EECS1012

Net-centric Introduction to Computing

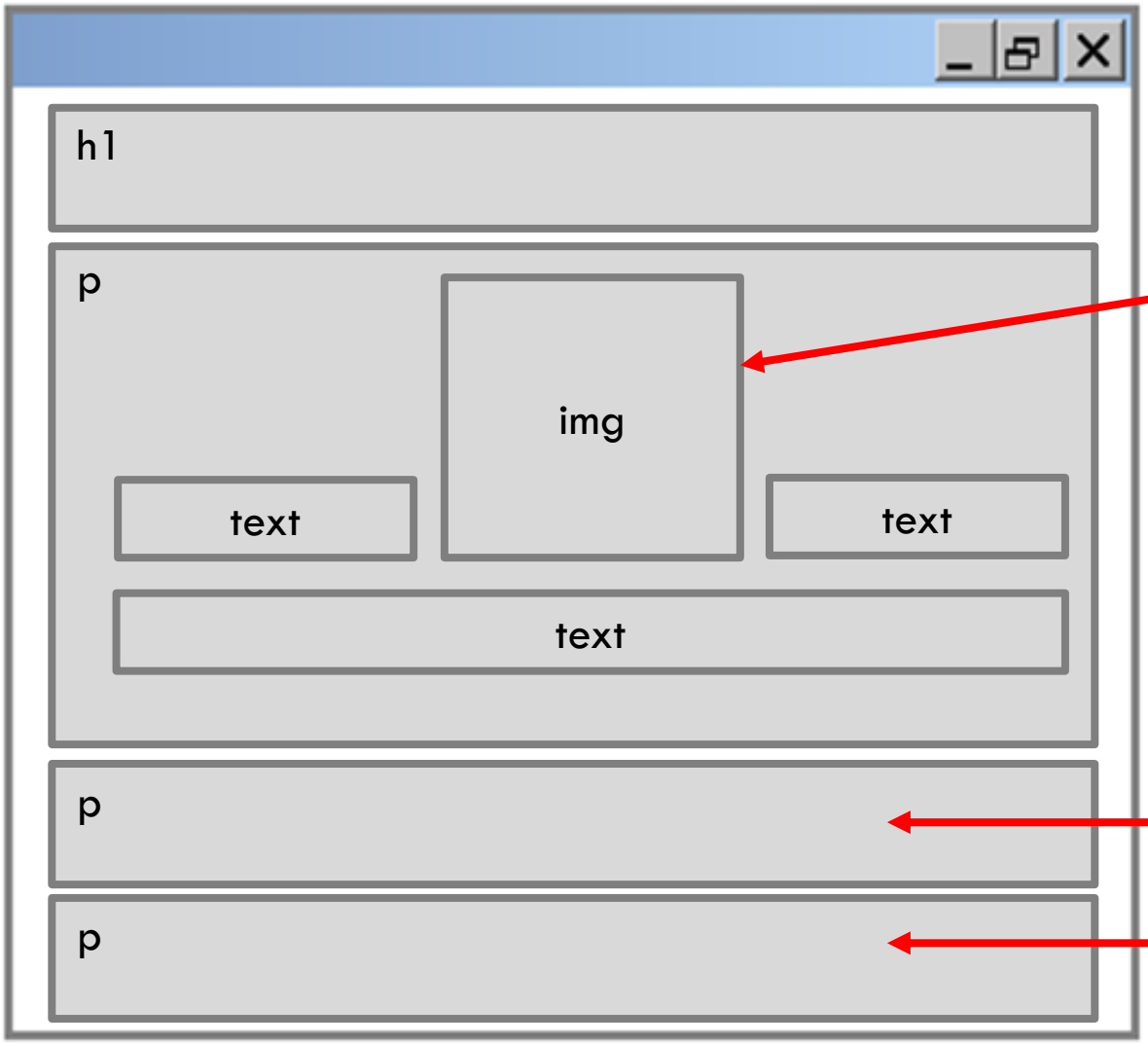
Lecture 5:
Yet more CSS (Float and Positioning)

Acknowledgements
Contents are adapted from web lectures for "Web Programming Step by Step", by M. Stepp, J. Miller, and V. Kirst.
Slides have been ported to PPT by Dr. Xenia Mountrouidou.
These slides have been edited for EECS1012, York University.

Floating Elements

Elements flow

Recall: <h1> and <p> are block elements. Text inside a paragraph are treated as inline content and elements.



Many times the default flow of the content in HTML is a bit ugly.

The default flow of block elements is to take up all the horizontal space.

