-type: none;
    padding: 0;
    margin: 0;
    position: absolute;
    top: 2em;
    left: 1em;
    width: 9em }
```

```
ul.navbar li {
    background: white;
    margin: 0.5em 0;
    padding: 0.3em;
    border-right: 1em solid black }
  ul.navbar a {
    text-decoration: none }
  a:link {
    color: blue }
  a:visited {
  color: purple }
```

- Traditionally, browsers show hyperlinks with underlines and with colors.
- Usually, the colors are similar to what we specificed here: blue for links to pages that you haven't visited yet (or visited a long time ago), purple for pages that you have already seen.
- In HTML, hyperlinks are created with <a> elements, so to specify the color, we need to add a style rule for “a”.
- To differentiate between visited and unvisited links, CSS provides two “pseudo-classes” (:link and :visited).
- They are called “pseudo-classes” to distinguish them from class attributes, that appear in the HTML directly, e.g., the class="navbar" in our example.

- Save the file again and reload it in the browser, you should now have the menu to the left of the main text.
- Now the Page look like below

- The final addition to the style sheet is a horizontal rule to separate the text from the signature at the bottom.
- We will use 'border-top' to add a dotted line above the <address> element:
- Add the following style

```
address {
    margin-top: 1em;
    padding-top: 1em;
    border-top: thin dotted }
```

- Now our style is complete.
- We now have an HTML file with an embedded style sheet.
- But if our site grows we probably want many pages to share the same style.
- There is a better method than copying the style sheet into every page: if we put the style sheet in a separate file, all pages can point to it.
- To make a style sheet file, we need to create another empty text file.
- Then cut and paste everything that is inside the <style> element from the HTML file into the new window.
