CM0133 Internet Computing

5. CSS Based Layouts

Recap - Styles

- A style is a set of formatting instructions that can be applied to a piece of text.
- Styles can be defined
  1. Within a single HTML tag – Inline styles (redundant inside documents)
  2. In the **<head>** section, and applied to the whole document – Global styles (redundant across documents)
  3. In external files, and can be applied to any document by including the URI of the file – Stylesheets (best reuse)
  4. Any combination is possible

Recap - Style rules

- Some properties can be given multiple values
  - The browser first looks for the "**Book Antiqua**" font
  - If this is not on the system, it looks for the **Times** font
  - Last resort: the browser uses the generic **serif** font

```
body {
 background-color: lightgreen;
}
h1 {
 color: lightgreen;
 background-color: blue;
 font-family: "Book Antiqua", Times, serif;
 border: thick groove #9baab2;
}
```

Recap - Cascading stylesheets

- Multiple stylesheets can be included in a document
- Styles defined in the first stylesheet are overridden by corresponding styles defined in the second stylesheet
  - the stylesheets are said to cascade
- Example
  - `mainstyles.css` – the company's stylesheet
  - `deptstyles.css` – the department's stylesheet
  - `mystyles.css` – the user's stylesheet
- If the stylesheets are included in this order, the user's style definitions will override the department styles, which in turn will override the company styles

Recap Inheritance

- Some values are inherited by the children of an element in the document tree, as described above. Each property defines whether it is inherited or not.

  Suppose there is an H1 element with an emphasizing element (EM) inside:

```
<H1>The headline <EM>is</EM> important!</H1>
```

- If no color has been assigned to the EM element, the emphasized "is" will inherit the color of the parent element, so if H1 has the color blue, the EM element will likewise be in blue.
- More details about how it works:

  http://www.w3.org/TR/CSS21/cascade.html

Objectives

- Today we look at possibilities design and implement a page layout with CSS.
- We will look at the Box Model and its properties.
- We will then review code examples of page layouts.
- We will look at general design principles for web pages. Most of these example as based on Joel Sklar's book “Principles of Web Design” (see literature slide).

CSS Versions

- CSS Level 1 was developed 1996 to complement HTML
- CSS Level 2 was published 1998 and allowed CSS based Layouts that previously were only possible with frames or tables.
- As with HTML there have been revisions and changes. The latest version is CSS 2.1 and is documented at http://www.w3.org/TR/CSS21/about.html
- The next version CSS 3.0 is already in preparation. Keep your eyes open for updates at http://www.w3.org

Html Tags = Containers + Hierarchy

```
<html lang="en-GB" xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-GB">
  <head profile="http://dublincore.org/documents/dcq-html/">
  <body class="hp3 glow-gecko">
    <div id="blq-accesslinks">
    <div id="blq-main">
      <div id="blq-banner">
      <div id="blq-content">
        <div id="hpOptionsBar">
        <div id="hpEditYourHomePage" style="display: none;">
        <form id="local" method="get" action="/home/d/" style="display: none;">
        <div id="hpFeatureBox">
        <div class="clearLeft hpColContainer">
          <div id="hpColOne" class="hpCol hpCol-first">
            <div id="a" class="hpMod altcolour2 ">
              <h2>
                <a href="/go/homepage/d/int/news/heading/-/news/">News</a>
              </h2>
              <div class="utils">
              <div class="hpSet" style="">
            </div>
            <div id="b" class="hpMod altcolour2 ">
              <h2>
                <a href="/go/homepage/d/int/sport/heading/-/news/sport/">Sport</a>
```

To create page layouts you need to interpret html as containers rather than tags. Html elements are boxes that can contain contents.

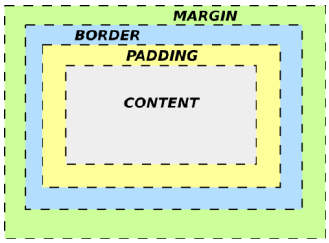
Another important aspect is that Html elements are organized in a hierarchical tree structure – the document object model.

