HTML & CSS Level 1: Week 3

May 19, 2015

Instructor: Devon Persing

This week

- Week 2 review
- Web fonts
- Block versus inline elements
- The CSS box model
- Data tables

Review!

History designed by Ema Dimitrova from the Noun Project

{ } Anatomy of a CSS rule

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

- **Selector** is the thing you want to style
- **Property** is what aspect you want to style
- **Value** is how you want to style it
- **Property** + **value** = **declaration**
- **Declarations** end in semicolons (`;`)

{ } CSS rule example

```
h1 { font-size: 2em; }
```

- **Selector** is `h1` (any `<h1>` on the page)
- **Property** is `font-size`
- **Value** is `2em`

{ } Type selectors

- **Type selectors** are single HTML element names that style all elements of that type on the page

```
h1 { font-size: 2em; }
```

{ } Descendent selectors

- **Descendent selectors** point to *children* of other selectors and are *more specific*

```
/* less specific type selector will style all <a>
elements */
a { font-weight: bold; }

/* more specific type selector will style only <a>
elements that are children of <p> elements */
p a { font-weight: normal; }
```

{ } Multiple declarations

- Rules can have **multiple declarations**

```
/* single declaration */
a { font-weight: bold; }

/* multiple declarations (and multiple selectors!) */
a, span {
  font-weight: bold;
  font-style: italic;
}
```

{ } Multiple selectors

- Rules can have **multiple selectors**
- Selectors can be of **mixed kinds**

```
/* single selector */
a { font-weight: bold; }

/* multiple selectors */
a, span { font-weight: bold; }

/* multiple selectors with a descendent selector */
p a, span { font-weight: bold; }
```

{ } Using styles in multiple places

- **Inline styles** are applied to only a single element (we'll talk about a better way to do this next week!)
- **Internal styles** are added in the head of a page and style only that page (what we've done so far)
- **External styles** are called into multiple pages, to style a whole site

{ } Stylesheet "location"

- Styles that are "closer" to the elements they style take precedence
  - Browser defaults
  - External styles (in a `.css` file)
  - Internal styles (in the `<head>`)
  - Inline styles (in an element)

Least specific

↓

Most specific

{ } Overriding styles

- Rules that target children are more specific than rules children inherit from parents
- Rules that come later override rules that come earlier
- **Hint:** Web Inspector will list the most specific styles on top and cross out overridden styles

What limitations did you run into with the CSS we covered last week?

?

Questions?

?

Web fonts

{ } Freedom from Arial!

- Web fonts let us style sites with **fonts that users may not have** on their own device
- Web font services **licence fonts for online use** specifically
- Files are either:
    - hosted by a service
    - served with your pages

A note about licensing

- **Not all fonts can be used online**, even if you own their rights for print, they're in Adobe products, etc.
- Fonts with online licensing will come with **documentation saying so**
- **Exception:** If you own the rights to use a font with software, you can use it to make images that are published online

Award designed by Luc Poupard from the Noun Project

{ } Some web font options

- **Google Fonts** is free and hosted
- **TypeKit** (owned by Adobe) is hosted and subscription based or bundled with Creative Cloud
- **FontSquirrel** is free and not hosted
- **FontDeck** is subscription based and not hosted
- Many integrate with other web solutions

{ } Using hosted Google Fonts

1. Go to google.com/fonts.
2. Build your font library.
3. Link to the stylesheet that Google Fonts generates in your HTML files.

Block and inline HTML

<> Block and inline elements

Block elements we know:

- Headings (**`h1`**, **`h2`**, etc.)
- Paragraphs (**`p`**)
- Lists (**`ul`**, **`ol`**)
- List items (**`li`**)

Inline elements we know:

- Links (**`a`**)

<> Block and inline elements con't.

Block elements:

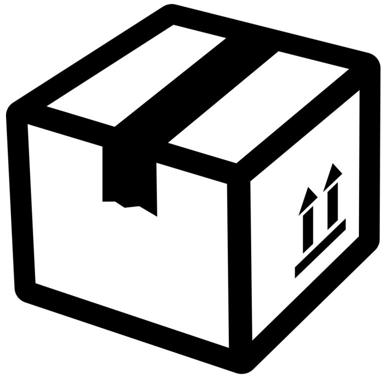
- Create **linebreaks**
- **Take up "space"** on the page

Inline elements:

- **Don't** create linebreaks
- **Flow within other content** on the page

<> `<div>` elements

- **`div`** elements are **generic block** elements
- Used to **create sections or groupings** in HTML pages for layout and style
- Function **like a box** to put content (or other **`div`** elements) inside
- Have heights and widths

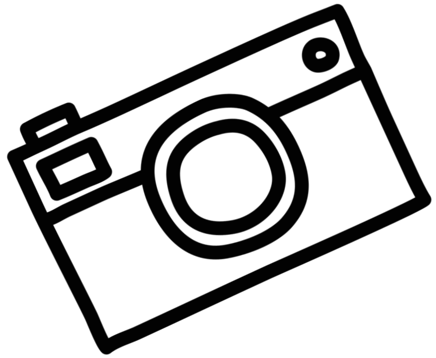

Box designed by Mourad Mokrane from the Noun Project

<> `<span>` elements

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

<> The rare inline-block element

- **Inline-block** elements behave a bit like both block and inline elements:
    - Take up height and width like block elements
    - Flow with content around them
- So far we know **`img`** elements