- Don't copy the <style> and </style> themselves. They belong to HTML, not to CSS.

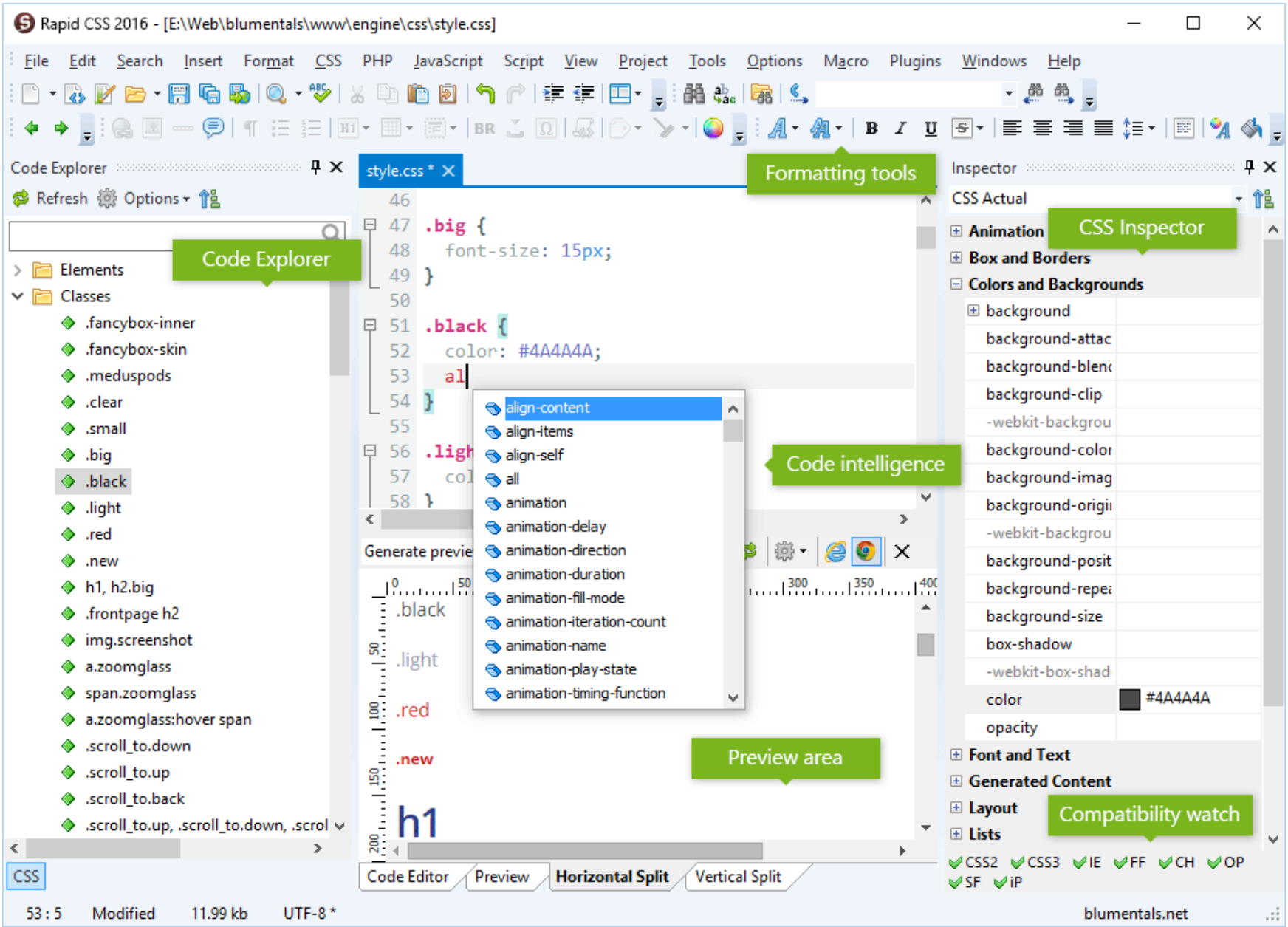
- Now go back to the window with the HTML code.
- Remove everything from the <style> tag up to and including the </style> tag and replace it with a <link> element, as follows:

```
<head>
  <title>My first styled page</title>
  <link rel="stylesheet"
href="mystyle.css">
</head>
```

```
body {
  padding-left: 11em;
  font-family: Georgia, "Times New Roman",
        Times, serif;
  color: purple;
  background-color: #d8da3d }
ul.navbar {
  list-style-type: none;
  padding: 0;
  margin: 0;
  position: absolute;
  top: 2em;
  left: 1em;
  width: 9em }
h1 {
  font-family: Helvetica, Geneva, Arial,
        SunSans-Regular, sans-serif }
ul.navbar li {
  background: white;
  margin: 0.5em 0;
  padding: 0.3em;
  border-right: 1em solid black }
ul.navbar a {
  text-decoration: none }
a:link {
  color: blue }
a:visited {
  color: purple }
address {
  margin-top: 1em;
  padding-top: 1em;
  border-top: thin dotted }
```

- This will tell the browser that the style sheet is found in the file called “mystyle.css”
- Since no directory is mentioned, the browser will look in the same directory where it found the HTML file.
- If you save the HTML file and reload it in the browser, you should see no change: the page is still styled the same way, but now the style comes from an external file.
- The next step is to put both files, mypage.html and mystyle.css on your Web site.

- Download and install Evaluation version of RapidCSS Editor from www.rapidcsseditor.com
- Rapid CSS editor makes it easy to create, design and edit modern CSS-based websites.
- Write the CSS code manually or let the style sheet editor do it for you!
- It is easy because of the many features such as auto complete, code inspector, CSS checker and instant built-in multi-browser preview. Rapid CSS editor is designed to save youtime and make your job easier.