The Box Model

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.
- The box model allows us to place a border around elements and space elements in relation to other elements.
- Not all browsers implement the Box Model according to specification, i.e W3C

The Box Model

- **Margin** - Clears an area around the border. The margin does not have a background color, and it is completely transparent
- **Border** - A border that lies around the padding and content. The border is affected by the background color of the box
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box
- **Content** - The content of the box, where text and images appear



In order to set the width and height of an element correctly in all browsers, you need to know how the box model works!

Width and Height of an Element

- Important: When you specify the width and height properties of an element with CSS, you are just setting the width and height of the content area. To know the full size of the element, you must also add the padding, border and margin.

- The total width of an element specified below is 300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

Let's do the math:

250px (width)

+ 20px (left and right padding)

+ 10px (left and right border)

+ 20px (left and right margin)

= 300px

Total size of an Element

- The **total width** of an element should always be calculated like this:

  Total element width = width + left padding + right padding + left border + right border + left margin + right margin

- The **total height** of an element should always be calculated like this:

  Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

The Box Model Error

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html>
<head>
<style type="text/css">
div.ex
{
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
}
</style>
</head>
```

If you tested the previous example in Internet Explorer, you saw that the total width was not exactly 300px. IE includes padding and border in the width, when the width property is set, unless a DOCTYPE is declared. To fix this problem, just add a DOCTYPE to the code.

CSS Border

CSS:

```
p.one {
    border-style:solid;
    border-width:5px;
}
p.two  {
    border-style:solid;
    border-width:medium;
}
p.three {
    border-style:solid;
    border-width:1px;
}
```

HTML:

```
<p class="one">Some text.</p>
<p class="two">Some text.</p>
<p class="three">Some text.</p>
```

Some text.

Some text.

Some text.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

CSS Margin

- The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.
- The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

```
margin: top right bottom left;
```

CSS Padding

- The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.
- The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

```
Padding: 1em;          // Top/Left/Right/Bottom = 1
Padding: 1em 2em;      // Top/Bottom = 1 , Left/Right = 2
Padding: 1em 2em 3em;  // Top = 1, Left/Right = 2, Bottom =  3
Padding: 1em 2em 3em 4em;  /* Top = 1, Right= 2,
                              Bottom = 3, Left = 4 */
```


CSS Units

- Absolute Units
  - Generally to avoid because they do not scale. To be preferred when you know the exact measurements of the destination medium (screens can vary in resolution!)

    p {margin: 1.25in;}
- Relative Units
  - Designed to build scalable pages. CSS 2 designers (Hakon Lie and Bert Bos) recommend to always use relative sizes to set font sizes in your web page. This ensures that your type sizes are properly displayed relative to each other.

    p {margin: 1.25em;}
- Percentage
  - Are always relative to another unit. Child elements inherit the percentage values of their parents.

    p {font-size: 12pt}

    p b {font-size: 125%} results in <b> being displayed in 15pts.

CSS Units - Overview

Unit	Abbreviation	Description
Absolute Units		
Centimetre	cm	Standard metric centimetre
Inch	in	Standard U.S. inch
Millimetre	mm	Standard metric millimetre
Pica	pc	Standard publishing unit equal to 12pt
Point	pt	Standard publishing unit with 72 points in an each
Relative Units		
Em	em	The width of the capital M in the current font, usually the same as the font size
Ex	ex	The height of the letter x in the current font
Pixel	px	The size of a pixel on the current monitor
Percentage	Eg: 150 %	Sets a font size relative to the base font size, e.g. 150% is 1,5 times the base font size

Example 1

- This example shall guide you step by step to create a page layout with CSS.
- First we create a two column layout.
- Use tags as containers for “navigation” and “content”.
- For the “navigation” we use a list whose id will be used as a selector for the CSS definition. The div for the contents als gets an id so that we can easily assign a style to it.
- HTML elements that are positioned with float have to be coded before any other elements.
- Use a DOCTYPE declaration to avoid troubles in display with browsers that do not adhere to the Standard.

Look also at the code handout !

Example 1 - 3 Column Layout

- To achieve a 3 Column Layout with an additional area at the right to variable content area you just need to add a further box with the CSS property float:right and set the width of the element
- To place several boxes with float next to each other omit a definition of float and width for one of the boxes (content). All remaining boxes have to be placed in the html code prior to the 'definitionless box', so that they can claim their defined width first.

Look also at the code handout !

Example 1 - 3 Column Layout

- Now we set margin-left and margin right so that the “content” box does not float around the neighbouring boxes.
- The values set for the margins need to be calculated as sum of the width of the boxes plus the space you want between the boxes. Watch out: Length measures in em depend on the font-size of the used html element !
  - Assignment: Experiments with other units and observe the difference.
- With min-width we can avoid that our boxes 'collapse' when we change the size of the browser window.
- Note: The containers do not influence each other regards their height. The container paradigm requires you to stop thinking in “rows and columns” as you would with tables.
- Note in this example also how font-size is inherited !

Look also at the code handout !

Example 1 - Add a Header/Footer

- Header
  - We use a <h1> element to introduce a header.
  - We add the header definition at the beginning of the Stylesheet and also move the <h1> element to the top of the html code.
  - Hierarchically next are <h2> elements which are used as headers for the containers.
- Footer
  - To override the previously set float property that was used to create the columns we use now the CSS-property clear to re-establish a nice page flow.

Look also at the code handout !

Example 2 - Centred Design

