HTML & CSS: LEVEL 1

Instructor: Beck Johnson

Week 3

SESSION OVERVIEW

- Review of week 2
- Block vs inline elements
- The CSS box model
- Understanding classes and IDs

REVIEW!

REVIEW: WEB GRAPHICS

- **Minimize** file sizes to help load times in browser
- **Optimizes** images for RGB displays with correct **resolution** for browsers
- **Flattens** layers and removes metadata from graphics

REVIEW: WEB IMAGE TYPES

JPG/JPEG

- Millions of colors
- Uses a compression algorithm called **lossy**
- No animation
- No transparency
- Small file size

REVIEW: WEB IMAGE TYPES

GIF

- 256 colors max
- Animation
- Pixels are either on or off (no partial transparency)

REVIEW: WEB IMAGE TYPES

PNG

- Millions of colors
- No animation
- Full alpha transparency
- No compression, so larger file sizes

REVIEW: ANATOMY OF A CSS RULE

```
selector { property: value; }
```

- `selector` is the **thing** you want to style
- `property` is the **attribute** you want to style
- `value` is how you want to style it
- Values always end in semicolons (`;`)

REVIEW: EXAMPLE CSS RULE

```
p { color: blue; }
```

- `selector` is `p` (all `<p>` tags in the HTML)
- `property` is `color`
- `value` is `blue` (many color names are supported, or use the hex code `#0000ff`)

{} REVIEW: COMMON FONT PROPERTIES

`font-style:` normal, *italic* or *oblique*

`font-weight:` normal, **bold**, or values of 100, 200, etc (depending on the typeface)

`font-family:` the name of a typeface installed on the user's computer

`line-height:` a number followed by a measurement of the height of a line of that element

`font-size:` a number followed by a measurement of the height of that element's text

{} REVIEW: COLORS

`color:` changes the color of text

`background-color:` sets the background color of an element

Color `value` can be set using **names**, **HEX, RGB,** or **RGBA**

- Color name: `white`
- Hex: `#ffffff`
- RGB: `rgb(255, 255, 255)`
- RGBA: `rgba(255, 255, 255, 0.8)`

{} REVIEW: LINKING TO EXTERNAL STYLESHEET

```
<link href="css/styles.css" rel="stylesheet">
```

- Tells the browser to find and load the styles.css file from the css directory
- The `rel` attribute stands for "relation" - in this case, this link's relationship to the document is "stylesheet"
- This tag goes inside the `<head>` element
- Should be on every page that needs the styles

{} REVIEW: THE "CASCADING" PART

The beauty of CSS is being able to create styles and then override them when you want to customize the look of your pages.

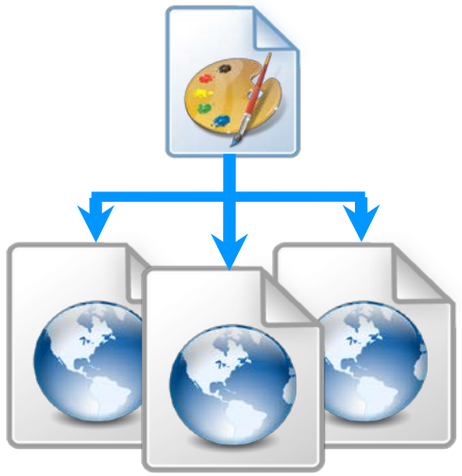
There are three big rules for determining how styles get applied:

- Styles are loaded from far to near.
- Styles are loaded from top to bottom.
- Children elements are more specific than parents.

