GENERAL ASSEMBLY

LAYOUT

Jonathan Cheng

LEARNING OBJECTIVES

- Experiment and predict effects of floats and clearing CSS positioning
- Experiment and predict effects of relative, absolute, and fixed positioning
- Apply header, footer, sidebar, and multi-column layouts to develop a web page

REVIEW

REVIEW

- Define the CSS box model
- Manipulate content, padding, border, and margin to change element sizes
- Apply colors to elements
- Apply inline and block attributes to elements

POSITIONING

Rules like top, left, right, and bottom are given pixel values originating from the direction they name

POSITIONING TYPES

Position static: The default. True love. Nothing else matters

“I'll never let go, Jack. I'll never let go.”

POSITION STATIC

```
{
    position: static;
}
```

- Position static is the default position attribute of all dom elements
- Position static will ignore all other positioning rules like top, right, bottom, left and z-index

Position relative: the mildly controlling significant other of css positioning

“I like the way you look, but if I could just fix this one little thing…”

POSITION RELATIVE

```
{
    position: relative;
    top: 3px;
    left: 12px;
}
```

- Position relative will lay out the element as if it were static, then tweak it using extra rules without affecting the layout of surrounding elements
- Position relative usually comes with one or more additional rules that tweak the positioning of an element:
- top, right, bottom, left and z-index

Position absolute: 50 shades of strange css positioning

“Come, I want to show you my playroom.”

“You want to play on your Xbox?”

POSITION ABSOLUTE

```
{
    position: absolute;
    top: 3px;
    left: 12px;
}
```

- Position absolute will remove an element from the flow of the page and position it based on its closest containing element that is positioned with something other than static.
- Other elements around where it was originally will collapse
- It usually takes one or more additional position rules to set where the element should end up

Position fixed: the dependable best friend who always stays by your side… or up top

“You are my superior officer. You are also my friend. I have been and always shall be yours”

POSITION FIXED

```
{
    position: fixed;
    top: 3px;
    left: 12px;
}
```

- Position fixed will remove an element from the flow of the page and position it based on the viewport of the page, essentially nailing it your your screen
- Position fixed will not move when you scroll the page
- It usually takes one or more additional position rules to set where the element should end up

Z-INDEX

Normally elements are laid out like so:

- Normal elements in order from top to bottom
- Positioned elements in order from top to bottom

Normal elements that come later will cover normal elements that came before, and positioned elements will cover normal elements and other positioned elements that came before

Sometimes we'd like to reorder the elements of the page so that certain elements cover others.

Z-index: the animated superhero of css positioning

“What does the scouter say about his power level?”

“It’s over 9000!”

Z-INDEX

```
{
    position: relative;
    z-index: 5;
}
```

- Z-index controls the order in which elements are layered
- The z-index attribute is given an integer value or power level. Elements with higher power levels beat out elements with lower power levels and are shown in front.
- Z-indexed elements require a common container which has non-static positioning (called a “stacking context”)

FLOATING

Sometimes we want to take elements out of the normal block/ inline flow of the page and force content to wrap around it

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod quam neque, nec vehicula enim dictum eu. Praesent sit amet auctor ultrices enim. Praesent mattis molestie luctus mauris, vel lobortis orci. Donec condimentum ullamcorper sapien a gravida. In hendrerit lacus dignissim massa condimentum fermentum. Proin erat felis, mattis non nisl non, auctor ultrices enim. Praesent mattis molestie justo, eu malesuada dolor vestibulum id.

This would normally be pretty darn hard using only inline and block styles. Let’s see exactly how hard…

Baseline: an image and text sitting next to each other

luctus mauris, vel lobortis orci. Donec condimentum ullamcorper sapien a gravida. In hendrerit lacus dignissim massa condimentum fermentum. Proin erat felis, mattis non nisl non, auctor ultrices enim. Praesent mattis molestie justo, eu malesuada dolor vestibulum id.

Attempt 1: use {vertical-align: top;} to get the text to line up with the top of the image

Lorem ipsum dolor sit amet, consectetur

adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod quam neque, nec vehicula enim dictum eu. Praesent sit amet auctor ultrices enim. Praesent mattis molestieluctus mauris, vel lobortis orci.

Attempt 2: use {position: relative; top: 0px; left 50px;} to manually move the text up

luctus mauris, vel lobortis orci
condimentum ullamcorper sapien a gravida. In hendrerit l
massa condimentum fermentum. Proin erat felis, mattis n
auctor ultrices enim. Praesent mattis molestie justo, eu m
vestibulum id.