<> More inline elements

- `em` elements are used to show the equivalent of *spoken emphasis*
- `strong` elements are used to show **importance in context**

```
<p>"Oh, great. Someone ate <em>my only clean socks</em>."
</p>

<p>"Was it <strong>the cat</strong>?"</p>

<p>"No, it was <strong>the dog</strong>."</p>
```

{ } Width and height

- Some elements have width and height by default
- You can set the width and height of images with HTML attributes:

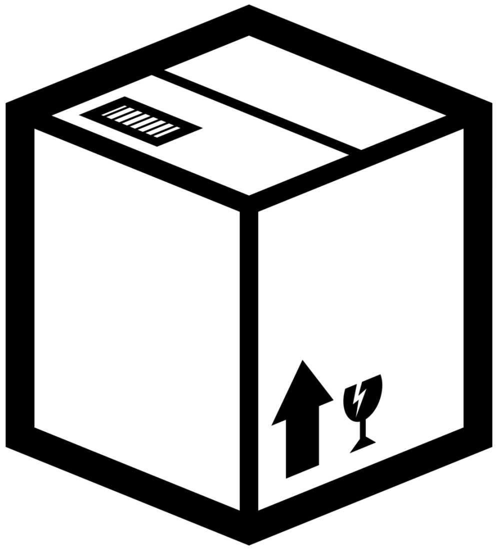
```
<img src="example.jpg" alt="" width="
300px" height="200px">
```

- But it's recommended to adjust them with CSS:

```
img { width: 300px; height: 200px;}

img { width: 300px; height: auto;}
```

