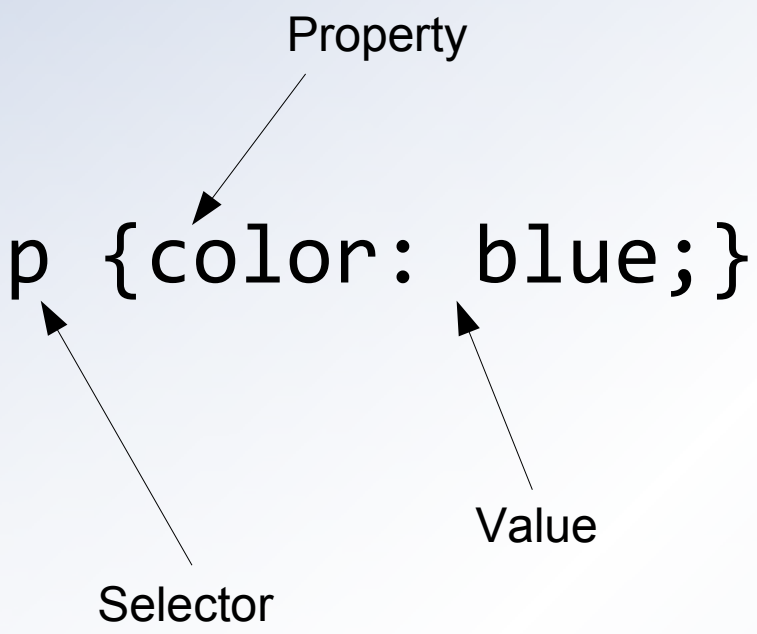
PIC 40A

Lecture 6: CSS properties

CSS Properties

Recall a typical CSS rule:



Color Properties

Font text color, and default border color is specified with property `color`

background color with property `background-color`

Example:

```
p  {
     background-color: blue;
     color: white;
   }
```

Predefined colors

```
Black = #000000        Green = #008000
Silver = #C0C0C0       Lime = #00FF00
Gray = #808080         Olive = #808000
White = #FFFFFF        Yellow = #FFFF00
Maroon = #800000       Navy = #000080
Red = #FF0000            Blue = #0000FF
Purple = #800080       Teal = #008080
Fuchsia = #FF00FF      Aqua = #00FFFF
```

Ways to specify a color value

```
color: #0000FF;     /* 3 2-digit hexadecimals*/
color: #00F;    /* 3 1-digit hexadecimals*/
                /* #ABC = #AABBCC */

color: rgb(0,0,255);   /* 3 8-bit values 0-255 */
                    /* 0 = empty, 255 = full */

color: rgb(0,0,100%);  /* red, green, blue %'s */
                    /* 0% = empty,100%=full */
```

Examples:

```
p { color: #CCCCCC;}
h1 { color: rgb(100,100,100);}
div { color: rgb (20%, 50%,0);}
```

Font properties

font-family

used to specify font

Example:

p { font-family: verdana;}

Some popular font families are: arial, helvetica, verdana, times new roman, courier etc.

Example:

div p { font-family: verdana, arial, courier;}

The browser will use the first font that it understands

The five generic font families

- serif

typeface has decorative **serifs** (slab-like letter strokes) on the ends of certain letters (better for print)
eg Times New Roman

- sans-serif

has straight letters with no serifs (better for screen)
eg Arial

- monospace

all chars have the same width (better for code)
eg Courier New

- cursive

emulates a script or handwritten appearance
eg Comic Sans

- fantasy

purely decorative, for headlines
eg **Impact**

How to use a generic font family

The best way to use the font-family is to specify your fonts in order of preference:

```
p { font-family: Arial, Helvetica, Futura, sans-serif;}
```

Last one should be a generic font family.

Font properties

font-size

Example:

```
p {font-size: 12px; font-family: Arial, sans-serif;};
```

Size in absolute terms e.g. 20px or 20pt. px is preferred over pt since pt can look different on different operating systems. pt is generally used for print media and px used for web.

Relative terms e.g. small, large, x-large etc. These sizes are in relation to default font size of users browser which is medium.

larger and smaller are defined in relation to parent elements font size.

em is the size of the parent elements font size. So .5em would be half the fo size of the parent.

Other font properties and values

- **`font-style`** (controls posture of font)
  - `normal`, `italic`, `oblique`
- **`font-weight`** (controls intensity of font)
  - `normal`, `bold`, `lighter`, `100`, `200`, etc…, `900`

Text properties

Most important text property is probably text-decoration.

Example:
CSS:

```
p span { text-decoration: underline;}
```

XHTML:

```
<p>Is that <span>really</span> what she said?</p>
```

Browser displays:

Is that <u>really</u> what she said?

Some values for text-decoration are: `underline`, `overline`, `line-through`, `none`.

Text properties

For your reference only. You do not need to memorize these ones.

