Styling Chat

We are going to tackle styling that chat page

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Chat</title>
  </head>
  <body>
    <div id="chat-app">
      <ul id="users">
        <li>
          <div class="user">
            <span class="username">Amit</span>
          </div>
        </li>
        <li>
          <div class="user">
            <span class="username">Bao</span>
          </div>
        </li>
      </ul>
      <ol id="messages">
        <li>
          <div class="message">
            <div class="sender">
              <img class="avatar" alt="user-avatar" src="/images/avatar-amit.jpg"/
              <span class="username">Amit</span>
              <span class="timestamp">18:45:32</span>
            </div>
            <p class="message-text">You up?</p>
          </div>
        </li>
        <li>
```

```html
        <div class="message">
          <div class="sender">
            <img class="avatar" alt="user-avatar" src="/images/avatar-bao.jpg"/>
            <span class="username">Bao</span>
            <span class="timestamp">18:46:50</span>
          </div>
          <p class="message-text">Yeah, still working on this INFO6250 work, but
        </div>
      </li>
    </ol>
    <div id="outgoing">
      <form action="/chat">
```

Inline Styling

Every element can have CSS as a property.

```
<div id="chat-app" style="background-color: aqua;">
  App Here
</div>
```

App Here

General Rule: Don't do this.

Why not Inline CSS

- It can't be overridden
    - That's a bug, not a feature
- It can't be applied to other elements, just this one
    - That's a waste of time
- It's a pain to edit
    - In HTML, that's CSS squished inside an *attribute value*
- It's even harder to read
    - Most of programming is reading, not writing

Exception: If you are putting dynamic size/position data on an attribute

Style Tag

CSS can be pulled out into the body (content) of a `<style>` tag

```
<style>
  #chat-app {
    color: green;
    background-color: #C0FFEE;
  }
</style>
<div id="chat-app">
  App Here
</div>
```

App Here

General Rule: Don't do this

Why not Style Tag

- Better than inline, but still annoying to edit
    - CSS inside HTML
    - Longer files
- In one file - CSS can't be shared among many files

Exception: Many sites use a "build" process that combines files - so they will write with CSS in separate files, but it ENDS UP in a style tag. More on this later.

Stylesheet

CSS can be in one or multiple separate files, known as "stylesheets"

In HTML:

```
<link rel="stylesheet" href="/path/to/file.css"/>
```

In CSS file:

```
#chat-app {
  color: green;
  background-color: #C0FFEE;
}
```

- Can share styling across many HTML files
- Does involve extra HTTP Request to load file

Flash Of Unstyled Content

During Rendering Browser reads HTML file and figures out the sizes and appearance of everything.

If it encounters CSS, it applies that styling.

If that CSS is in another file, the browser has to pause to load the file before it can apply the styling.

Anything already displayed stays displayed until that CSS loads, then it changes to show the new styling.

This is called a FOUC (Flash Of Unstyled Content)

Prevent by loading essential CSS in `<head>` - nothing will be displayed to "flash"

Starting Chat Styling

```html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="chat.css"/>
    <title>Chat</title>
  </head>
  <body>
    <div id="chat-app">
    ...
```

chat.css:

```css
#chat-app {
  border: 1px solid black;
}
```

Let's see what that looks like

What was that?

```
border: 1px solid black;
```

This is a `shorthand` property. It is the same as:

```
border-color: black;
border-style: solid;
border-width: 1px;
```

These are actually ALSO shorthand, with each having a version for the four directions.

