Styling Chat

```
<div id="chat-app">
  <ul id="users">
    <li>
      <div class="user">
        <span class="username">Amit</span>
      </div>
    </li>
  </ul>
  <ol id="messages">
    <li>
      <div class="message">
        <div class="sender">
          <img class="avatar" alt="avatar of amit" src="/images/avatar-amit.jpg"/>
          <span class="username">Amit</span>
        </div>
        <p class="message-text">You up?</p>
      </div>
    </li>
  </ol>
  <div id="outgoing">
    <form action="/chat">
      <input class="to-send" value="" placeholder="Enter message to send"/>
      <button type="submit">Send</button>
    </form>
  </div>
</div>
```

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 `style` attribute (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 Element

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

```
<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 Element

- 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: Sites may use a "build" process

- Devs edit separate CSS files
- Site is served with CSS contents in `<style>`
- More later

Separate `.css` files

CSS can be in one or multiple separate files

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

- Decides the sizes and appearance of elements
- If it encounters CSS, it applies that styling
- If that CSS is in another file
    - Rendering pauses to load the file
    - Anything already displayed stays displayed
    - After CSS loads, display updates to new styles

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

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

chat.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

Only use shorthand when meaning is clear

- "clear" depends on your team

Example:

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

Means:

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

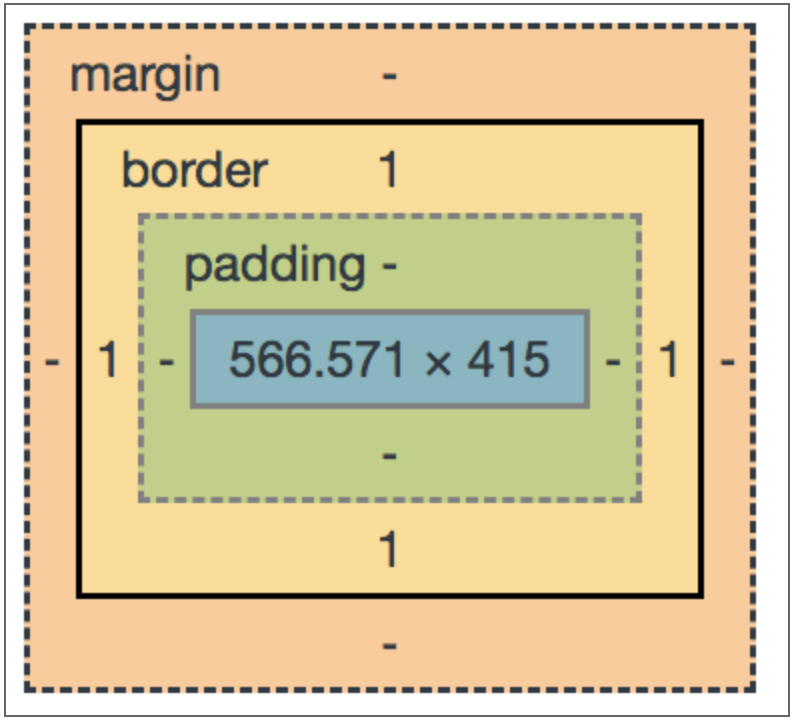
Saved bytes vs confused devs

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



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 you write your own CSS 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
    - They have 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 Layout

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 2 boxes, but we have 3 boxes

```
<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 have value

- Only so many semantic options
- But occasionally needs help

Could add a wrapping `<div>` purely for styling option

- This is known as "**divitis**"
- Keep divitis mild
- Often unavoidable
- Newer options (CSS Grids!) reduce the need

Still worth it to start semantically

Added Top-Level Wrapper

- Hard to semantically name divitis elements
- But still name them for the concept of contents
- This example not actually required
    - CSS Grid could handle it
    - We'll continue with this anyway

```
<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

- `id`s must be unique on page
    - Hard to know full contents of modern sites?
- CSS selectors using `id` override CSS using `class`

Easier to prefer using `class`, use `id` only when needed

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

Can use `id` ancestor to "scope" styling to part of page

Flexbox and Grid - Worth learning more than this

- Flexbox: Arranges children into row(s)/col(s)
- Grid: Places children in grid of row(s) AND col(s)

```
#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

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

.users {
  border: 1px dashed red;
}

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

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.

```
.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
- right - text

Messages Divitis

```
<div class="sender">
  <div class="sender-info">
    <img class="avatar" alt="amit avatar" src="/images/avatar-amit.jpg"/>
    <span class="username">Amit</span>
  </div>
</div>
```

Rename 'sender' into 'meta-info'

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

Message CSS

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

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

Still ugly, but can see shape starting to show through

Minimal Fixed Sizes

DO NOT set all elements to specific sizes

- Requires data that fits
- Requires your screen dimensions and resolution

Instead:

- Use default automatic sizing
- Use mins, maxes, relative sizing
- Allow for wrapping
- Consider scrolling cases
- Size only non-structural elements

Structural elements

- Mostly: Allow contents to determine size
  - Limit (min/max) as needed
- Example:
  - User list width impacts message list width
  - Set a min/max width on the user list
  - Don't set the size of the message area
- Example:
  - Avatar images can be fixed width without much concern
  - Usernames can be different sizes, but images are safer to limit

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 elements
- Generally use classes to style, not types or IDs
- Avoid a FOUC: put core 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 and Grid give powerful layout control
- Arrange elements, one element at a time
- Minimize Divitis, but some may be unavoidable
- Give them semantic class names anyway