Rapid CSS 2016 - [E:\Web\blumentals\www\engine\css\style.css]
File Edit Search Insert Format CSS PHP JavaScript Script View Project Tools Options Macro Plugins Windows Help
Code Explorer
Refresh Options
Elements
Classes
.fancybox-inner
.fancybox-skin
.meduspods
.clear
.small
.big
.black
.light
.red
.new
h1, h2.big
.frontpage h2
img.screenshot
a.zoomglass
span.zoomglass
a.zoomglass:hover span
.scroll_to.down
.scroll_to.up
.scroll_to.back
.scroll_to.up, .scroll_to.down, .scrol
CSS
Code Explorer
style.css *
Formatting tools
46
47 .big {
48 font-size: 15px;
49 }
50
51 .black {
52 color: #4A4A4A;
53 al
54 }
55
56 .ligh
57 col
58 }
align-content
align-items
align-self
all
animation
animation-delay
animation-direction
animation-duration
animation-fill-mode
animation-iteration-count
animation-name
animation-play-state
animation-timing-function
Code intelligence
Generate previe
.black
.light
.red
.new
h1
Preview area
Code Editor
Preview
Horizontal Split
Vertical Split
Inspector
CSS Actual
CSS Inspector
Animation
Box and Borders
Colors and Backgrounds
background
background-attac
background-blenc
background-clip
-webkit-backgrou
background-color
background-imag
background-origi
-webkit-backgrou
background-posit
background-repea
background-size
box-shadow
-webkit-box-shad
color #4A4A4A
opacity
Font and Text
Generated Content
Layout
Lists
Compatibility watch
CSS2 CSS3 IE FF CH OP
SF iP
53 : 5 Modified 11.99 kb UTF-8 *
blumentals.net

- **Quick and lightweight**
  Loads much faster than any other CSS editor or IDE with similar features
- **Powerful syntax highlighting Updated!**
  Supports HTML, CSS, LESS, SASS, JavaScript, PHP, XML, ASP, Perl and more
- **Code intelligence Updated!**
  Tons of intelligent HTML and CSS code completion, navigation and suggestion features
- **Smart code re-use**
  Code snippet library and code templates with assignable shortcuts
- **HTML5 and CSS3 ready Updated!**
  Coding features are up-to-date with modern standards
- **Direct FTP/SFTP/FTPS Updated!**
  Edit directly on your web server or publish local development copy updates with a single click

- **Mobile web development**
  Media queries, viewport assistant, screen-size preview
- **Advanced search and replace**
  Quick search, detailed search, file search, regular expression support, detailed results and more
- **Powerful CSS tools Updated!**
  Compatibility watch, prefixer, shadow assistant, box assistant, web font assistant and much more
- **Browser preview**
  Built-in multi-browser preview, split-screen mode, screen-size testing, XRay
- **Powerful color picker Updated!**
  Advanced color picker with project color management
- **Integrated validation Updated!**
  Spell checker, CSS checker, W3 HTML and CSS validator

- Desktops & Laptops
  - 1024×768 pixels or higher
- Apple Products

	Pixel Size	Viewport
iPhone		
7 Plus	1080 x 1920	414 x 736
7	750 x 1334	375 x 667
6 Plus/6S Plus	1080 x 1920	414 x 736
6/6S	750 x 1334	375 x 667
5	640 x 1136	320 x 568
iPod		
Touch	640 x 1136	320 x 568
iPad		
Pro	2048 x 2732	1024 x 1366
Third & Fourth Generation	1536 x 2048	768 x 1024
Air 1 & 2	1536 x 2048	768 x 1024
Mini	768 x 1024	768 x 1024
Mini 2 & 3	1536 x 2048	768 x 1024

- Desktops & Laptops
  - 1024×768 pixels or higher
- Android Products

	Pixel Size	Viewport
Phones		
Nexus 6P	1440 x 2560	411 x 731
Nexus 5X	1080 x 1920	411 x 731
Google Pixel	1080 x 1920	411 x 731
Google Pixel XL	1440 x 2560	411 x 731
Samsung Galaxy Note 5	1440 x 2560	480 x 853
LG G5	1440 x 2560	480 x 853
One Plus 3	1080 x 1920	480 x 853
Samsung Galaxy S7	1440 x 2560	340 x 640
Samsung Galaxy S7 Edge	1440 x 2560	411 x 731
Tablets		
Nexus 7 (2013)	1200 x 1920	600 x 960
Nexus 9	1536 x 2048	768 x 1024
Samsung Galaxy Tab 10	800 x 1280	800 x 1280
Chromebook Pixel	2560 x 1700	1280 x 850

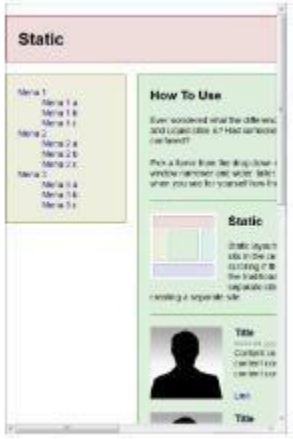
- The width of a website is fixed regardless of the size of the browser window.