Modifying flow

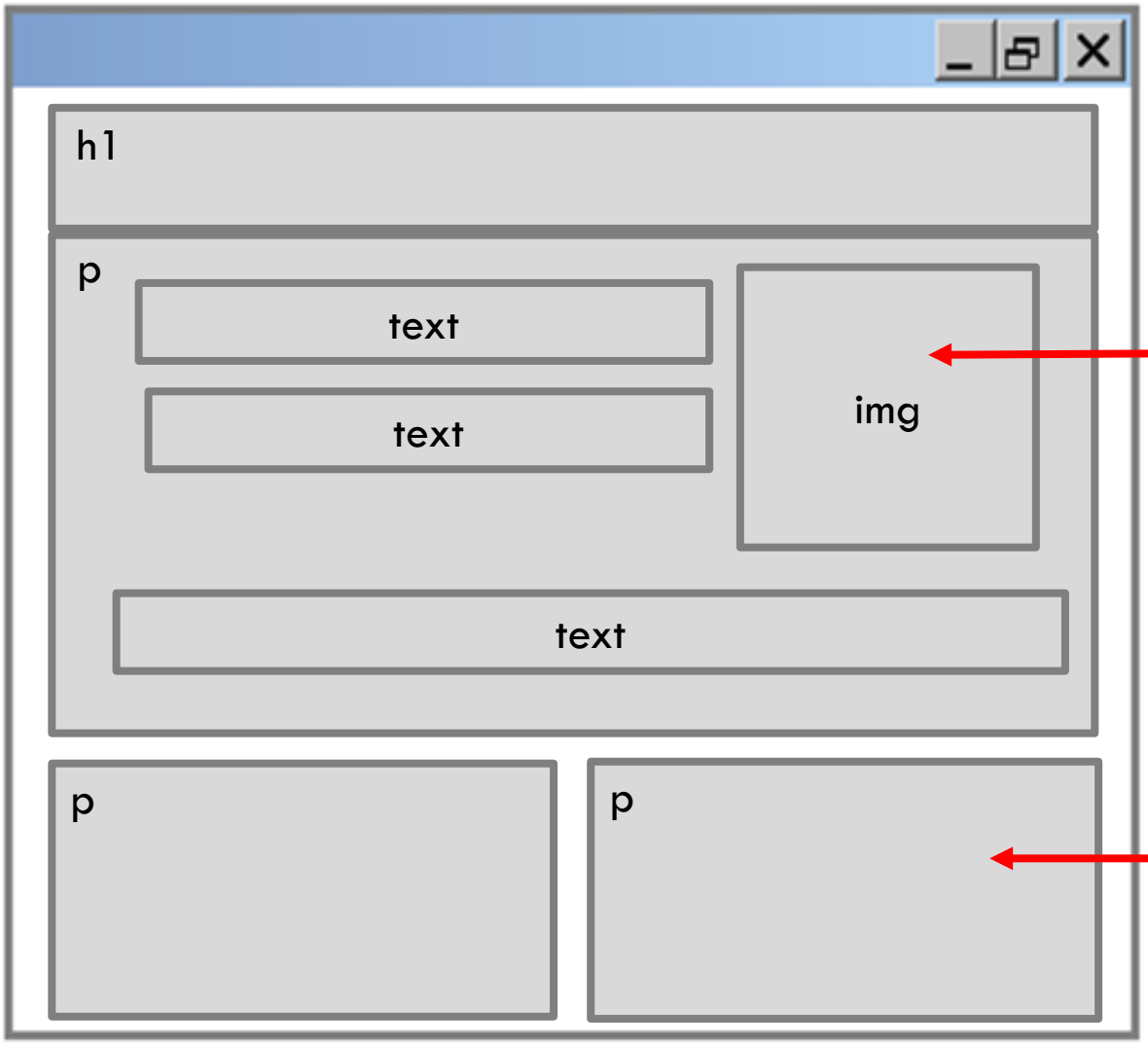


Image is aligned to the right.

Two columns for paragraphs.

Alignment might not work

Let's try to generate two side by side <p>


```css
.left {text-align: left; width: 280px; border: solid 1px;}

.right {text-align: right; width: 280px; border: solid 1px;}
```