{} REVIEW: STYLES "LOCATION"

Styles that are "closer" to the elements they style take precedence

Browser default



External styles
(in a **.css** file)

Internal styles
(in the <head>)

Inline styles
(directly on an element)

Closer to element

{} REVIEW: TOP TO BOTTOM

If the same `property` is styled multiple times for the same `selector`, the last one sticks.

```
p { color: #2f4251; }
P { color:  #daa645; } /*this wins*/
```

{} REVIEW: CHILDREN ARE SPECIFIC

Children elements **inherit** styles from their parents but can **override** parents with their own styles

```
body { color: #2f4251; } /* parent */

p { color: #daa645; }    /* child */
```

{} REVIEW: SELECTORS CAN BE MORE SPECIFIC

If one style is **more specific** than another, it takes precedence

```
p { color: #daa645; } /* all paragraphs */

a { color: #e7c0c8; } /* links in general */

p a { color: #c4fe46; } /* links in paragraphs */
```

QUESTIONS?

BLOCK VS. INLINE ELEMENTS

BLOCK-LEVEL ELEMENTS

If no width is set, will expand naturally to fill its parent container
If no height is set, will expand naturally to fit its child elements (assuming they are not floated or positioned)
Can have margins and/or padding
By default, will be placed below previous elements in the markup (assuming no floats or positioning on surrounding elements)

Ignores the vertical-align property

<> BLOCK & INLINE ELEMENTS

Block-level elements we've learned:

- Headings (<h1>, <h2>, etc)
- Paragraphs (<p>)
- Lists (<ul>, <ol>)
- List items (<li>)

<> BLOCK & INLINE ELEMENTS

BLOCK ELEMENTS EXPAND NATURALLY →

AND NATURALLY DROP BELOW OTHER ELEMENTS ↙

<> BLOCK & INLINE ELEMENTS

INLINE ELEMENTS

Flows along with text content
Will ignore top and bottom margin settings, but will apply left and right margins, and any padding

Will ignore the width and height properties
If floated left or right, will automatically become a block-level element, subject to all block characteristics