```
body {
   margin:50px 0px; padding:0px;
   text-align:center;
}

#Content {
   width:500px;
   margin:0px auto;
   text-align:left;
   padding:15px;
   border:1px dashed #333;
   background-color:#eee;
}
```

Source: http://www.bluerobot.com/web/css/center1.html

Web Design Principles

- Design for the **computer medium**
- Create a **unified site design**
- Design for the **user**
- Design for the **screen**

Where to Start ?

- The best way to learn about web page designs is to look at a lot of them.
- Consider our own school home page: http://www.cs.cf.ac.uk/
- Ask yourself some basic questions:
  - How does it *appeal* to you?
  - Who is the intended *audience*?
  - Can you *find* what you need?
  - Is it easy to *navigate*?

Design for the Computer Medium

- Craft the look and feel
  - Images and colors convey emotion
- Make your design portable
  - Be mindful of browser-specific features
- Plan for clear presentation and easy access to your information

Design for Portability

- Your Web site design must be portable and accessible across different browsers, operating systems, and computer platforms
- You must always remember to test your work even when you feel confident of your results

Design for Clear Presentation

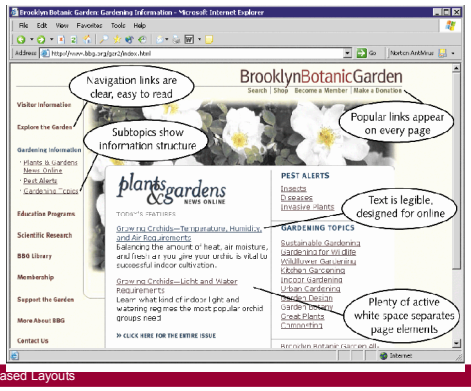
- Old school
  - You are paying for every square centimeter on the screen
  - Get as much content in as possible
- New Thinking
  - Allow for "white space"
  - Friendly sites (kinder, gentler)
  - Ok to scroll (longer pages becoming more acceptable)

Old School Petco