CSS

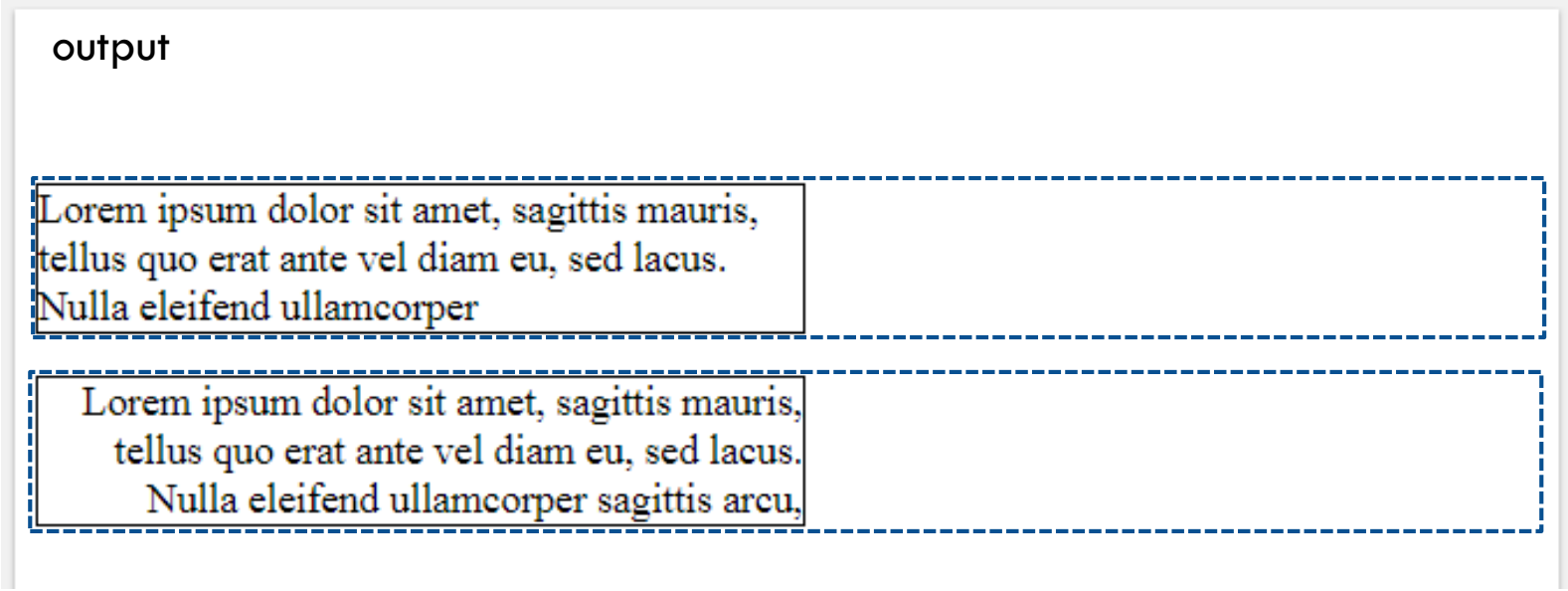
```html
<p class="left"> <!-- align left -->
Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat
ante vel diam eu, sed lacus. Nulla eleifend ullamcorper </p>

<p class="right"> <!-- align right -->
Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat
ante vel diam eu, sed lacus. Nulla eleifend ullamcorper
sagittis arcu, </p>
```

html

Alignment might not work

output



Text-align only affects the **content** inside the block elements, the block elements is still take up the entire horizontal space.

Float - layout property


```css
.left {
      float: left; width: 280px; border: solid 1px;
}

.right {
    float: right; width: 280px; border: solid 1px;
}
```

CSS

```html
<p class="left"> Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat ante vel diam eu, sed lacus. Nulla eleifend ullamcorper </p>

<p class="right">
Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat ante vel diam eu, sed lacus. Nulla eleifend ullamcorper sagittis arcu, </p>
```

html

Floating elements

output

Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat ante vel diam eu, sed lacus. Nulla eleifend ullamcorper

Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat ante vel diam eu, sed lacus. Nulla eleifend ullamcorper sagittis arcu,

The **float** property removes an element from the normal flow.
In this case, the first paragraph floats to the left and the second floats to the right. As long as there is sufficient screen space that they don't overlay, they will share the horizontal space. Note that is generally necessary to specify the ***width*** of the elements.

Float property values


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

CSS

property	description
float	side to hover on; can be **left**, **right**, or **none** (default)

Another example


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

CSS

```html
<p style="border: solid;">
<img class="right" src="https://tinyurl.com/ybafqu2h">
Cute cat. </p>

<p style="border: solid;">
Lorem ipsum dolor sit amet, sagittis mauris, tellus quo
erat ante vel diam eu, sed lacus. Nulla eleifend
ullamcorper sagittis arcu, </p>
```