Is subject to the vertical-align property

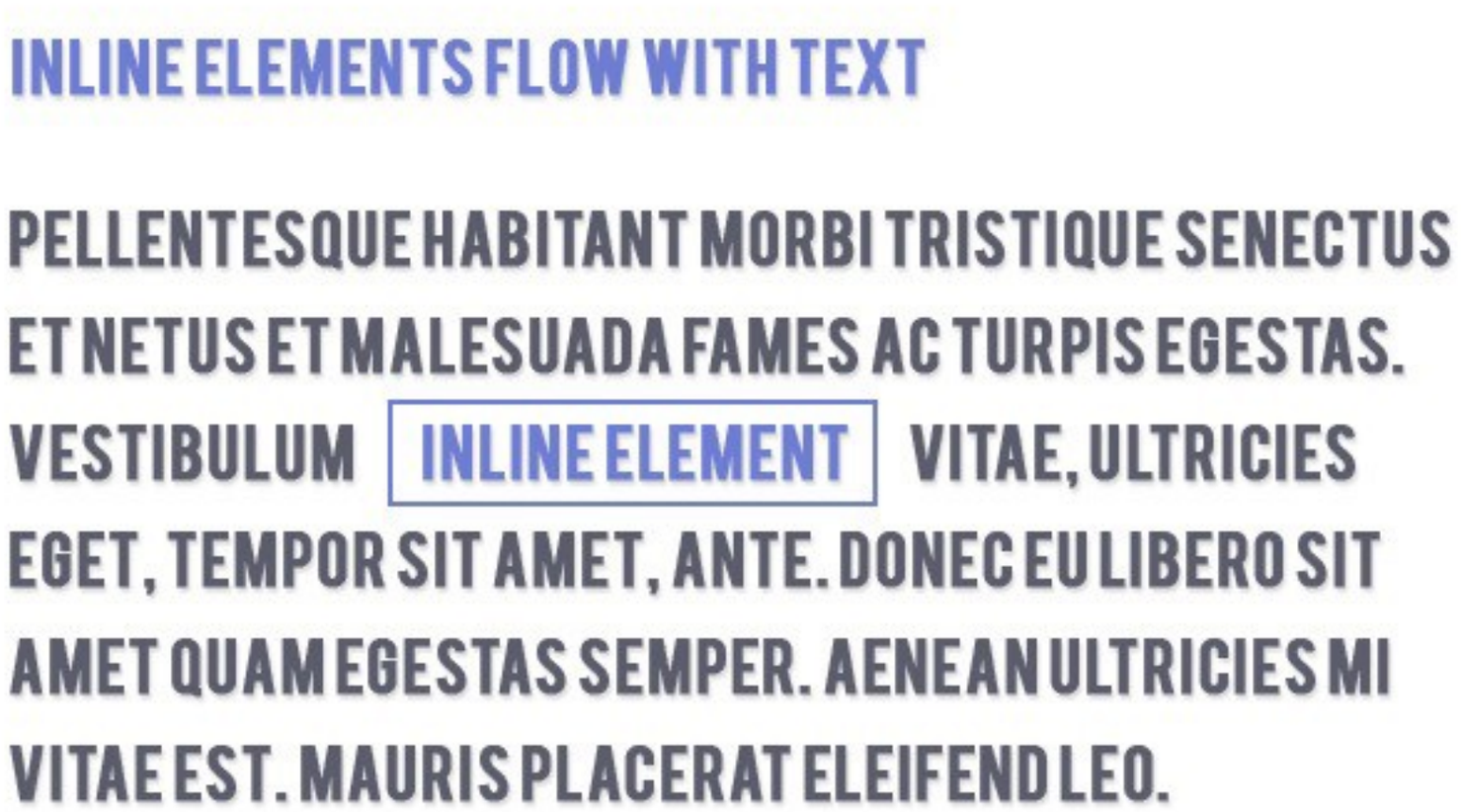
<> BLOCK & INLINE ELEMENTS

Inline elements we've learned:

- Links (<a>)
- Coming soon: **, <b>, **

<> BLOCK & INLINE ELEMENTS

Inline elements



<> BLOCK & INLINE ELEMENTS

“Inline-block” elements

- Inline/Block hybrid.
- Take up width and height like block-level elements.
- Flow with content around them.
- Can have margin and padding.
- Elements we know: **<img>** elements.

