4.1: Styling Page Sections

- 4.1: Styling Page Sections
- 4.2: Introduction to Layout
- 4.3: Floating Elements
- 4.4: Sizing and Positioning
Motivation for page sections

- want to be able to style individual elements, groups of elements, sections of text or of the page
- (later) want to create complex page layouts

Sections of a page: `<div>` (4.1.1)

*a section or division of your HTML page (block)*

```
<div class="standout">
  <h2>Spatula City! Spatula City!</h2>
  <p class="special">See our spectacular spatula specials!</p>
  <p>We'll beat any advertised price!</p>
</div>
```

- a tag used to indicate a logical section or area of a page
- has no appearance by default, but you can apply styles to it
Inline sections: `<span>` (4.1.2)

An inline element used purely as a range for applying styles

```html
<h2>Spatula City! Spatula City!</h2>
<p>See our <span class="special">spectacular</span> spatula specials!</p>
<p>We'll beat <span class="standout">any advertised price</span>!</p>
```

**Spatula City! Spatula City!**

See our **spectacular** spatula specials!

We'll beat **any advertised price**!

- has no onscreen appearance, but you can apply a style or ID to it, which will be applied to the text inside the `span`

CSS context selectors (4.1.3)

```css
selector1 selector2 { properties }
```

- applies the given properties to `selector2` only if it is inside a `selector1` on the page

```css
selector1 > selector2 { properties }
```

- applies the given properties to `selector2` only if it is *directly* inside a `selector1` on the page (`selector1` tag is immediately inside `selector2` with no tags in between)
Context selector example

```html
<p>Shop at <strong>Hardwick's Hardware</strong>...</p>
<ul>
  <li>The <strong>best</strong> prices in town!</li>
  <li>Act while supplies last!</li>
</ul>
```

```css
li strong { text-decoration: underline; }
```

Shop at Hardwick's Hardware...

- The **best** prices in town!
- Act while supplies last!

---

More complex example

```html
<div id="ad">
  <p>Shop at <strong>Hardwick's Hardware</strong>...</p>
  <ul>
    <li class="important">The <strong>best</strong> prices in town!</li>
    <li>Act <strong>while supplies last!</strong></li>
  </ul>
</div>
```

```css
#ad li.important strong { text-decoration: underline; }
```

Shop at Hardwick's Hardware...

- The **best** prices in town!
- Act **while supplies last!**
The CSS Box Model (4.2.1)

- for layout purposes, every element is composed of:
  - the actual element's **content**
  - a **border** around the element
  - **padding** between the content and the border *(inside)*
  - a **margin** between the border and other content *(outside)*
- width = content width + L/R padding + L/R border + L/R margin
- height = content height + T/B padding + T/B border + T/B margin
  - IE6 doesn't do this right
Document flow - block elements

Document flow - inline elements
CSS properties for borders

```
h2 { border: 5px solid red; }
```

This is a heading.

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border</td>
<td>thickness/style/size of border on all 4 sides</td>
</tr>
</tbody>
</table>

- **thickness** (specified in px, pt, em, or thin, medium, thick)
- **style** (none, hidden, dotted, dashed, double, groove, inset, outset, ridge, solid)
- **color** (specified as seen previously for text and background colors)
More border properties

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>border-color, border-width, border-style</td>
<td>specific properties of border on all 4 sides</td>
</tr>
<tr>
<td>border-bottom, border-left, border-right, border-top</td>
<td>all properties of border on a particular side</td>
</tr>
<tr>
<td>border-bottom-color, border-bottom-style, border-bottom-width, border-left-color, border-left-style, border-left-width, border-right-color, border-right-style, border-right-width, border-top-color, border-top-style, border-top-width</td>
<td>properties of border on a particular side</td>
</tr>
</tbody>
</table>

Complete list of border properties

Border example 2

```css
h2 {
    border-left: thick dotted #CC0088;
    border-bottom-color: rgb(0, 128, 128);
    border-bottom-style: double;
}
```

This is a heading.