html

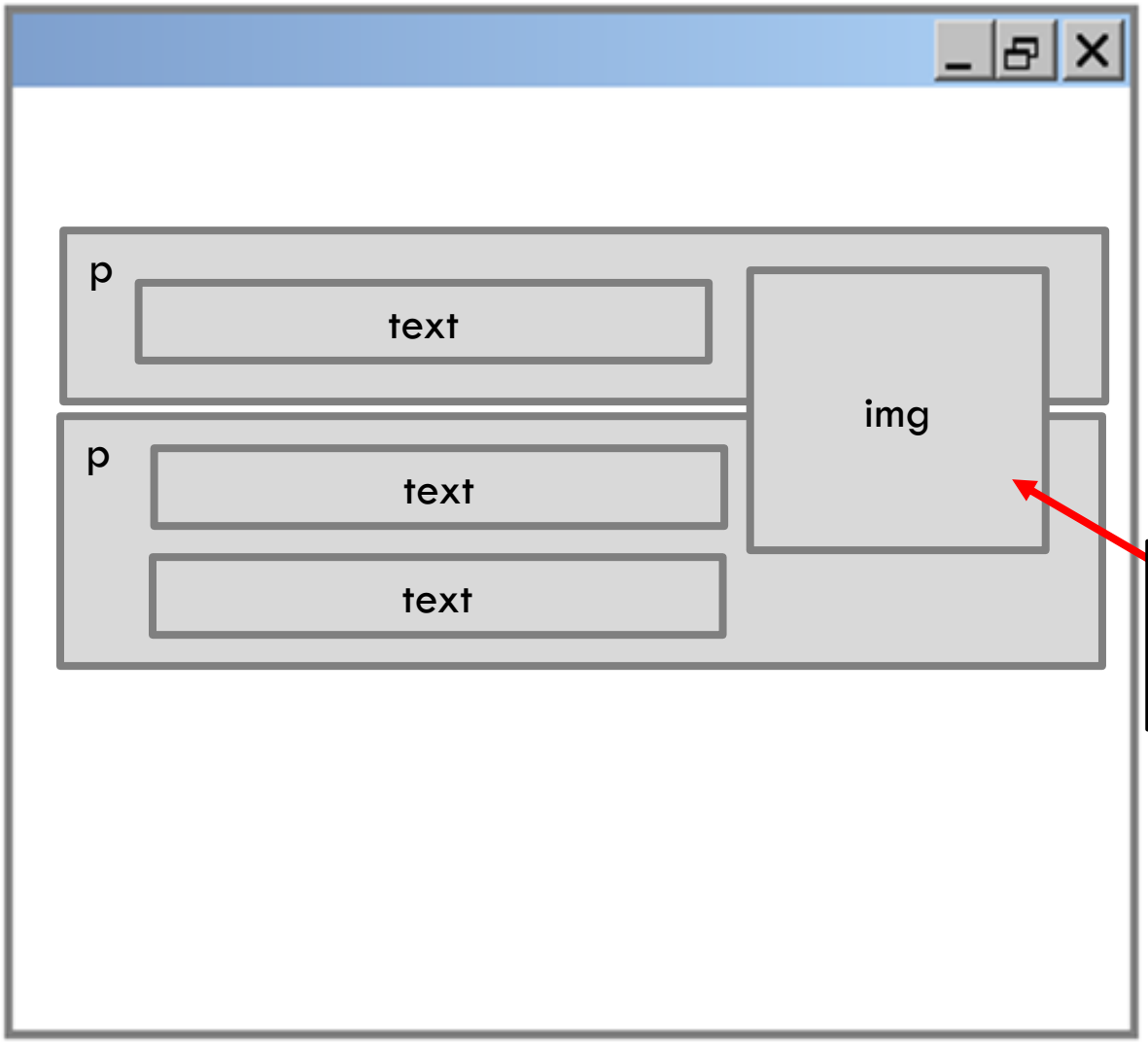
Floating has disadvantages

Cute cat.

Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat ante vel diam eu, sed lacus. Nulla eleifend ullamcorper sagittis arcu,

In this case, the floating <img> element is inside a <p> element, but its content extends beyond the containing <p> element. In this case, it "floats" into the next paragraph and affects the flow of the next element!

Floating has disadvantages



Clear layout property


```css
img.right {
        float: right; width: 130px;
}
p.avoidfloat { clear: both; }
```

CSS

```html
<p style="border: solid;">
<img class="right" src="https://tinyurl.com/ybafqu2h">
Cute cat. </p>

<p class="avoidfloat" style="border: solid;">
Lorem ipsum dolor sit amet, sagittis mauris, tellus quo
erat ante vel diam eu, sed lacus. Nulla eleifend
ullamcorper sagittis arcu, </p>
```

html

Clear property example

Cute cat.

Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat ante vel diam eu, sed lacus. Nulla eleifend ullamcorper sagittis arcu,

The second paragraph has its clear property set to "both", which means clear both "left" and "right" floating elements. As a result, the 2nd paragraph shifts down to clear the floating element.

The `clear` property (cont.)

property	description
clear	disallows floating elements from overlapping this element; can be left, right, both, or none (default)

Try it out here: https://www.w3schools.com/cssref/pr_class_clear.asp

Overflow property


```css
img.right {
                float: right; width: 130px
        }
.catP { overflow: hidden;
          border: 1px black solid;
}
```

CSS

```html
<p class="catP">
<img class="right" src="https://tinyurl.com/ybafqu2h">
Cute cat. </p>

<p style="border: solid;">
Lorem ipsum dolor sit amet, sagittis mauris, tellus quo
erat ante vel diam eu, sed lacus. Nulla eleifend
ullamcorper sagittis arcu, </p>
```

html

Overflow property example

Cute cat.