<html>

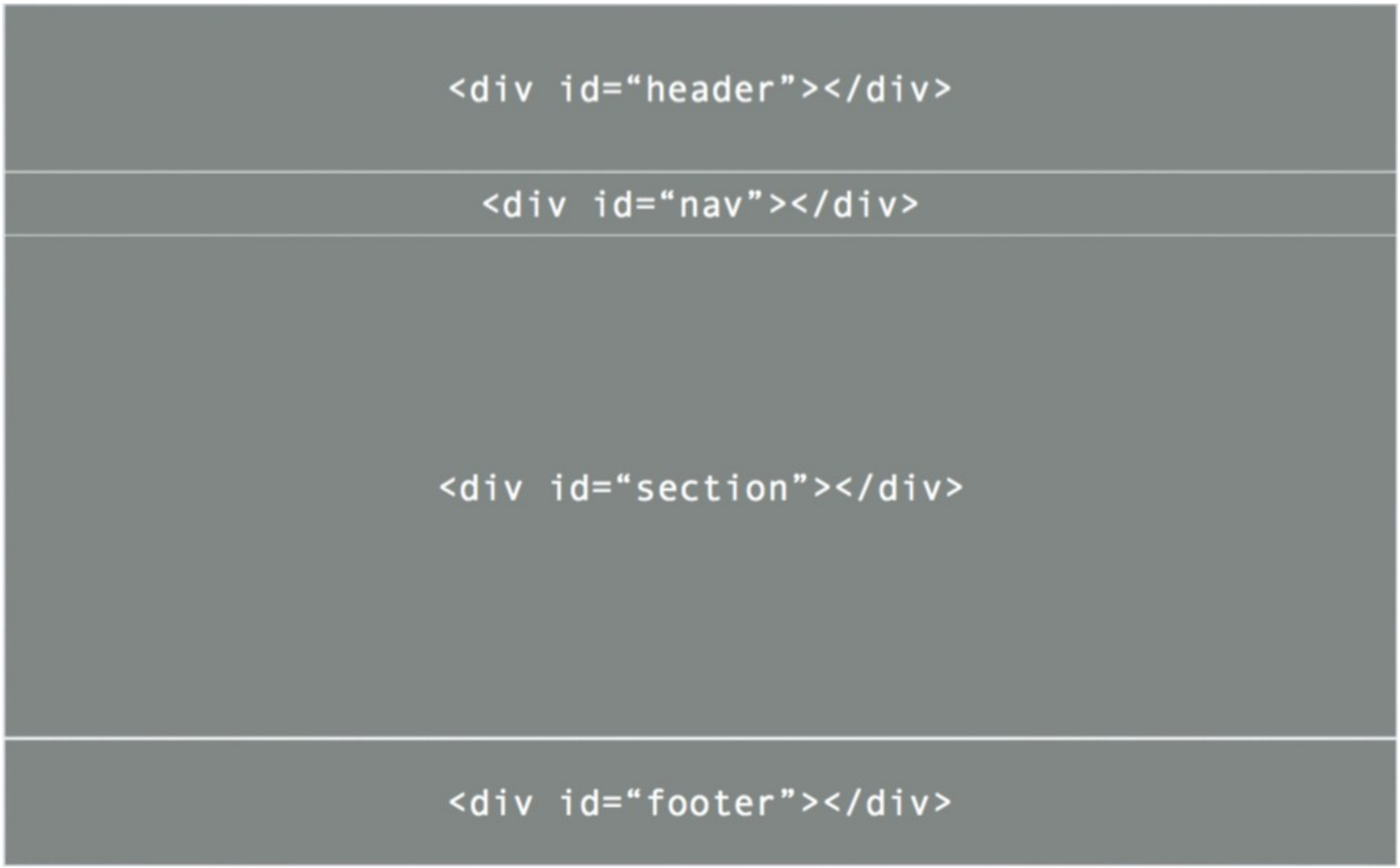
(MORE) HTML ELEMENTS

<DIV> ELEMENTS

```
<div></div>
```

- elements are generic **block elements.**
- Used to create **sections or groups** in HTML for layout.
- Can be used wrappers for other elements (including other divs!) for creating complex layouts.
- Have height and width
- **Building blocks of HTML layouts!**

<DIV> LAYOUT EXAMPLE



<SPAN> ELEMENTS

```
<span></span>
```

- elements are generic **inline elements.**
- Can **nest inside** other block or inline elements.
- Used to **style unique inline content** or **content inside block elements.**
- Flow with content around them.

<> MORE INLINE ELEMENTS

- **em** elements are used to show *emphasis. (like italic).*
- **strong** elements are used to show **importance in context (like bold)**

"Oh, great. Someone ate my only clean socks."

"Was it the cat?"