```
border-right-color: black;
border-right-style: solid;
border-right-width: 1px;
border-bottom-color: black;
border-bottom-style: solid;
border-bottom-width: 1px;
...
```

Tips with shorthand properties

Some say "Always use shorthand properties"

I say "Only use shorthand properties if the meaning is obvious"

Example:

```
padding: 5px 10px 3px;
```

Means:

```
padding-top: 5px;
padding-right: 10px;
padding-bottom: 3px;
padding-left: 10px;
```

I had to try it to be sure - the meaning was not clear.

The CSS box model

Each element is *several* boxes:

- Content (height and width)
- Padding
  - space between the content and border
  - has a width in each direction
- Border
  - has a width in each direction
- Margin
  - space between the border and next element
  - has a width in each direction

CSS Box Model Diagram

margin   -

border   1

padding -

- 1 - 566.571 × 415 - 1 -

-

1

-
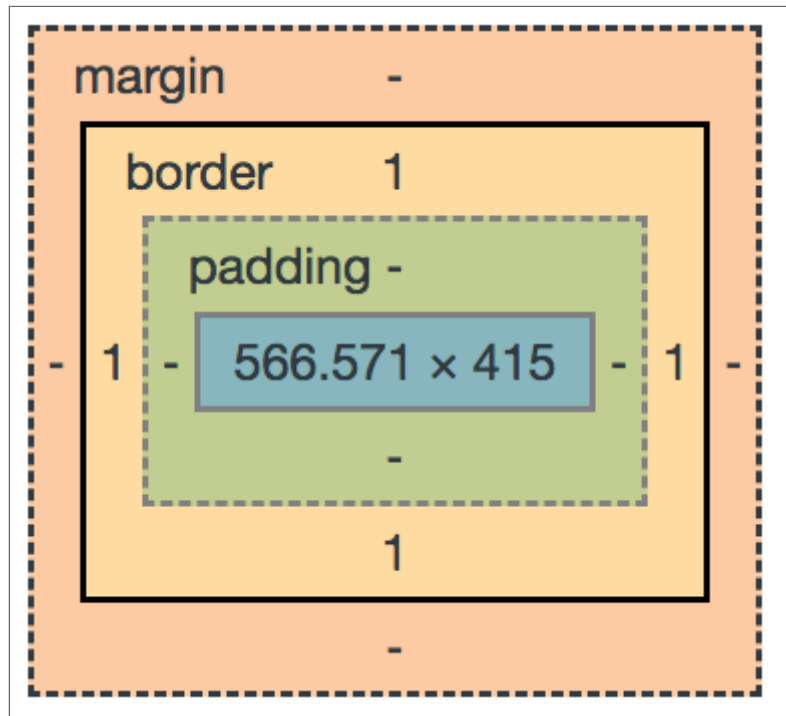
Common CSS issues

Many common desires are complicated in CSS

- Centering? Feel inconsistent
- Vertical Centering? Feels hard!

Newer options are better:

- Flexbox
- CSS Grids

Many complex stylesheets you use and apply special class names to your elements exist

I recommend **AGAINST** using these while learning.

Styling the Chat Parts

When styling, one good approach is to

1. Position the big pieces relative to one another
2. Only then worry about contents
   - Contents may have this same cycle of positioning

Generally, the children of each box know nothing about the whole page except for properties that apply to them

High level positions

Here's our first layout:

- Page will have two sections
    - Top (Users and Messages)
        - Full width
        - Most of the Height
    - Bottom
        - Full width
        - Just a few lines height for the typing

We already have a problem

We want to style two boxes, but that's not what we have:

```
<div id="chat-app">
  <ul id="users">
  </ul>
  <ol id="messages">
  </ol>
  <div id="outgoing">
  </div>
</div>
```

What?! But we worked so hard on the Semantics!

CSS isn't perfect

Semantics are worthwhile, but it still occasionally needs help.

We'll add a wrapping `<div>` purely for the ability to style it

This is known as "**divitis**". You want to keep it mild, but it is often unavoidable to have a little.

It is still worth it to start Semantically - easier to add a few divs than to rip it apart and rebuild

Added Top-Level Wrapper

It can be very hard to NOT name these purely-for-visuals semantically. Even when is for visuals, there is a concept behind it other than just what it looks like.