- **`text-transform`**
  - val: `capitalize`, `lowercase`, `uppercase`, `none`

- **`text-indent`**

  To indent the first line of a paragraph for example
  - val: How much you want to indent example: `30px`

- **`text-align`**
  - val: `left`, `right`, `center`, `justify`

- **`vertical-align`**
  - val: `top`, `middle`, `bottom` or amount which is positive or negative length.

List CSS Properties

- **`list-style-type`**

Controls what the bullet points look like.

–val: `disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha upper-alpha`

Example:

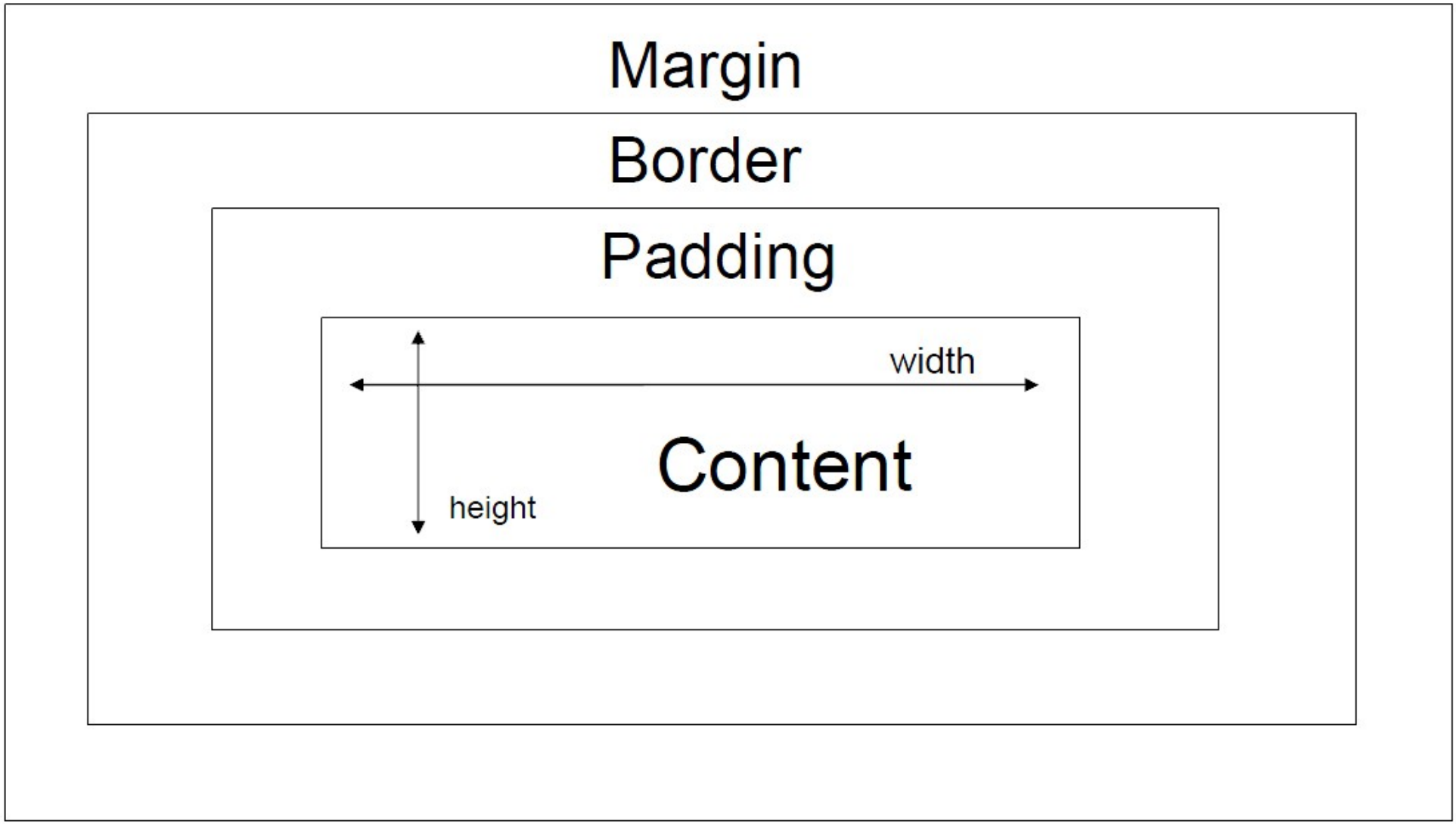
```
ol {list-style-type: circle;}
ol ol {list-style-type: square;}
```

- **`list-style-image`**

Create your own bullet image.

```
ul {list-style-image:url('sqpurple.gif');}
```

CSS Box Model



This is the anatomy of an element's appearance

Margin and padding

- `padding-top` (same for **margin**, **border**)
- `padding-bottom`
- `padding-left`
- `padding-right`

Example:

```
div { padding-top: 20px; }
```

It is possible to set all padding (or margin and) with single command:

```
p {padding:25px 50px 75px 100px;}
```

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

It is most readable to use padding property (or margin property) by itself when you want all sides to have same padding (margin).

```
p {padding: 0px;}
```

CSS Box Model Properties

This is for your reference. I only expect you to remember width.