Attempt 3: use {display: inline-block; width: 100px;} to block the text and make it sit next to the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod quam neque, nec vehicula enim dictum eu. Praesent sit amet auctor ultrices enim. Praesent mattis molestie luctus mauris, vel lobortis orci. Donec condimentum ullamcorper

Attempt 4: use two separate elements, one that sits next to the image and one that sits under

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod quam neque, nec vehicula enim dictum eu. Praesent sit amet auctor ultrices enim. Praesent mattis molestie

luctus mauris, vel lobortis orci. Donec condimentum ullamcorper sapien a gravida. In hendrerit lacus dignissim massa condimentum fermentum. Proin erat felis, mattis non nisl non, auctor ultrices enim. Praesent mattis molestie justo, eu malesuada dolor vestibulum id.

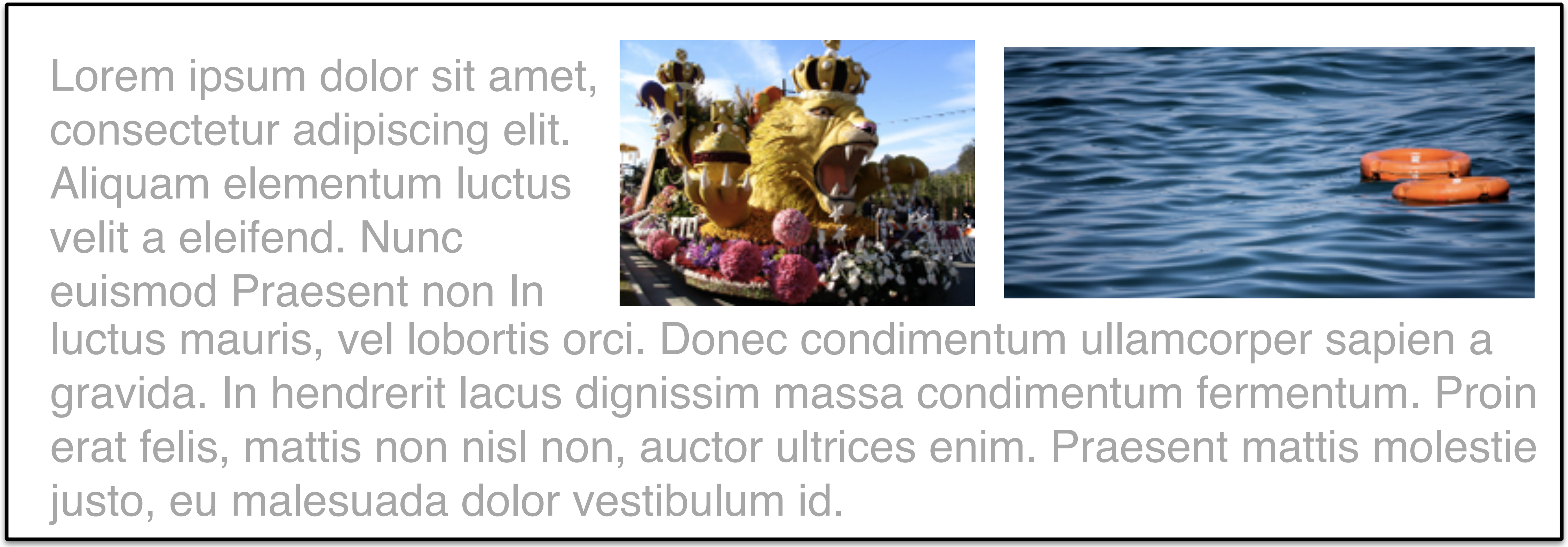
Attempt 5: float the image to the left

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod quam neque, nec vehicula enim dictum eu. Praesent sit amet auctor ultrices enim. Praesent mattis molestie luctus mauris, vel lobortis orci. Donec condimentum ullamcorper sapien a gravida. In hendrerit lacus dignissim massa condimentum fermentum. Proin erat felis, mattis non nisl non, auctor ultrices enim. Praesent mattis molestie justo, eu malesuada dolor vestibulum id.