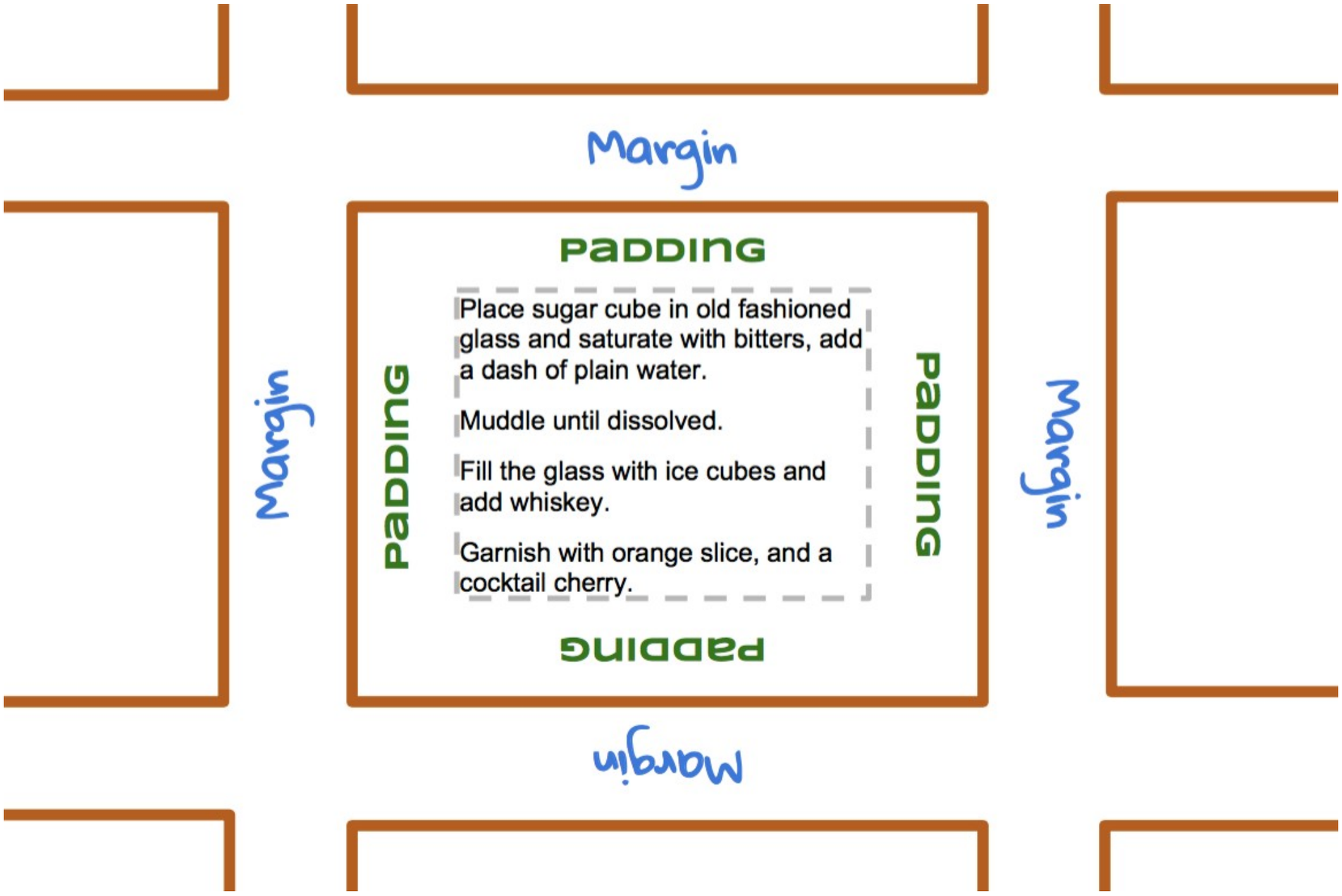
"No, it was the dog.