- **`width, min-width, max-width`**
- **`height, min-height, max-height`**
- **`overflow`**
  - `scroll, hidden, visible, auto`

More CSS Box Model Properties

- **`border-width`**
  -Thickness of the border

- **`border-style`**
  –`none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`

- **`border-color`**

- **`border`** (to specify all 3 properties at once)

```
div {border:1px solid blue;}
```

`Display` property

- **`display`**

–allows the programmer to decide how an element should be rendered.

–Commonly used with pseudo-class selector `hover` to make elements appear or disappear when a mouse hovers over

Important values of `display` property

- `none` - The element is not displayed at all.
- `block` - Element is displayed as a block element.
- `inline` - Element is displayed as an inline element.

Example: Drop down menu. See examples page.

Document flow

- Normally, elements are positioned in the order in which they appear in an XHTML document

- XHTML document flow:
  - inline elements appear in a flow from left to right within enclosing block
  - block level elements appear in a flow as a stack of blocks from top to bottom

`Position` property

values:

- `absolute`

–relative to the enclosing block element
–takes element out of normal flow

- `relative`

–relative to element's normal position in flow

- `fixed`

–relative to the browser window
–Element remains at the specified position regardless of scrolling

Note on `position` property

The `position` property can be followed by the following offset properties:

```
top
bottom
left
right
```

Example

```
position: absolute;
top: 30px;
left: 10px;
```

This rule positions an element 30 px down from the top and 10px from the left of the containing block element.

Overlapping elements

If elements overlap, you can control their stacking order using the property

`z-index`

Example:

```
#under{z-index:1;}
#over{z-index:3;}
```

Element with the highest z-index value is on top.

Floating elements

- **Floating** means moving an element to one side of the screen while *following* content flows around it.

- A floated element will move as far to the left or right as it can in the containing element.

- If an image is floated to the right, following text flows around it, to the left.

- Use the **`float`** property
  - val: `left, right`

Reference slides

Following slides are for your reference. I will not ask you to memorize their content. If I have a question in the exam that refers to the material from the following slides, I will give you the properties/values needed to do the problem.

Resources for properties and values

See our class website for many examples

Properties and values with explanations and examples:

http://www.pageresource.com/dhtml/cssprops.htm

Official reference, but harder to understand:

http://www.w3.org/TR/CSS2/propidx.html

Units of measurements for web pages

• Measurement Values • Unit	• Description
• %	• percentage
• in	• inch
• cm	• centimeter
• mm	• millimeter
• em	• 1em is equal to the current font size. 2em means 2 times the size of the current font. E.g., if an element is displayed with a font of 12 pt, then '2em' is 24 pt. The 'em' is a very useful unit in CSS, since it can adapt automatically to the font that the reader uses
• ex	• one ex is the x-height of a font (x-height is usually about half the font-size)
• pt	• point (1 pt is the same as 1/72 inch)
• pc	• pica (1 pc is the same as 12 points)
• px	• pixels (a dot on the computer screen)

Notes on the `font` propert

`font` (Shorthand property for all font properties)

```
p { font:italic bold 12px  Georgia, serif; }
```

•Must include `font-size` and `font-family` as the last 2 properties in the list (in that order)

•Optional properties `font-style`, `font-variant`, and `font-weight` may appear in any order but must be before `font-size` and `font-family`

•Omitted properties in the list are reset to some initial value by the browser (*Beware of this!*).

Font Examples

```
h1{ font-size: 1.75em;
    font-family: sans-serif;
}

h2{ font: italic
    font-weight: bold
    font-size: 120%
    font-family: cursive;
}

p {
font-style: italic;
font-variant: normal;
font-weight: normal;
font-size: large;
font-family: serif;
}
```

More Text Properties

- **`letter-spacing`**
  - -How far apart letters are from each other
- **`word-spacing`**
  - -How far apart words are from each other
- **`line-height`**
  - -How far apart lines are from each other

All values in units of length.

Text Examples

```
p.mistake{ text-decoration: line-though;
           text-indent: 3em;
}

p.over{ text-decoration: overline;
        white-space: nowrap;
}

p.important{ text-decoration: underline;
             text-transform: uppercase;
             text-align: justify;
}
p.loud{ text-decoration: none;
        letter-spacing: 1.5ex;
}
```

Table CSS Properties

- **`caption-side`** (for `caption` elements)
  - val: `top`, `bottom`
- **`table-layout`**
  - val: `auto`, `fixed` (See next slide for explanation.)
- **`empty-cells`**
  - val: `show`, `hide`

CSS table-layout values

`auto` (default)

–Column width set by the widest unbreakable content in the cells

–Slow since it needs to read through all table content before determining the final layout

`fixed`

–Horizontal layout only depends on the width of the table and columns, not the contents of the cells

–Faster than automatic layout since the browser can start displaying the table after first row has been received