- The designer is certain that the website doesn’t change. With resolutions higher than the original the website looks good but it doesn’t optimally use the whole space available – the wider the window, the larger the margins on the sides.
- It’s worse with resolutions lower than the original – the site just doesn’t fit the screen.
- It doesn’t fit well with smaller screens of mobile devices.

❑ Fixed Width Layout in Various Resolutions

fixed-width layout, 1366×768

fixed-width layout, 1440×900

fixed-width layout, 1024×768



480×800

- **“No Mobile” Approach**
  - Website that does not offer a tailored mobile experience (either app or website)
  - Can still be viewable on most devices, but not particularly usable
- **Native Mobile Applications**
  - Barrier to entry ○ – Device and even OS version
  - Separation of content and features & Costly
- **Mobile Websites**
  - Offer experience tailored to mobile devices
- **Responsive (universal) design**
  - One website for all devices!
  - Optimized for different contexts using:
  - Fluid grids
  - Flexible media
  - CSS Media Queries

- **Responsive Web design** is the approach that suggests that design and development should respond to the user's behaviour and environment based on screen size, platform and orientation.
- The practice consists of a mix of flexible grids and layouts, images and an intelligent use of CSS media queries.
- As the user switches from their laptop to iPad, the website should automatically switch to accommodate for resolution, image size and scripting abilities.
- In other words, the website should have the technology to automatically *respond* to the user's preferences.
- This would eliminate the need for a different design and development phase for each new gadget on the market.

- The elements in the page Adjusts itself to various resolutions

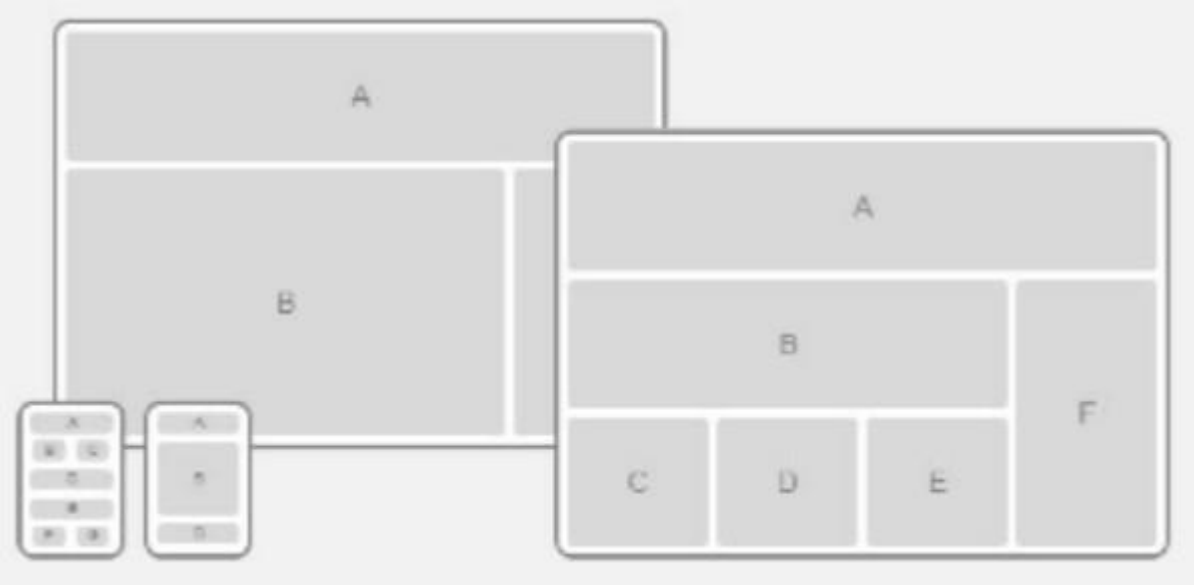
responsive layout, 1366×768

responsive layout, 1440×900

responsive layout, 1024×768

480×800

- A way of organizing different pieces of information along vertical and horizontal axes.