THE CSS BOX MODEL

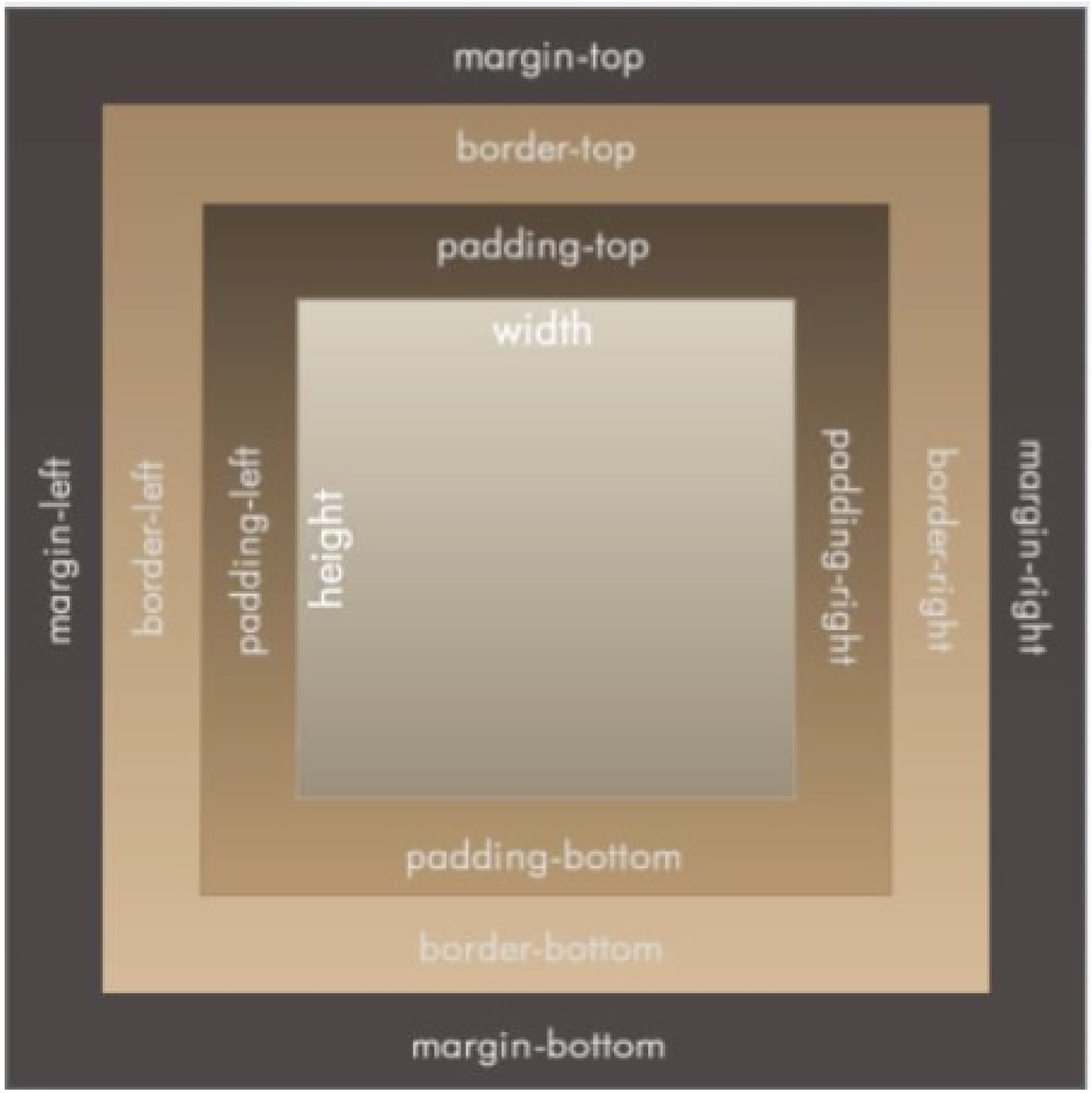
CSS BOX MODEL

- **Content:** stuff in the box
- **Padding:** bubble wrap and packing peanuts
- **Border:** sides of the box
- **Margin:** space between multiple boxes
- In general, the box model applies to **block** ad **inline-block** elements.

CSS BOX MODEL



CSS BOX MODEL



BOX SIZING

```
body {
    box-sizing: content-box; /* browser default*/
}
```

```
body {
    box-sizing: border-box;
}
```

- **border-box** includes border and padding **inside** of width (recommended)

PADDING

- **padding** creates space between content and the **border** for readability/visual spacing.

```
#my-box {
    padding-top: 20px;
    padding-right: 40px;
    padding-bottom 40px;
    padding-left: 20px;
}
/* or.. */
#my-box {
    padding: 20px 40px 40px 20px;
}
```

PADDING

- If **top/bottom** and **left/right** padding match...

```
#my-box {
    padding-top: 20px;
    padding-right: 40px;
    padding-bottom 20px;
    padding-left: 40px;
}
/* COMBINE THEM! */
#my-box {
    padding: 20px 40px;
}
```