Link to New Petco

Old School bbg.org

Link to new bbg.org



Create a Unified Site Design

- Plan the unifying themes and structures
- Create smooth transitions
- Use a grid to provide visual structure
  - Design with tables (Old School)
  - Design with CSS (New Thinking)
- Use active white space

Link to new CDC site

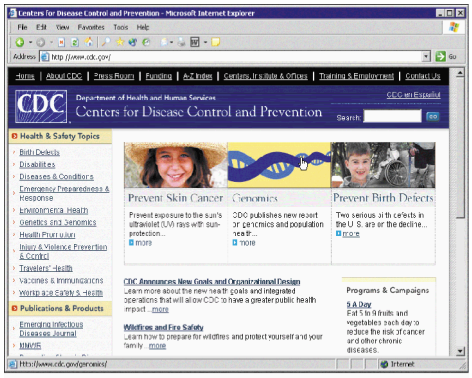


Figure 2-5 Centers for Disease Control main page

Harsh Transitions vs Smooth Transitions

- Plan smooth transitions
  - Plan to create a unified look
  - Reinforce the identifying elements
  - Avoid random, jarring changes in format
- Harsh transition can be disrupting
  - But, remember your audience (what about kid's pages) - sometimes a harsh transition is what you want

Use a Grid to Provide Visual Structure

- The **grid** is a conceptual layout device that organizes content into columns and rows
- A grid provides visual consistency
- Use CSS !

Figure 2-9 Grid provides visual structure

Use Active White Space

- Use white space deliberately in your design
- Good use of white space guides the reader and defines the areas of your page
- Active white space is an integral part of your design that structures and separates content
- http://www.bbc.co.uk/
  - (view source for layout)

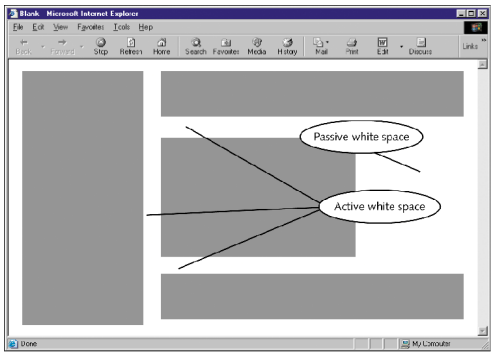


Figure 2-10 Areas of active and passive white space

Link to new Csmonitor site

Use Graphics for Navigation and Linking

- *Standardize* your navigation graphics
- Provide *predictable* and *easily understandable* navigation cues for the user