Lorem ipsum dolor sit amet, sagittis mauris, tellus quo erat ante vel diam eu, sed lacus. Nulla eleifend ullamcorper sagittis arcu,

The first paragraph has its overflow property set to hidden. In this case, since the <p> element does not have a fixed size, it "hides" the overflowing floating element (the cat image) by expanding to fit the image (so there is actually no overflow anymore). ***You may need this for lab #2.***

See slide 32 for what happens with element size is fixed.

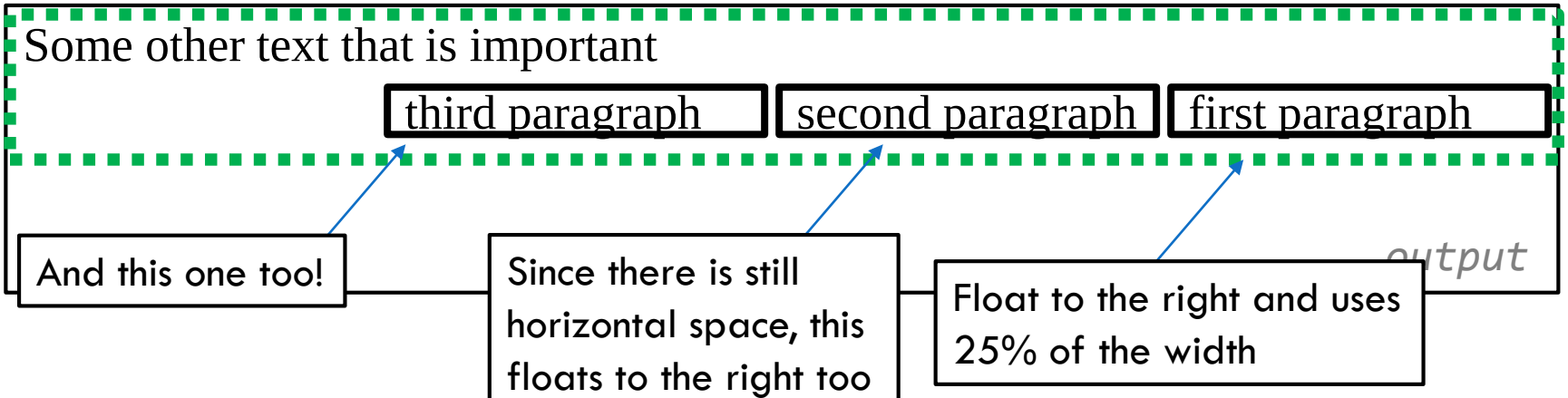
Floating multiple elements


```css
p { float: right; width: 25%; margin: 0.5em;
border: 2px solid black; }
div { border: 3px dotted green; overflow: hidden; }
```

CSS

```html
<div>
        <p>first paragraph</p>
        <p>second paragraph</p>
        <p>third paragraph</p>
        Some other text that is important.
</div>
```

HTML



Floating multiple elements


```
p { float: right; width: 25%; margin: 0.5em;
border: 2px solid black; clear: right;}
div { border: 3px dotted green; overflow: hidden; }
```

CSS

```
<div>
        <p>first paragraph</p>
        <p>second paragraph</p>
        <p>third paragraph</p>
        Some other text that is important
</div>
```

