

HTML



CSS



HTML & CSS: LEVEL 1

Instructor: Beck Johnson

Week 3



SESSION OVERVIEW

- Review of week 2
- The CSS box model
- Block vs inline elements
- A couple new HTML elements
- 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: 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

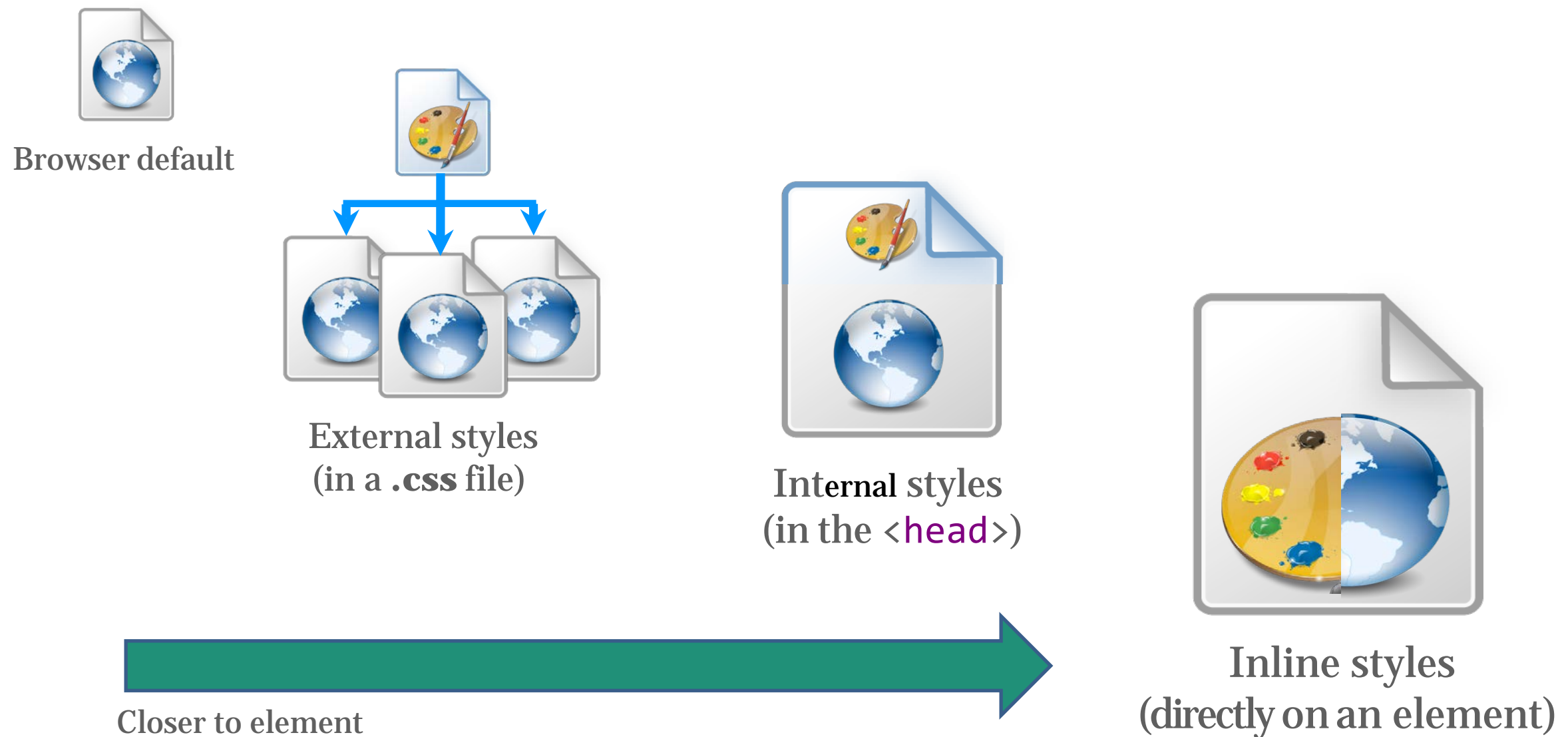
{ REVIEW: THE “CASCADING” PART

The 3 rules for determining how styles get applied:

- Styles are applied from **far** to **near**
- Styles are applied from **top** to **bottom**
- **Children** elements are more specific than **parents**

{ REVIEW: NEAR TO FAR

Styles that are “closer” to the elements they style take precedence



🔗 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 */
```

BACKGROUND-IMAGE: REVIEW

Can set background of an element as an **image** using `background-image`

```
p {  
    background-image: url("images/kitten.jpg");  
    color: white;  
}
```

BACKGROUND: REVIEW

background-position: allows you to move a background image around within its container

background-repeat: defines if (and how) the background image will repeat

background-attachment: changes if the image stays in place when the user scrolls the page or scrolls with the page

background-size: specifies how much of the container that the image covers

PSEUDO REVIEW

A **CSS pseudo-class selector** specifies a special state of the element we want to style

:first-letter styles the first letter of a block of text

:first-child and **:last-child** style the first and last children of a parent

:focus styles an element that has the current keyboard focus, from either click or tab

HEIGHT AND WIDTH: REVIEW

`height` and `width` can be set on (most) elements to change how much room they take up on the page.

```
header { height: 6em; }
```

To ensure an element is **never larger** than a certain value, use `max-height` or `max-width`.

Specify `min-height` or `min-width` if you want to ensure an element is **never smaller** than a certain value.

QUESTIONS?



THE CSS BOX MODEL

CSS BOX MODEL

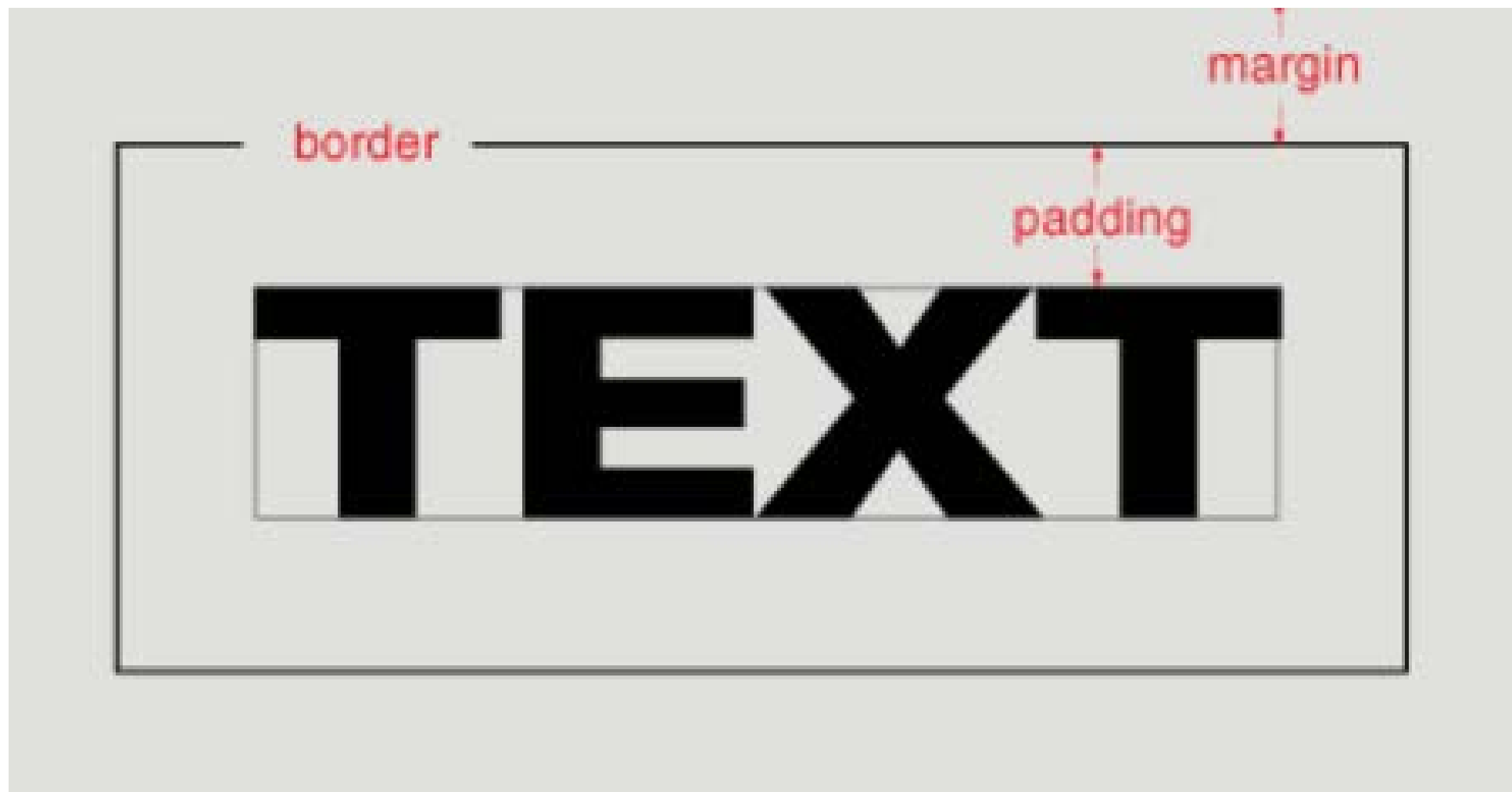
CONTENT: stuff in the box

PADDING: bubble wrap and packing peanuts

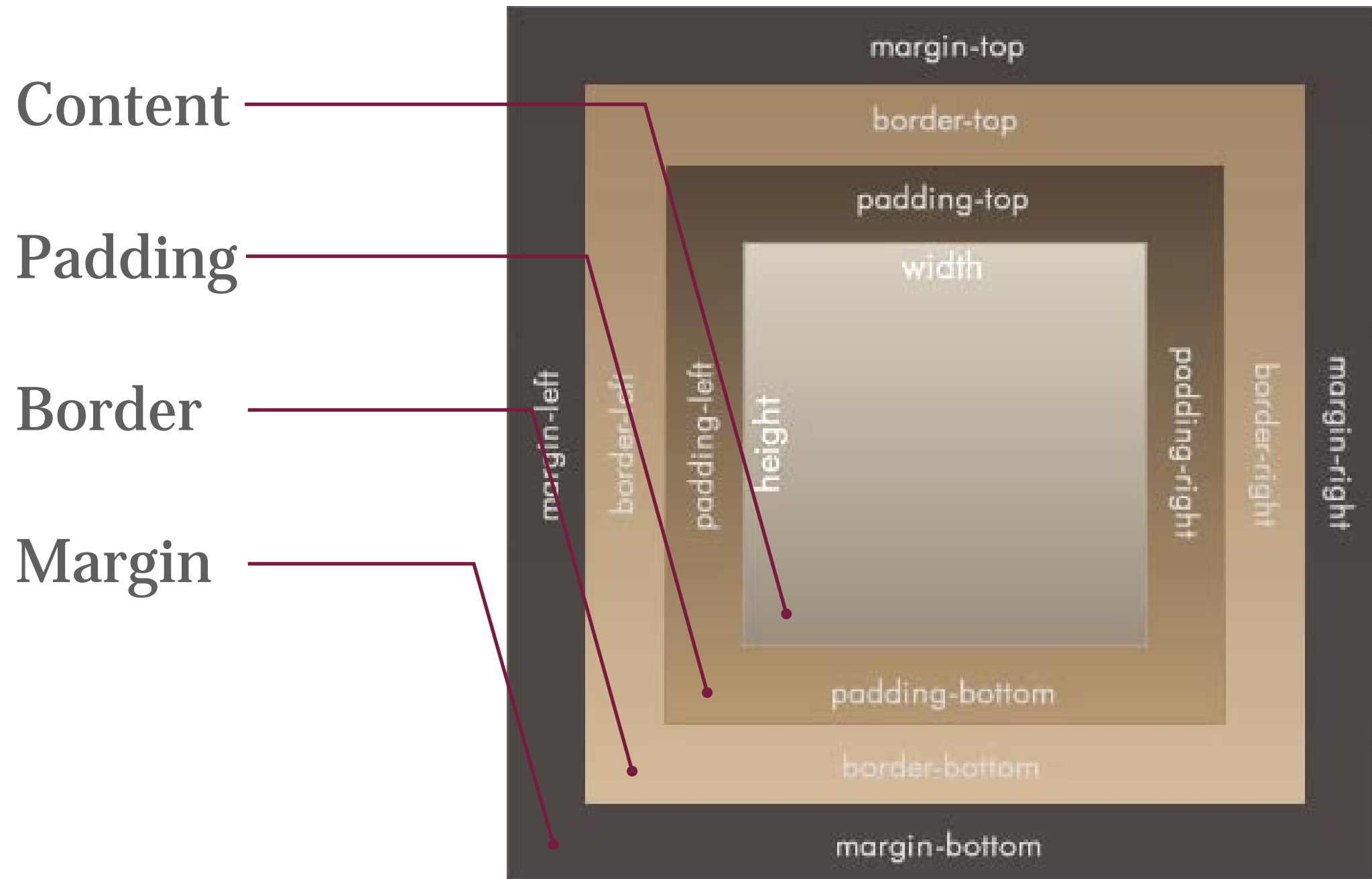
BORDER: sides of the box

MARGIN: space between multiple boxes

CSS BOX MODEL



CSS BOX MODEL

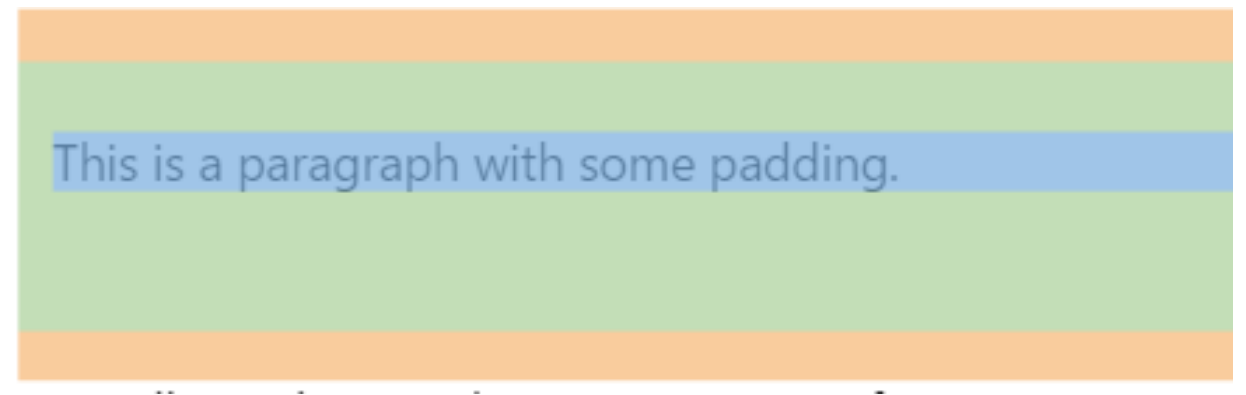


PADDING

Padding creates space **inside** an element.

Padding affects how far content is from the border.