PADDING

- If **ALL** padding match...

```
#my-box {
    padding-top: 20px;
    padding-right: 20px;
    padding-bottom 20px;
    padding-left: 20px;
}
/* COMBINE THEM EVEN MORE! */
#my-box {
    padding: 20px;
}
```

MARGIN

- Goes outside the **border.**
- Creates space between the “boxes” of elements.
- Same abbreviation style as padding.
- Can take **negative** values to shift elements opposite direction.

```
#my-box {
    margin-top: -20px;
    margin-left: 30px;
}
```

MARGIN

- Goes outside the **border.**
- Creates space between the “boxes” of elements.
- Same abbreviation style as padding.
- Can take **negative** values to shift elements opposite direction.

```
#my-box {
    margin-top: -20px;
    margin-left: 30px;
}
```

```
#my-box {
    margin: 20px;
}
```

BORDER STYLES

- Allow you to specify the style, width, and color of an element **border**.

```
#my-box {
    border-width: 4px;
    border-color: #000000
    border-style: dotted
}
```

- Abbreviation:

```
#my-box { border 4px dotted #000000}
```

BORDER STYLES

- Learn more about border styles:
- https://developer.mozilla.org/en-US/docs/Web/CSS/border

ID & CLASS SELECTORS

ELEMENT CSS SELECTOR

- Remember we can target all elements in CSS like so:

```
p {
 /* all paragraphs on the page will be targeted */
}
```

CLASSES AND IDS

- **class** and **id** attributes can be added to any HTML element.
- **Classes** are for multiple elements on the page. (styles to re-used)
- **IDs** are for single, unique elements on a page.
- You can create whatever **class** and **id** values you want.

```
<div id="header"></div>
<div class="comment-box"></div>
```

CLASS ATTRIBUTES

```
.comment-box {
  width: 300px;
  padding: 20px;
  margin: auto;
}
```

- **classes** can be shared by multiple elements on a page.
- Elements can have **multiple** classes.

```
<div class="comment-box bg-blue margin-sm"></div>
```

CLASS SELECTORS IN CSS

- Start with a **period** (.)
- Can style any element with the class.

```
.kittens { width: 300px; }
```

- Or can be used to style only a **specific type** of element with the class.

```
h3.kittens { width: 400px; }
```

- Classes are **more specific** than an HTML selector.

ID ATTRIBUTES

- **IDs** *cannot* be shared by multiple elements on a single page.
- Elements ***cannot*** have multiple IDs.

```
<div id="header"></div>
<div id="main"></div>
<div id="footer"></div>
```

ID SELECTORS IN CSS

- Start with a **hash/pound sign** (#)
- Can style the single element with the ID.

```
#kittens { background-color: #000000; }
```

- IDs are **more specific** than class selectors!

MIXING CLASS AND ID ATTRIBUTES

- Elements can have **id** and **class** attributes at the same time.

```
<div id="kittens">…</div>
<div id="puppies" class="small fluffy"></div>
<div id="birds" class="small feathery"></div>
```

- IDs selector styles can be used to override class selector styles.

TIPS

- Recommended order of attack:
  - Type/element selectors
  - Class selectors
  - Descendant selectors
  - ID selectors

- If you overuse **IDs** in your styles, you're going to have a hard time.