- Fluid grid = width of boxes is defined in percentage rather than fixed units (pixels, em)
  - Can grow or shrink as the screen width changes
  - Allows for utilizing all available space
  - Avoids issue of horizontal scrolling

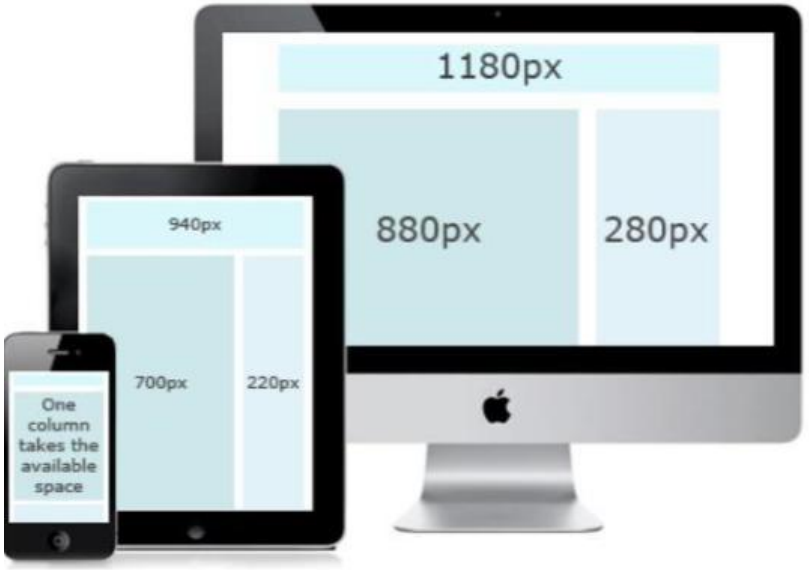


- Similar concept to fluid grids, but applied to images and movies so dimensions of media can change depending on screen size.

  img, object { max-width: 100%; }
- Can result in problems in older browsers that don't support max-width (Internet Explorer 7)
  - Set width to 100%
- Image considerations
  - If images are going to be viewed at small sizes, no point in serving large resolution images?
  - Existing images look blurry on displays with high pixel density
  - May serve different images based on media queries?

- Part of CSS3 specification
- Extends existing media type functionality that allowed style sheets for print, screen, etc.
- Gives more granular control as to when different CSS rules are applied
  - Based on media features such as screen width/height, screen orientation, pixel density, etc…
- Example
  - @media screen and (min-width: 650px) and (max-width: 960px) {...} ---------
  - @media (min-width:800px) and (max-width:1200px) and (orientation:portrait) { ... } ----------
  - @media screen and (-webkit-device-pixel-ratio: 1.5), screen and (resolution: 144dpi) { ... }

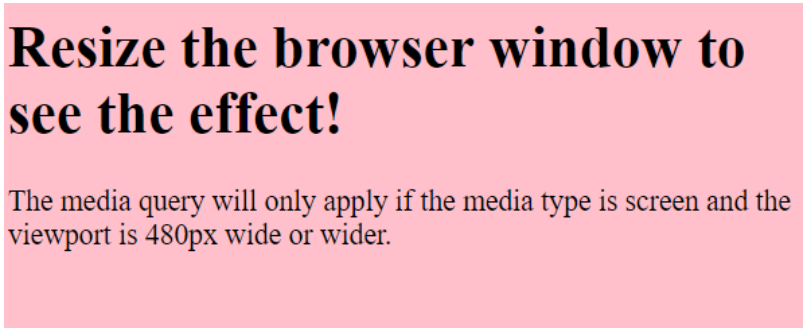
- Breakpoints are defined resolution points (typically width) specified in media queries at which different CSS styles are applied.