Attempt 5: multiple floats can be stacked next to each other

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod Praesent non In luctus mauris, vel lobortis orci. Donec condimentum ullamcorper sapien a gravida. In hendrerit lacus dignissim massa condimentum fermentum. Proin erat felis, mattis non nisl non, auctor ultrices enim. Praesent mattis molestie justo, eu malesuada dolor vestibulum id.

Attempt 5: floating right will reverse the order of elements



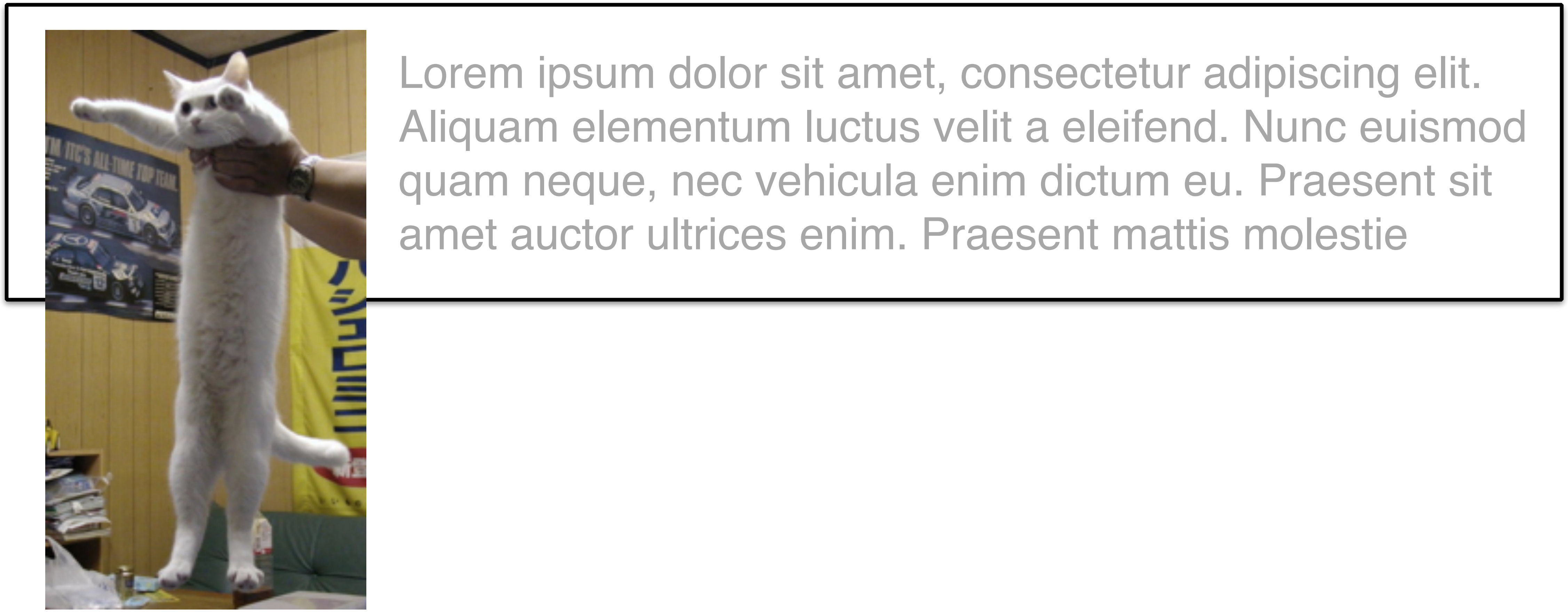
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod Praesent non In luctus mauris, vel lobortis orci. Donec condimentum ullamcorper sapien a gravida. In hendrerit lacus dignissim massa condimentum fermentum. Proin erat felis, mattis non nisl non, auctor ultrices enim. Praesent mattis molestie justo, eu malesuada dolor vestibulum id.

SUCCESS?

KINDA...

Since floats are pulled out of the normal flow of the page, they don't take up space within their containing elements either.