- each side's border properties can be set individually
- if you omit some properties, they receive default values (e.g. `border-bottom-width` above)
CSS properties for padding

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>padding</td>
<td>padding on all 4 sides</td>
</tr>
<tr>
<td>padding-bottom</td>
<td>padding on bottom side only</td>
</tr>
<tr>
<td>padding-left</td>
<td>padding on left side only</td>
</tr>
<tr>
<td>padding-right</td>
<td>padding on right side only</td>
</tr>
<tr>
<td>padding-top</td>
<td>padding on top side only</td>
</tr>
</tbody>
</table>

Complete list of padding properties

Padding example 1

```css
p { padding: 20px; border: 3px solid black; }
h2 { padding: 0px; background-color: yellow; }
```

This is the first paragraph

This is the second paragraph

This is a heading
Padding example 2

```css
p {
  padding-left: 200px; padding-top: 30px;
  background-color: fuchsia;
}
```

This is the first paragraph

This is the second paragraph

- each side's padding can be set individually
- notice that padding shares the background color of the element

CSS properties for margins

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>margin</td>
<td>margin on all 4 sides</td>
</tr>
<tr>
<td>margin-bottom</td>
<td>margin on bottom side only</td>
</tr>
<tr>
<td>margin-left</td>
<td>margin on left side only</td>
</tr>
<tr>
<td>margin-right</td>
<td>margin on right side only</td>
</tr>
<tr>
<td>margin-top</td>
<td>margin on top side only</td>
</tr>
</tbody>
</table>

Complete list of margin properties
Margin example 1

```
p {  
  margin: 50px;  
  background-color: fuchsia;  
}
```

```
This is the first paragraph

This is the second paragraph
```

- notice that margins are always transparent
  (they don't contain the element's background color, etc.)

Margin example 2

```
p {  
  margin-left: 8em;  
  background-color: fuchsia;  
}
```

```
This is the first paragraph

This is the second paragraph
```

- each side's margin can be set individually
Free Web Building Tutorials

At W3Schools you will find all the Web-building tutorials you need, from basic HTML and XHT to advanced XML, XSL, Multimedia and WAP.

4.3: Floating Elements

- 4.1: Styling Page Sections
- 4.2: Introduction to Layout
- **4.3: Floating Elements**
- 4.4: Sizing and Positioning
CSS properties for **dimensions** (4.3, 4.4.1)

```
p { width: 350px; background-color: yellow; }
h2 { width: 50%; background-color: aqua; }
```

This paragraph uses the first style above.

**An h2 heading**

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width, height</td>
<td>how wide or tall to make this element (block elements only)</td>
</tr>
<tr>
<td>max-width, max-height, min-width, min-height</td>
<td>max/min size of this element in given dimension</td>
</tr>
</tbody>
</table>

---

**Centering a block element: auto margins**

```
p {
    margin-left: auto;
    margin-right: auto;
    width: 750px;
}
```

*Lorem ipsum* dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- works best if width is set (otherwise, may occupy entire width of page)
- to center inline elements within a block element, use `text-align: center;`
The CSS `float` property ([reference](#)) (4.3.1)

```
img.headericon {
  float: right;
  width: 130px;
}
```

**Borat** Sagdiyev (born July 30, 1972) is a fictional Kazakhstani journalist played by British-Jewish comedian Sacha Baron Cohen. He is the main character portrayed in the controversial and successful film *Borat: Cultural Learnings of America for Make Benefit Glorious ...*

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>side to hover on; can be left, right, or none (default)</td>
</tr>
</tbody>
</table>

- removed from normal document flow; underlying text wraps around as necessary

**Floating elements diagram**

![Floating elements diagram](image-url)
Common float bug: missing width

I am not floating, no width

I am floating right, no width

I am not floating, 45% width

I am floating right, 45% width

- often floating block elements must have a width property value
  - if no width is specified, the floating element may occupy 100% of the page width, so no content can wrap around it

The clear property (4.3.2)

```css
p { background-color: fuchsia; }

h2 { clear: right; background-color: yellow; }
```

Homestar Runner is a Flash animated Internet cartoon. It mixes surreal humour with references to 1980s and 1990s pop culture, notably video games, classic television and popular music.

My Homestar Runner Fan Site

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear</td>
<td>disallows floating elements from overlapping this element; can be left, right, or none (default)</td>
</tr>
</tbody>
</table>
4.4: Sizing and Positioning

- 4.1: Styling Page Sections
- 4.2: Introduction to Layout
- 4.3: Floating Elements
- 4.4: Sizing and Positioning
The **position** property (examples) (4.4.2)

```css
div#ad {
    position: fixed;
    right: 10%;
    top: 45%;
}
```

<table>
<thead>
<tr>
<th>property</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>static</td>
<td>default position</td>
</tr>
<tr>
<td></td>
<td>relative</td>
<td>offset from its normal static position</td>
</tr>
<tr>
<td></td>
<td>absolute</td>
<td>a fixed position <em>within its containing element</em></td>
</tr>
<tr>
<td></td>
<td>fixed</td>
<td>a fixed position <em>within the browser window</em></td>
</tr>
<tr>
<td>top, bottom, left, right</td>
<td>positions of box's corners</td>
<td></td>
</tr>
</tbody>
</table>