- Breakpoints example:
  - Below 650 (small screen)
  - 650-960 (tablet)
  - Above 960 (desktop)
- Should be chosen based on your content rather than known resolutions of popular devices



```
<!DOCTYPE html>
<html>
<head>
<style>
body {
   background-color: pink;
}
@media screen and (min-width: 480px) {
   body {
      background-color: lightgreen;
   }
}
</style>
</head>
<body>
```

Screen Resolution Greater Than 480

Screen Resolution Less Than 480



```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, scale=1.0">
<style>
.wrapper {overflow: auto;}

#main {margin-left: 4px;}

• #leftsidebar { float: none; width: auto;
• }
• #menulist { margin: 0;
• padding: 0;
• }
• .menuitem { background: #CDF0F6;
• border: 1px solid #d4d4d4;
• border-radius: 4px; list-style-type: none; margin: 4px; padding: 2px;
• }
• @media screen and (min-width: 480px) { #leftsidebar {width: 200px; float: left;} #main {margin-left: 216px;}
• }
```

Screen Resolution Greater Than 480

initial-

- Menu-item 1
- Menu-item 2
- Menu-item 3
- Menu-item 4
- Menu-item 5

Resize the browser window to see the effect!

This example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider. If the viewport is less than 480 pixels, the menu will be on top of the content.

Screen Resolution Less Than 480

- Menu-item 1
- Menu-item 2
- Menu-item 3
- Menu-item 4
- Menu-item 5

Resize the browser window to see the effect!

This example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider. If the viewport is less than 480 pixels, the menu will be on top of the content.

```
<body>
<div class="wrapper">
 <div id="leftsidebar">
   <ul id="menulist">
     <li class="menuitem">Menu-item  1</li>
     <li class="menuitem">Menu-item  2</li>
     <li class="menuitem">Menu-item  3</li>
     <li class="menuitem">Menu-item  4</li>
     <li class="menuitem">Menu-item  5</li>
  </ul>
 </div>
 <div id="main">
   <h1>Resize the browser window to see the effect!</h1>
   <p>This example shows a menu that will float to the left of the
page if the viewport is 480 pixels wide or wider. If the viewport is
less than 480 pixels, the menu will be on top of the content.</p>
 </div>
</div>


</body>
</html>
```

- A responsive website design should have at least 3 layouts for different browser widths.
  - **Small:** under 600px. This is how content will look on most phones.
  - **Medium:** 600px – 900px. This is how content will look on most tablets, some large phones, and small netbook-type computers.
  - **Large:** over 900px. This is how content will look on most personal computers.
- Each of these layouts should include the same text and graphical elements, but each should be designed to best display that content based on the user's device.
- Scaling down the page to fit on smaller screen sizes will make the content unreadable, but if you scale the content relative to one another and switch to 1 column it makes it much more readable.

- Mobile browsers ○
  - iOS Safari (3.2+)
  - Android Browser (2.1+)
- Desktop browsers
  - Internet Explorer (9+)
  - Firefox (3.5+)
  - Chrome (4+)
  - Safari (4+)
- Full List http://caniuse.com/#feat=css-mediaqueries

- Mobile-first approach for other browsers
  - Default CSS = single column layout
  - Introduce additional complexity inside media queries (unsupported browsers will just ignore this)
- Respond.js - solid but limiting (no support for device-width, device-height,orientation, aspect-ratio, color, monochrome or resolution)
- CSS3-MediaQueries-js - more supported features but slow to load
- Conditional IE Style Sheets
  - Your media queries are simple enough to include in a single style sheet;
  - You do not have to support more legacy desktop browsers.

- one content site to manage
  - content is always current
  - less Maintenance
- development costs are low(er)
- adjusts to different width/sizes
- device OS independent
  - Smartphone, Tablet, iPhone, Android, etc
- clear need for an alternative to app development

THANK YOU