- *Repeat* graphics to minimize download time
- Use *consistent placement* and design of navigation graphics to reassure the user

Clear Indication of Present Location

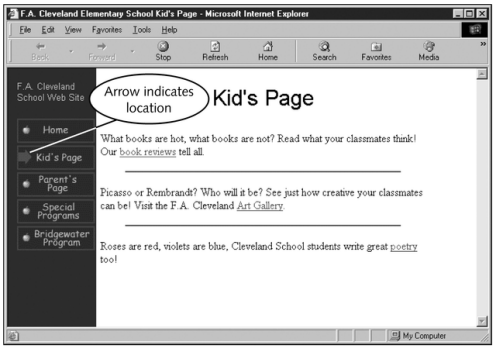


Figure 4-20 Navigation graphic indicates location

Graphics for Navigation

Icons for Navigation

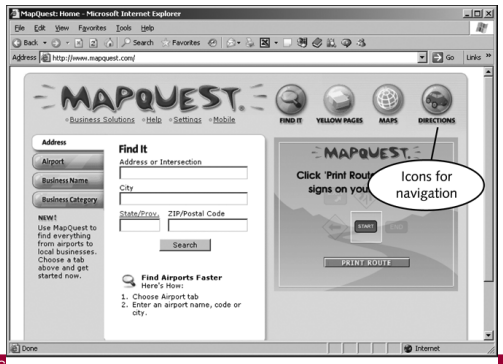


Figure 4-17 Icons for navigation

Design for the User

- Keep your design efforts centered solely around your user
- Design for interaction
- Design for location
- Guide the user's eye
- Decide whether the user will read or scan

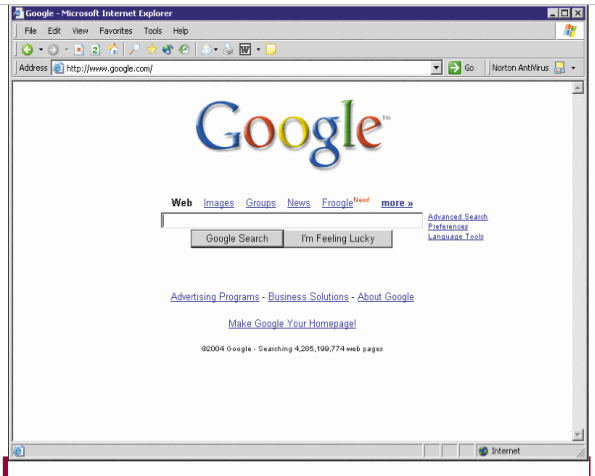


Figure 2-12 Google's simple, task-oriented design

Figure 2-13 A hectic design for E! online's audience

Figure 2-14 A paper-based design for the *Atlantic*'s audience

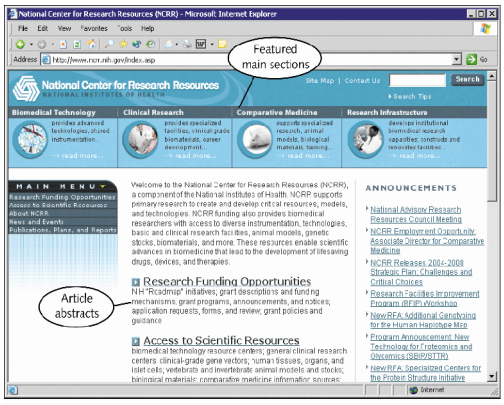


Figure 2-15 Page designed for scanning

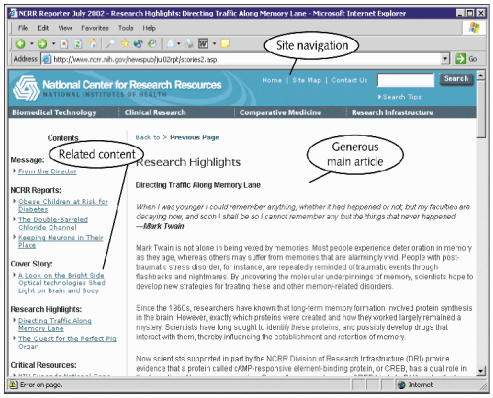


Figure 2-16 Page designed for reading

Basic Layout Design

- "Area of Importance" or "Wheel" Design
- "Paper-based Reading" or "Scan" Design
- "Circular" Design

Area of Importance

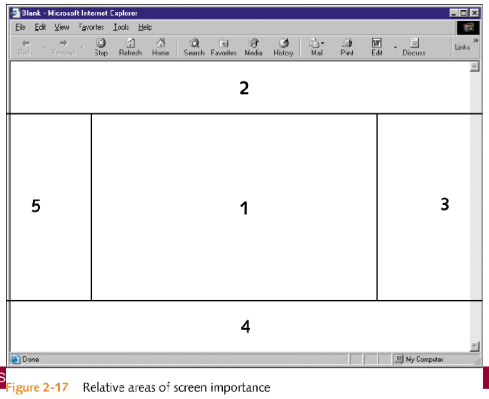
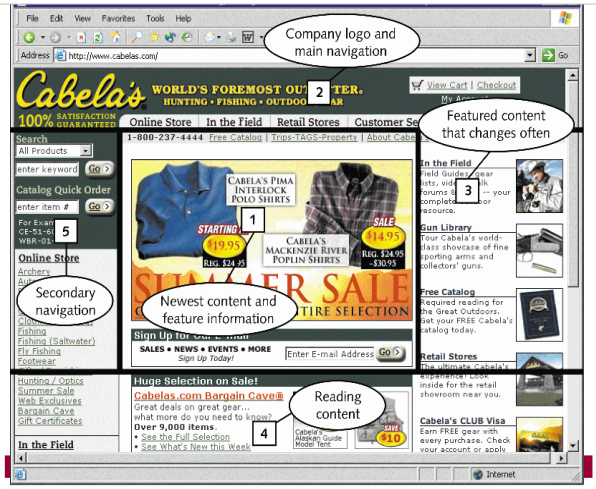


Figure 2-17 Relative areas of screen importance



Paper Based Reading or Scan Design

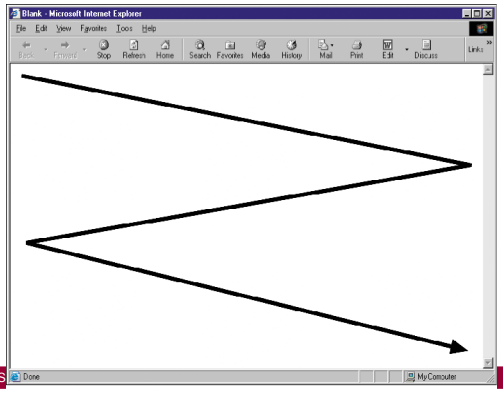


Figure 2-19 Paper-based reading pattern

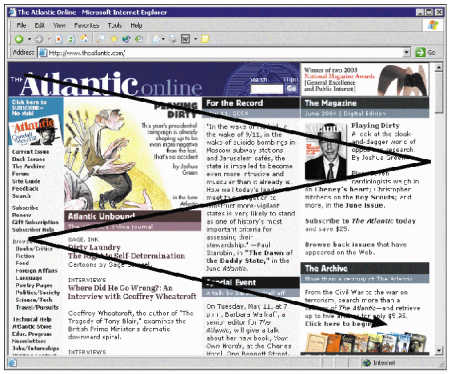


Figure 2-20 Paper-based reading pattern applied to theatlantic.com Web page

Circular Design

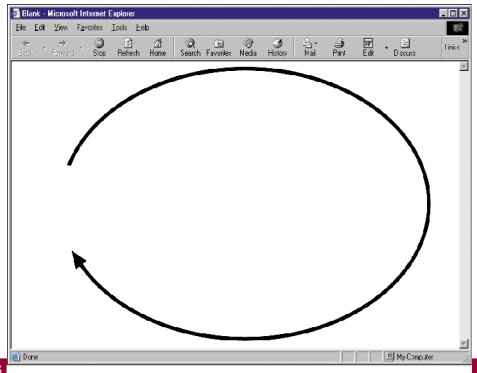


Figure 2-21 Screen-based viewing pattern