**Absolute positioning**

```css
#sidebar {
    position: absolute;
    left: 400px;
    top: 50px;
}
```

- removed from normal flow (like floating ones)
- positioned relative to the block element containing them (assuming that block also uses absolute or relative positioning)
- actual position determined by `top`, `bottom`, `left`, `right` values
- should often specify a `width` property as well
Relative positioning

- absolute-positioned elements are normally positioned at an offset from the corner of the overall web page
- to instead cause the absolute element to position itself relative to some other element's corner, wrap the absolute element in an element whose position is relative

Fixed positioning

- removed from normal flow (like floating ones)
- positioned relative to the browser window
  - even when the user scrolls the window, element will remain in the same place
Alignment vs. float vs. position

1. if possible, lay out an element by aligning its content
   - horizontal alignment: text-align
     ■ set this on a block element; it aligns the content within it (not the block element itself)
   - vertical alignment: vertical-align
     ■ set this on an inline element, and it aligns it vertically within its containing element
2. if alignment won't work, try floating the element
3. if floating won't work, try positioning the element
   - absolute/fixed positioning are a last resort and should not be overused

Details about inline boxes

- size properties (width, height, min-width, etc.) are ignored for inline boxes
- margin-top and margin-bottom are ignored, but margin-left and margin-right are not
- the containing block box's text-align property controls horizontal position of inline boxes within it
  - text-align does not align block boxes within the page
- each inline box's vertical-align property aligns it vertically within its block box
**The vertical-align property**

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertical-align</td>
<td>specifies where an inline element should be aligned vertically, with respect to other content on the same line within its block element's box</td>
</tr>
</tbody>
</table>

- can be top, middle, bottom, baseline (default), sub, super, text-top, text-bottom, or a length value or %
  - baseline means aligned with bottom of non-hanging letters

---

**vertical-align example**

```html
<p style="background-color: yellow;">
  <span style="vertical-align: top; border: 1px solid red;">
    Don't be sad! Turn that frown upside down!
  </span>
  <img src="images/sad.jpg" alt="sad" />
  <img style="vertical-align: bottom" src="images/smiley.jpg" alt="smile" />
  Smiling burns calories, you know.
  <img style="vertical-align: middle" src="images/puppy.jpg" alt="puppy" />
  Anyway, look at this cute puppy; isn't he adorable! So cheer up, and have a nice day. The End.
</p>
```

Output: Don't be sad! Turn that frown upside down! Smiling burns calories, you know. Anyway, look at this cute puppy; isn't he adorable! So cheer up, and have a nice day. The End.
Common bug: space under image

```html
<p style="background-color: red; padding: 0px; margin: 0px"> 
<img src="images/smiley.png" alt="smile" /> 
</p>
```

- red space under the image, despite padding and margin of 0
- this is because the image is vertically aligned to the baseline of the paragraph (not the same as the bottom)
- setting vertical-align to bottom fixes the problem (so does setting line-height to 0px)

The **z-index** property (4.4.3)

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z-index</td>
<td>sets which absolute positioned element will appear on top of another that occupies the same space</td>
</tr>
</tbody>
</table>

- higher z-index elements appears on top of lower ones
- can be auto (default) or a number
The **display** property (4.4.4)

```css
h2 { display: inline; background-color: yellow; }
```

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>sets the type of CSS box model an element is displayed with</td>
</tr>
</tbody>
</table>

- values: none, inline, block, run-in, compact, ...
- use sparingly, because it can radically alter the page layout

### Displaying block elements as inline

```html
<ul id="topmenu">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

```css
#topmenu li {
  display: inline;
  border: 2px solid gray;
  margin-right: 1em;
}
```

- lists and other block elements can be displayed inline
  - flow left-to-right on same line
  - width is determined by content (block elements are 100% of page width)
The **visibility** property

```css
p.secret {
    visibility: hidden;
}
```

<table>
<thead>
<tr>
<th>property</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>visibility</td>
<td>sets whether an element should be shown onscreen; can be <strong>visible</strong> (default) or <strong>hidden</strong></td>
</tr>
</tbody>
</table>

- hidden elements will still take up space onscreen, but will not be shown
  - to make it not take up any space, set `display` to `none` instead
- can be used to show/hide dynamic HTML content on the page in response to events