Attempt 5 (if the image were taller than the text)



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod quam neque, nec vehicula enim dictum eu. Praesent sit amet auctor ultrices enim. Praesent mattis molestie

What we'd like to do is make sure the bottom of the container clears the floated element instead of collapsing behind it. Let's see how to do this…

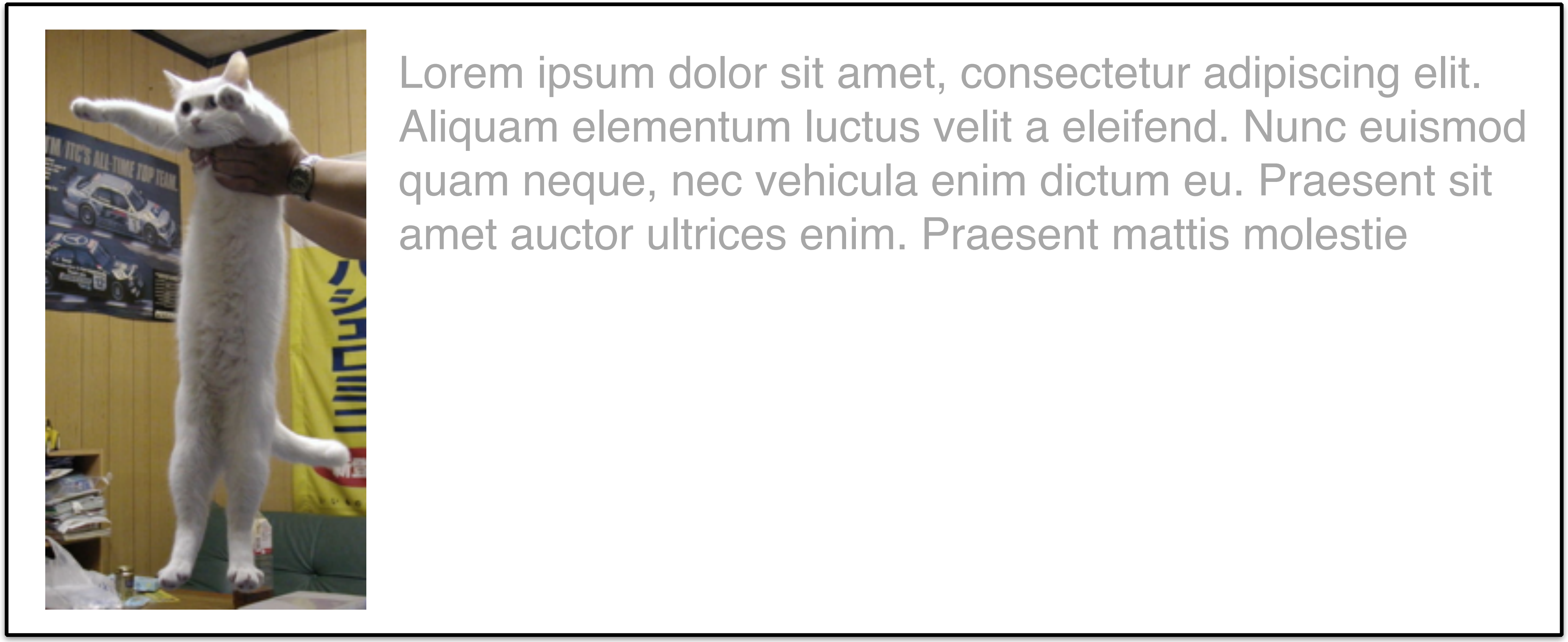
ENTER CLEARFIX

CLEARFIX

A Clearfix is a set of rules we can apply to a container to make sure any elements floated inside it are properly cleared.
Add this to a container and bam, you’re clear:

```
.clearfix:after {
    content: " ";
    display: table;
    clear: both;
}
```

Attempt 5: with clearfix applied to the container


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam elementum luctus velit a eleifend. Nunc euismod quam neque, nec vehicula enim dictum eu. Praesent sit amet auctor ultrices enim. Praesent mattis molestie

There are many css rule combinations that lead to a clearfix, and variations generally depend on what browsers we want to support. For a super solid clearfix and a great explanation, you can read more here: http://nicolasgallagher.com/micro-clearfix-hack/

LETS TAKE A BREAK!

LAB

WRAP UP

LEARNING OBJECTIVES

- Experiment and predict effects of floats and clearing CSS positioning
- Experiment and predict effects of relative, absolute, and fixed positioning
- Apply header, footer, sidebar, and multi-column layouts to develop a web page

HOMEWORK

STYLE A FASHION BLOG

HOMEWORK: FASHION BLOG

- Your assignment this week is to use css to style a fashion blog
- Use the images and html provided to get as close as you can to the finished image
- **Bonus points:** Make sure all the content is centered on the page. Adding only one class to the body, change the visual style of the page completely. You can also add an ascii picture to the comments of your html.
- **To submit:** host your work on BitBalloon. Be sure to share your work with us on slack