HTML

Some other text that is important

third paragraph | second paragraph | first paragraph

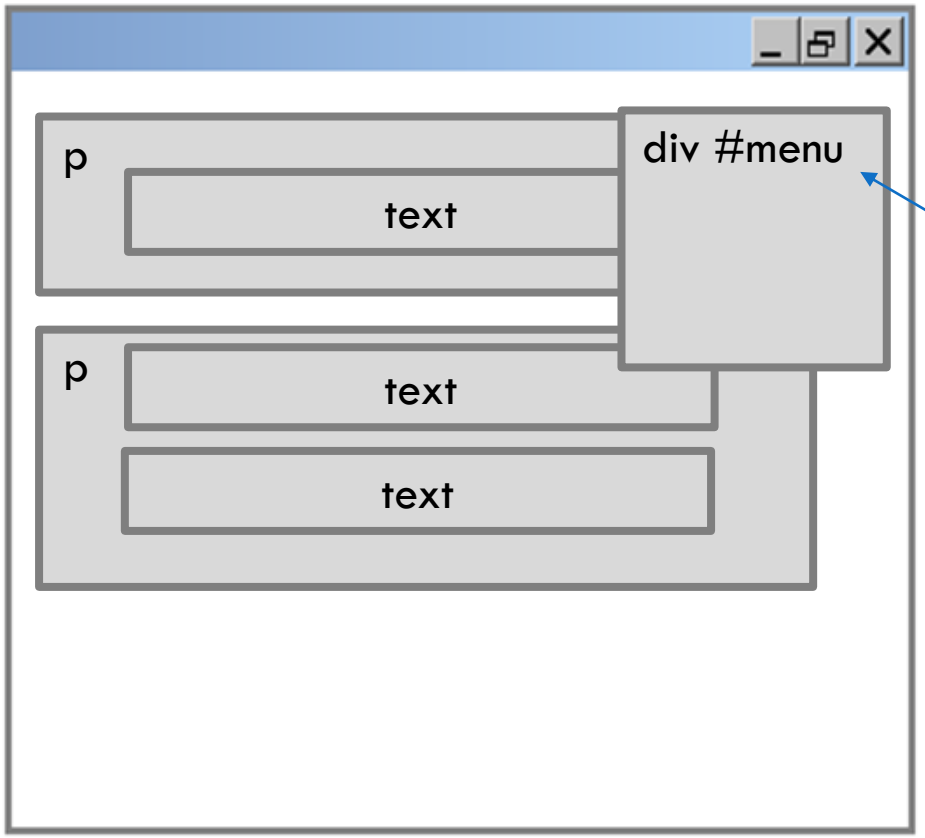
utput

Q: What would happen if we add a "clear" property?

Sizing and Positioning

Instead of float . . give the position

- We can try to give the position of the element directly on the page.



Specify element to appear at this position.

Often used with **div** sections.

An example demo layout

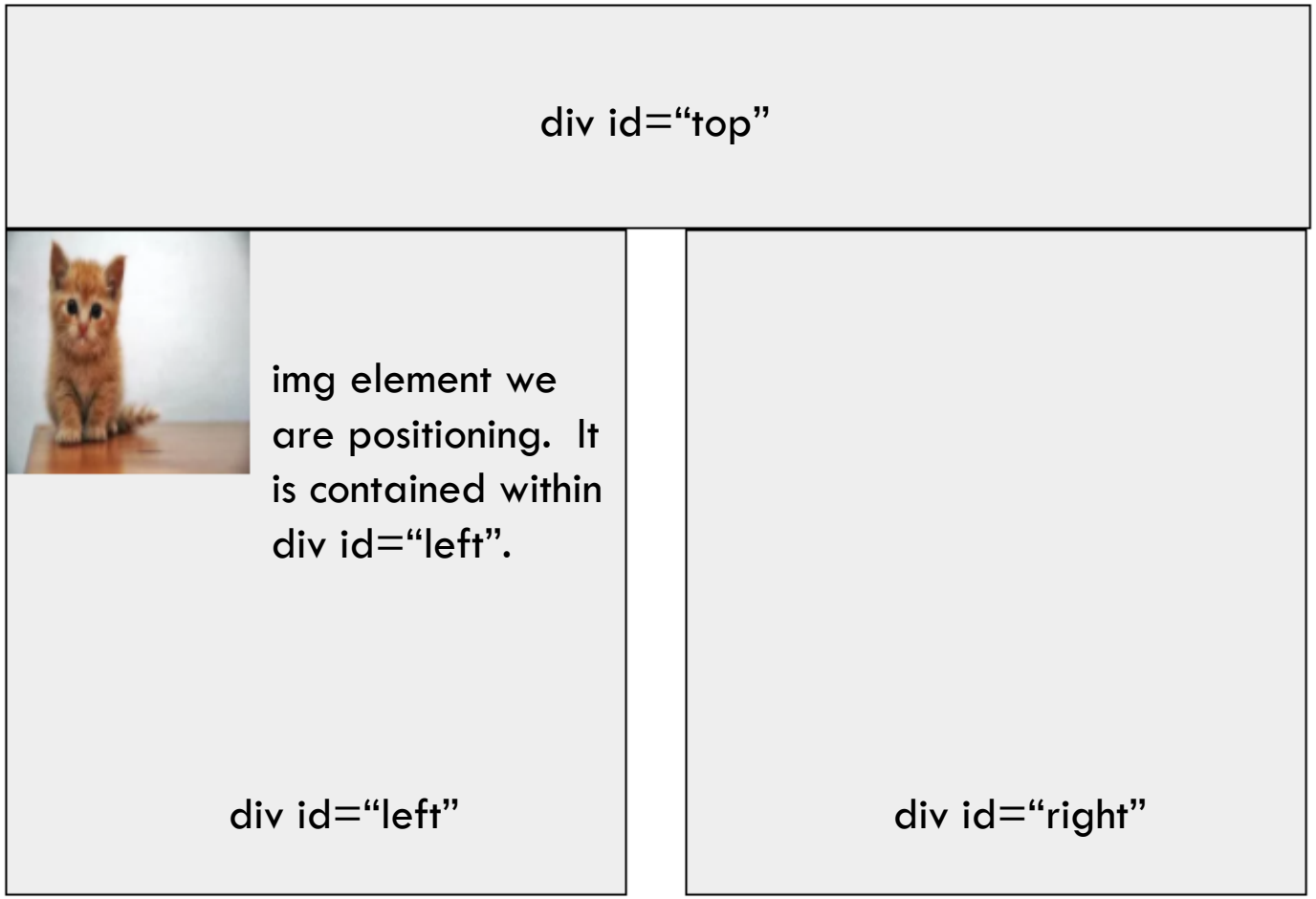

```
#left, #right, #top { border: solid 1px; background-color:
#eee;}
#top      {clear: both;  width: 590px; height:100px; }
#left     {float: left;  width: 280px; height: 300px;  }
#right    {float: right; width: 280px; height: 300px;}
.pos {
    position: static;
    width : 110px; height: 110px; margin:0;
    background-color: red;
}
```