```
p {  
  padding-top: 20px;  
  padding-right: 5px;  
  padding-bottom: 40px;  
  padding-left: 10px;  
}
```



```
p {  
  padding: 20px 5px 40px 10px;  
}
```

PADDING

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

```
p {  
    padding-top: 20px;  
    padding-right: 10px;  
    padding-bottom: 20px;  
    padding-left: 10px;  
}
```

Combine them!

```
p { padding: 20px 10px; }
```


PADDING

If **all** padding matches...

```
p {  
    padding-top: 20px;  
    padding-right: 20px;  
    padding-bottom: 20px;  
    padding-left: 20px;  
}
```

Combine **EVEN MORE!**

```
p { padding: 20px; }
```

MARGIN

Margin creates space **outside** an element.

- Same abbreviation style as padding
- Can take **negative** values to shift elements opposite direction

```
p {  
    margin-top: 20px;  
    margin-left: -20px;  
}
```

BORDER STYLES

Border is between margin and padding.

Specify:

- Width (in pixels)
- Border style (solid, dotted, dashed, etc)
- Color


```
p {  
    border: 2px dotted #ff0000;  
}
```

BORDER STYLES

Border styles:


solid

Solid line.




dotted

Series of dots.




dashed

Series of dashes.



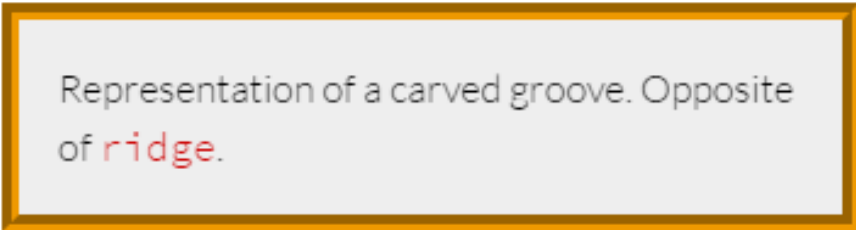
double

Two solid lines.



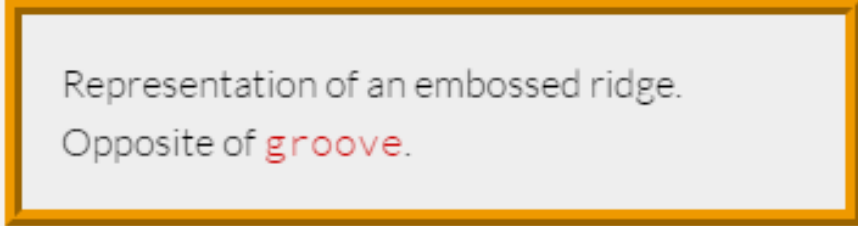
groove

Representation of a carved groove. Opposite of **ridge**.



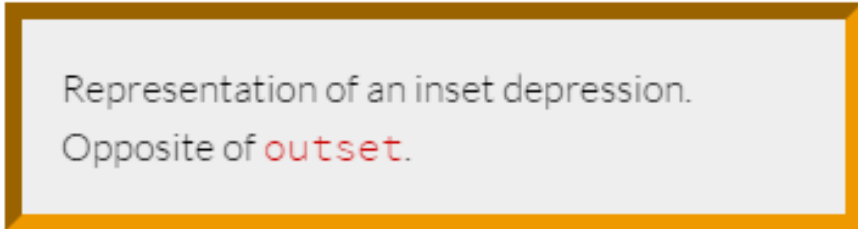
ridge

Representation of an embossed ridge.
Opposite of **groove**.



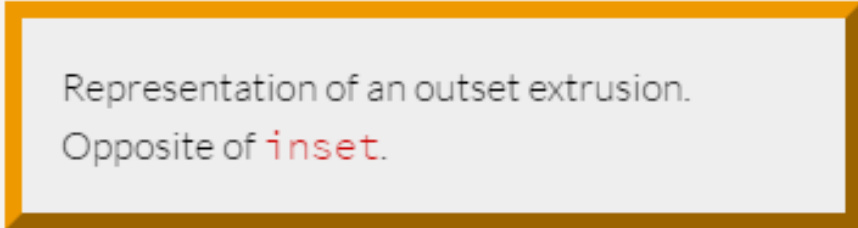
inset

Representation of an inset depression.
Opposite of **outset**.



outset

Representation of an outset extrusion.
Opposite of **inset**.





MARGIN & PADDING DEMO



BLOCK VS. INLINE ELEMENTS

<> BLOCK & INLINE ELEMENTS

BLOCK ELEMENTS

- Expand naturally to fill their parent container
- Can have margin and/or padding
- By default, will be placed **below** previous elements in the markup
- Examples of block elements:
 - Headings `<h1>...<h6>`
 - Paragraphs `<p>`
 - Lists ``, ``

<> BLOCK & INLINE ELEMENTS

BLOCK ELEMENTS EXPAND NATURALLY



AND NATURALLY DROP BELOW OTHER ELEMENTS



<> BLOCK & INLINE ELEMENTS

INLINE ELEMENTS

- Flow along with text content
- Ignores top and bottom margins, but honors left and right margins (and any padding)
- Ignores width and height properties
- Examples of inline elements:
 - Links `<a>`
 - Font emphasis `` and ``

<> BLOCK & INLINE ELEMENTS

INLINE ELEMENTS FLOW WITH TEXT

PELLENTESSQUE HABITANT MORBI TRISTIQUE SENECTUS
ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS.
VESTIBULUM **INLINE ELEMENT** VITAE, ULTRICIES
EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT
AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI
VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

<> BLOCK & INLINE ELEMENTS

INLINE-BLOCK ELEMENT

- Is a hybrid of inline and block
- Takes up width and height like block-level elements
- Flows with content
- Can have margin and padding
- Examples of inline-block elements:
 - Image ``



ID & CLASS SELECTORS

CLASSES AND IDS

CSS lets us target all elements like this:

```
p {  
    font-size: 20px;  
}
```

But what if we want to style only **some** elements?

CLASSES AND IDS

You can add **class** and **id** to any HTML element to identify it for styling.

- **Multiple** elements can have the same **class**
- If an element will be single and **unique**, you can use **id**
- You decide the **class** and **id** values – be descriptive!

```
<p class="important">Big text</p>
```

CLASS SELECTORS IN CSS

- In CSS, target a class with a **period**
- Can style **any** element that has that class

```
.kittens { color: gray; }
```

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

```
h3.kittens { color: gray; }
```

CLASS ATTRIBUTES

- **Multiple** elements can have the same **class**
- Elements can have **multiple** classes

```
.important { font-size: 20px; }
```

```
.margin-sm { margin: 5px; }
```

```
<p class="important margin-sm">Big text  
with a margin</p>
```


ID ATTRIBUTES

- An **id** can only be used **once** per page
- Elements can **not** have multiple **id** attributes

```
<div id="mainContent">  
</div>
```

ID SELECTORS IN CSS

In CSS, target an id with a **hash**

```
#kittenContainer {  
    color: gray;  
}
```

ID selectors are more **specific** than classes or elements

MIXING CLASS AND ID ATTRIBUTES

An element can have both **id** and **class** attributes

```
<div id="puppyContainer" class="small  
fluffy"></div>
```

```
<div id="birdContainer" class="small  
feathery"></div>
```

TIPS

When you style a site, start with **element selectors**.

If some elements need different styles, use **class selectors**.

If a single element needs a special style, use an **ID selector**.

- Overuse of **id** makes it hard for styles to “cascade” and can create a lot of extra work

<html>

(MORE) HTML ELEMENTS

<DIV> ELEMENTS

<div></div>

A <div> is a **generic block element**

- No default style
- Heavily used as a wrapper for other elements, to create complex layouts

<DIV> LAYOUT EXAMPLE

```
<div class="header">
  <h1>Header div</h1>
</div>

<div class="row">

  <div class="sidebar">
    Sidebar div
  </div>

  <div class="main">

    <h2>Main container div</h2>
    <p>This is the div that holds the content for the main container</p>

    <div class="callout">
      <h4>This div is for callouts</h4>
      <p>And holds some special data.</p>
    </div>
  </div>
</div>
```

<DIV> LAYOUT EXAMPLE

Header div

Sidebar div

Main container div

This is the div that holds the content for the main container

This div is for callouts
And holds some special data.

 ELEMENTS

A is a **generic inline element**

- No default style
- Used to style inline content



PRACTICE TIME!

ASSIGNMENT

- Create a new 1 page website that has a header, nav, main section, and footer, all in a main container div.
- Use semantic elements (elements that have meaning)
- Add two links to the nav.
- Add a background color or image and a border to the navigation menu.
- Give your elements some “breathing room” with padding and/or margin.
- Use classes when styling

“HOMEWORK”

- Practice!
- Optional: read chapter 8 of *HTML and CSS: Design and Build Websites*
- Try interacting with this [interactive demo](#) of the CSS box model