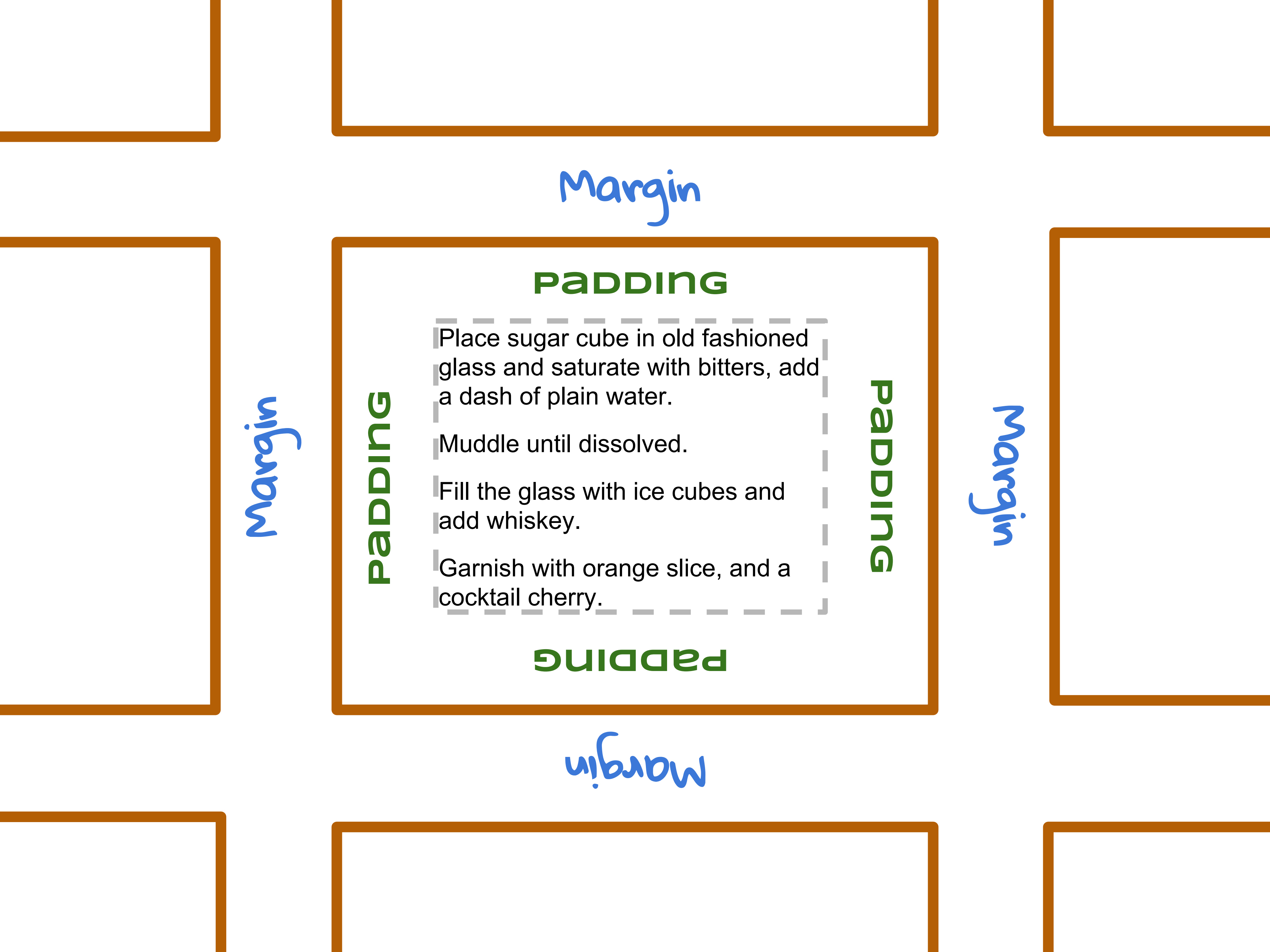
The CSS box model



Box designed by Cornelius Danger from the Noun Project

A CSS box model metaphor

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



Place sugar cube in old fashioned glass and saturate with bitters, add a dash of plain water.

Muddle until dissolved.

Fill the glass with ice cubes and add whiskey.

Garnish with orange slice, and a cocktail cherry.

{ } Box model content

- By default, content helps determines the default **`width`** and **`height`** of the element's box
- Defaults for block elements can be overridden with CSS

```
div { /* px, em, %, auto, etc. */
   width: 400px;
   height: 200px;
}
```

{ } Box model `padding`

- Creates space between content and the **`border`** for readability

```
padding-top: 20px;
padding-right: 20px;
padding-bottom: 40px;
padding-left: 40px;
```

{ } Box model `border`

- Goes around the edge of the element
- Default **`width`** is **`0`** for most elements
- Borders can have **`color`** and **`style`** too

```
border-width: 20px;
border-style: dotted;
border-color: #ff0000;

/* border-width for the bottom edge only */

border-bottom-width: 4px;

/* border-color for the left edge only */

border-left-color: #ff0000;
```

{ } Box model `margin`

- Goes outside the **`border`**
- Creates space between boxes
- **Can be negative** to shift elements

```
margin-top: -20px;
margin-right: 40px;
```

"Seeing" the box model

1. Open your Web Inspector (right click in the browser and choose "Inspect Element")
2. Hover your mouse over a line of code within the `<body>`
3. See different colors to denote different parts of the box

Toolset designed by Calvin Goodman from the Noun Project

{ } What is up with my box sizes?

- How containers take up space is dictated by the **`box-sizing`** property
- The default value for **`box-sizing`** is **`content-box`**

```
box-sizing: content-box;
```

- This means that **`width`** and **`height`** include only the content by default

{ } `border-box` to the rescue

- Changing **`content-box`** to **`border-box`** makes it so that the width and height include the border and padding

```
html {
  box-sizing: border-box;
}
*, *:before, *:after {
  box-sizing: inherit;
}
```

Data tables

<> What's a `table` good for?

- Presenting data in a tabular format
  - Listings of people, addresses, etc.
  - Financial data
  - Product feature comparisons
- HTML emails :(

<> Basic table elements

- **`<table>`** wraps all **table** elements
- **`<tr>`** creates a **row** of table cells
- **`<th>`** creates a **table header** cell for a column *or* a row
- **`<td>`** creates a regular **table data** cell within a row

A basic table

```
<table>
    <tr>
        <th>Column 1 Header</th>
        <th>Column 2 Header</th>
    </tr>
    <tr>
        <td>Column 1 Data Cell</td>
        <td>Column 2 Data Cell</td>
    </tr>
</table>
```

<> `<th>` attributes

- For accessibility, it's good practice to create an explicit connection for data cells that have multiple headers
- Use scope attributes for table header cells if there are row headers:
    - **`scope="col"`** for table headers that describe a column
    - **`scope="row"`** for table headers that describe a row

<> A table with scope attributes

```
<table>
    <tr>
        <th scope="col">Column 1 Header</th>
        <th scope="col">Column 2 Header</th>
        <th scope="col">Column 3 Header</th>
    </tr>
    <tr>
        <th scope="row">Row 2 Header</th>
        <td>Row 2, Column 2 Data Cell</td>
        <td>Row 2, Column 3 Data Cell</td>
    </tr>
</table>
```

{ } Styling table elements

- All of our box model styles can be applied!
- Some additional styles for cells:
  - **`border-spacing`** puts space between cells

    ```
    table { border-spacing: 4px; }
    ```
  - **`border-collapse`** makes cell borders overlap or sit up against each other each other

    ```
    table { border-collapse: collapse; }
    ```

"Homework" for next time

1. Try out different web font combinations.
2. Add `<div>` elements to your page to group related content.
3. Style your blocks and inline-blocks.
4. Make a table and style its borders and backgrounds.

Optional: HTML & CSS, ch. 6 and 13-14

Next time:

- Review!
- CSS abbreviations
- Using classes and ids for more specific styles
- Pseudo-classes
- Using images and positioning for background styles

Questions? Comments?

- Visit dpersing.github.io/svc for:
  - Class slides
  - Code samples
  - Resources
- Email me: dep@dpersing.com
- Tweet at me: @devonpersing