CSS

```
<div id="top">  </div>
<div id="left">
  <img class="pos" src="https://tinyurl.com/ybafqu2h">
</div>
<div id="right">  </div>
```

HTML

Layout example from previous slide



Position: relative


```
.pos {
    position: relative; left: 10px; top: 10px; }
```

div id="top"

element we are positioning.

div id="left"

div id="right"

Relative modifies the position relative to its **original position** in the flow.

You can use negative values! Since this is relative to its **original position**, setting:
left: -10px is the same as right: 10px;
bottom: 10px is the same as top: -10px;

Position: fixed


```
.pos {
    position: fixed; left: 0px; top: 0px; }
```

div id="top"

This modifies the position of the element with respect to the web browser's screen. It will not move, even if you scroll the webpage!

element we are positioning.

div id="left"

div id="right"

Position: absolute

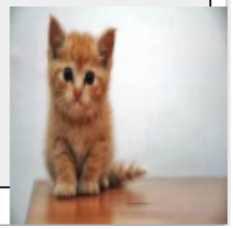

```
.pos {
    position: absolute; bottom: 0px; right: 0px; }
```

div id="top"

element we are positioning.

div id="left"

div id="right"



absolute position is *the most confusing*.
This positions the element with respect to the **last** "containing element" (called an *ancestor*) that also has a position style that was **not** "static". In this case, our original HTML and CSS did not set the position for div id="left" (container for the img element) so, the image is positioned with respect to the <body>. Unlike "fixed" position, the image will scroll with the browser's content.

Image is now positioned with respect to the "body' (that has a width and height of the screen)

Modify the CSS


```
#left, #right, #top { border: solid 1px; background-color:
#eee;}
#top    {clear: both; width: 590px; height:100px;    }
#left   {float: left; width: 280px; height: 300px;
            position: relative;}
#right  {float: right; width: 280px; height: 300px; }
```

CSS

No change.

By setting the position to relative in #left, we have activated its position not to be static. Since we didn't modify the position (we didn't change the left, top, etc), the div position didn't move. However, this will affect the behavior of any contained element that issues an "absolute" position later.

Example – position: **absolute**


```
.pos {
    position: absolute; bottom: 0px; right: 0px; }
```

div id="top"

element we are positioning.

div id="left"

div id="right"

absolute position for the image element is now performed with respect to the div id=left container - since the #left container has a position that is not set to "static".

The div #left defines its height and width, our image is positioned at the bottom of the #left div.

Image is now positioned with respect to "id'

Position property values

property	value	description
position	static	default position
	relative	offset from its normal static position
	absolute	a fixed position within its containing element (that has an active position)
	fixed	a fixed position within the browser window
top, bottom, left, right	positions of box's corners	

When content overflows . .


```
<div id="top"> <p class="pos"> <img class="pos"
src="https://tinyurl.com/ybafqu2h"> </p> </div>
<div id="left">  </div>
<div id="right">  </div>
```