```
<div id="chat-app">
  <div class="display-panel">
    <ul id="users">
    </ul>
    <ol id="messages">
    </ol>
  </div>
  <div id="outgoing">
  </div>
</div>
```

Classes over ids

Why a class instead of an ID?

IDs must be unique. In modern web dev it is hard to know everything that could be on the page.

It is much easier to avoid conflicts if every bit of functionality restricts itself to one ID

```html
<div id="chat-app">
  <div class="display-panel">
    <ul class="users">
    </ul>
    <ol class="messages">
    </ol>
  </div>
  <div class="outgoing">
  </div>
</div>
```

If needed, you can add the ID to all the selectors so that the classes are effectively unique within the app.

Flexbox

Flexbox is too large to cover here, but worth learning.

Key Concept: Most flex properties affect the **children**.

```css
#chat-app {
  border: 1px solid black;
  display: flex;
  flex-direction: column;
}

.display-panel {
  border: 1px dashed red;
}

.outgoing {
  border: 1px dashed blue;
}
```

Temp Visible Borders can help see what is happening

Repeat with the next level(s)

- User list
  - Only wide enough to show names
- Message List
  - The rest of the width of the panel

```css
.display-panel {
  display: flex;
  flex-direction: row;
}
.users {

  border: 1px dashed red;
}

.messages {
  border: 1px dashed blue;
}
```

Cleanup

Rough positions okay, but still funky

Because of default styling of elements like lists.

"reset stylesheet" removes these defaults, because they can vary between browsers/devices/platforms.

Here we will not use a reset and just manually tweak some settings to see what happens.

But it is risky! What if there is a default in Chrome we are okay with, but it is different elsewhere?

Explicit is better.

User List

You can see the styling in the Chrome Dev tools

`<ul>` has padding, margin, and list-style.

What scope to change? All `<ul>`? Just users?

Depends on the app. Let's keep it focused here.

```css
.users {
  border: 1px dashed red;
  list-style: none;
  padding: 0;
  margin: 0;
}
```

No longer funky, but now cramped. We'll come back.

Leave a solid black border

Messages

Similar issues with the `<ol>`, similar fixes.

Repeat cycle with children.

Message

- left - information about the message (username, avatar, timestamp)
- right - text

Sender Information (requires new div!)

- top - avatar and name
- bottom - timestamp

Messages Divitis

```html
<div class="sender">
  <div class="sender-info">
    <img class="avatar" alt="user-avatar" src="/images/avatar-amit.jpg"/>
    <span class="username">Amit</span>
  </div>
  <div class="message-info">
    <span class="timestamp">18:45:32</span>
  </div>
</div>
```

Rename 'sender' into 'meta-info'

```html
<div class="meta-info">
  <div class="sender-info">
    <img class="avatar" alt="user-avatar" src="/images/avatar-amit.jpg"/>
    <span class="username">Amit</span>
  </div>
  <div class="message-info">
    <span class="timestamp">18:45:32</span>
  </div>
</div>
<p class="message-text">You up?</p>
```

Message CSS

```css
.messages {
  list-style: none;
  padding: 0;
  margin: 0;
}

.message {
  display: flex;
  flex-direction: row;
}

.sender-info {
  border: 1px dashed blue;
}

.message-info {
  border: 1px dashed red;
}
```

Still ugly, but we can see the shape starting to show through.

Minimal Fixed Sizes

DO NOT set all elements to specific sizes

- Looks good only with data that fits
- Looks good only on your screen dimensions and resolution

Instead

- set mins and maxes
- allow for wrapping
- consider scrolling cases
- size non-structural elements

Structural elements

Whenever possible, allow contents to determine size, set min/max widths/height.

Example: The user list width will decide the width of message list area. Set a min/max width on the user list and don't set the size of the message area.

Example: The avatars can be set to a fixed width without much concern - while we might get users of different name lengths, we are comfortable giving the avatar sizes a fixed width.

Next Steps

Given the steps from here, you can cycle over the HTML and CSS to give it full polish

Summary

- Use stylesheets, not inline CSS or style tags
- Use classes, not IDs, except for one top-level ID
- Avoid a FOUC: put key CSS before the content
- Use Shorthand properties sensibly
- Each element is many boxes of height and width
- By default elements will size to fit their needs
- You should use that, not fight that
- Flexbox is worth learning
- Position children, one element at a time
- You may have to group with added divs for visuals
- Give them semantic class names anyway