Figure 2-22 Screen-based reading pattern for the UN Web site

Design for Accessibility

- Develop Web pages that remain accessible despite any physical, sensory, and cognitive disabilities
- Developing accessible content naturally leads to creating good design.
- Follow W3 Accessibility Initiative guidelines at
  - http://www.w3.org/WAI

Design for the Screen

- The computer display is different than print
  - The display is landscape-oriented
    - Except: UMDs and eBooks
  - Colors and contrasts are different

Coding for Multiple Screen Resolutions

- Flexible Design (our Example 1)
  - Adapt with fixed boxes and flexible boxes
  - Challenge to keep legible and navigable
  - http://www.amazon.co.uk/
- Centred Design (our Example 2)
  - Designed for base resolution, e.g. 800x600, Active whitespace left and right grows on higher resolutions
  - http://www.bbc.co.uk/
- Fixed Design
  - Consider that they do not scale.
  - have become less frequent

Design for the User

- Keep a flat hierarchy
- Provide plenty of linking options
- Provide location information
- Use plenty of textual links
- Don't overload the user with too much content
- Design for accessibility

Literature

- Sklar, Joel (2006) Principles of Web Design,3rd Edition Web Warrior Series, Thomson Course Technology

  Yes, it is available in Trevithick library !

  associated website with the newer 4th edition: http://oc.course.com/webwarrior/sklar4/
- W3C: http://www.w3.org/standards/webdesign/htmlcss
- W3 Schools : http://www.w3schools.com/css/
- As always – google, yahoo search, etc.