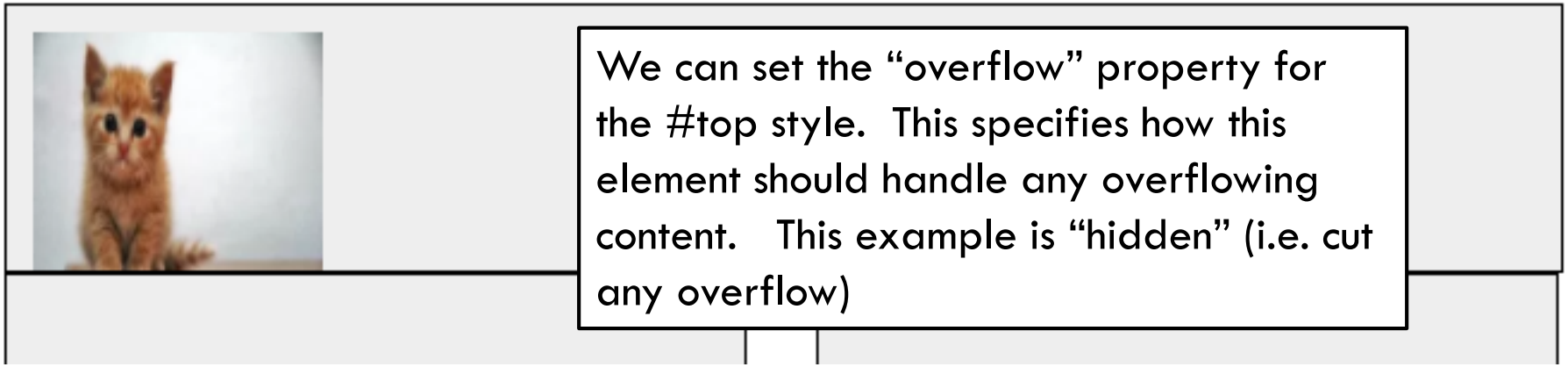
HTML

```
.pos { position: relative; left: 10px; top: 10px; }
```

In this example, I moved the image to be in the div id="top" container. The image "overflows" its container region since the height is fixed for this div section. Now the image overflow into the next section. How can we avoid it?
Note that this is not the same as "clearing" an alignment. The image was ***not* floated**, it was **positioned**!

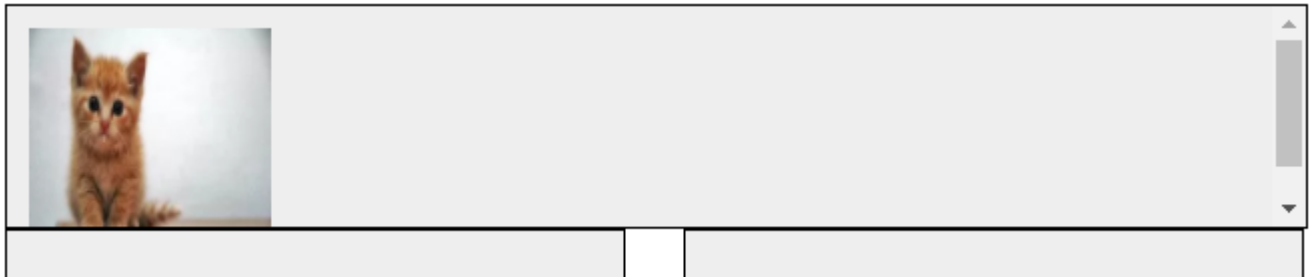
Overflow property


```
#top { clear: both; width: 590px; height:100px;
       overflow: hidden;
```
CSS



We can set the "overflow" property for the #top style. This specifies how this element should handle any overflowing content. This example is "hidden" (i.e. cut any overflow)

```
#top { clear: both; width: 590px; height:100px;
       overflow: auto;
```
CSS



auto property adds in a scroll-bar to see the overflow content!

Overflow property

Value	Description
visible	The overflow is not clipped. It renders outside the element's box. This is default
hidden	The overflow is clipped, and the rest of the content will be invisible
scroll	The overflow is clipped, but a scroll-bar is added to see the rest of the content
auto	If overflow is clipped, a scroll-bar should be added to see the rest of the content

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** ignored

Details about inline boxes

- 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
  - *This is the demonstrated on slide 6*
- each inline box's `vertical-align` property aligns it vertically within its block box

The `display` property


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

CSS

This is a heading **This is another heading**

output

property	description
display	sets the type of CSS box model an element is displayed with

- We can force items to act like inline or block!
- values: `none`, `inline`, `block`, `run-in`, `compact`, ...
- **use sparingly, because it can radically alter the page layout**

The `display` property (cont.)


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

CSS

output

- 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 (we will see this when we do JavaScript)

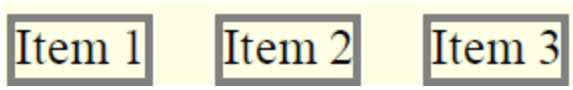
Example: `display` property


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

HTML

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

CSS



output

- lists and other block elements can be displayed *inline*
- The <li> elements above now flow left-to-right on same line
- width is determined by content

Recap

- We are now done with CSS and HTML
- There is a lot of information to absorb
- Keep a bookmark to w3schools.com
- Practice and experience helps, CSS can be very frustrating
- We will not do very complex HTML and CSS in this module

There is more to HTML and CSS

- You have seen a fairly comprehensive overview of the most common items in HTML and CSS
- There is (much) more . .
- E.g. frames - allow you to divide your browser into multiple independent views
- Many more tags, e.g. <marquee> Try it </marquee>
- HTML5 semantic sections
- more . . .