SEMANTIC ELEMENTS

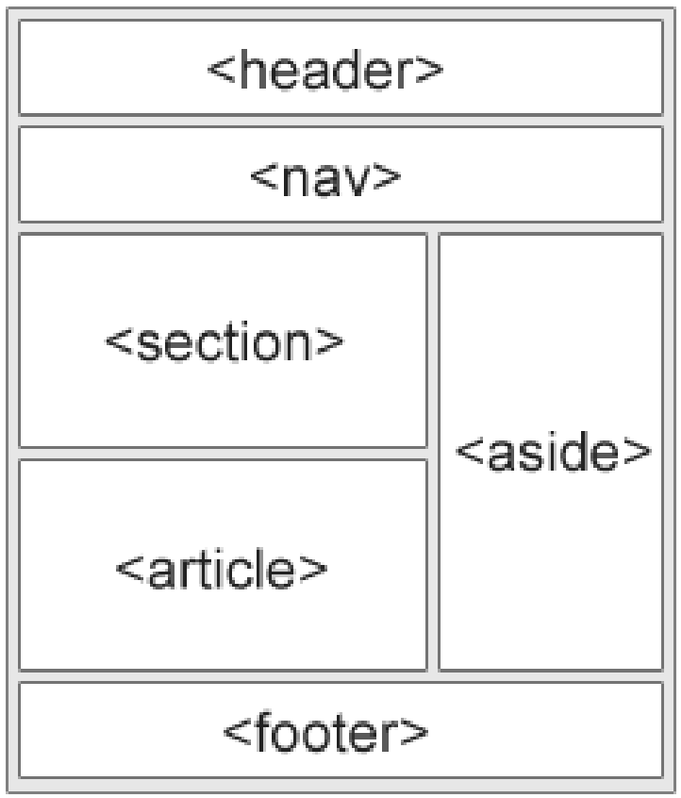
- **Semantics** is the study of the meanings of words and phrases in a language.
- **Semantic elements** = elements with a meaning.
- A semantic element clearly describes its meaning to both the **browser** and the **developer**.
- Non-semantic: ,
- Semantic: <form>, <article>, <section>

SEMANTIC ELEMENTS (HTML5)

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

FURTHER READING

- http://www.w3schools.com/html/html5_semantic_elements.asp



BACKGROUND STYLES

BACKGROUND COLOR REVIEW

```
#my-box {
    text-align: center;
    color: #ffffff
    background-color: rgba(0,0,0,0.5);
}
```

BACKGROUND IMAGES

- The property is **background-image.**
- The value is a **URL where the image lives.** (relative or absolute path)

```
#my-box {
    background-image: url(“images/kitten.jpg”);
}
```

BACKGROUND IMAGES STYLES

- **background-repeat:** repeat/tile image horizontally or vertically; or not at all. (useful for patterns)
- **background-position:** Start at the left or right, top or bottom, center or not.
- **background-attachment:** Is it fixed or does it scroll with page?
- **background-size:** How much of the container does it cover?
- https:/developer.mozilla.org/en-US/docs/Web/CSS/background

BACKGROUND REPEAT

- **background-repeat: repeat-x;**

/* repeat the background horizontally */

- **background-repeat: repeat-y;**

/* repeat the background vertically */

- **background-repeat: no-repeat;**

/* don’t repeat the background */

BACKGROUND POSITION

- **background-position:** Values include both x-axis and y-axis.
- x-axis is first, y-axis is second.
- Can be **left/right top/bottom** *or* any measurement (px, %, ems, etc)

```
#my-box {
    background-position: center center;
}
```

BACKGROUND POSITION EXAMPLES

```
background-position: center center;

background-position: left top;

background-position: right bottom;

background-position: 10px 30px;
```

BACKGROUND ATTACHMENT

- **background-attachment: scroll**
  - background will scroll (default)
- **background-attachment: fixed**
  - background will stick fixed always

“MAGIC” BACKGROUND IMAGE RECIPE

/* make a fill-sized, fixed image background that covers the whole container */

```
#header {
    background-image: url(“images/kittens.jpg”);
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: cover;
    background-position: center center;
}
```

PRACTICE TIME!

ASSIGNMENT

- Create a 1 page website that has a header, nav, main section, and footer, all in a main container div.
- Give them each unique IDs.
- Give them all a common class for base styles.
- Add two links to the nav.
- Add a background image in the header.
- Give your nav a background color and a border.
- Give your elements “breathing” room with padding and/or margin.
- Make use of some pseudo-class styles.