First HTML file

- Create a work directory
    - Good habit to learn early: Organize your work
    - Will you be able to find and understand later?
- Create a `index.html` file:

```
Hello World
```

- In Chrome:
    - File->Open File->Select your index.html
    - Windows: Ctrl-O->Select your index.html

Browsers are tolerant

Inspect the rendered page:

- Right-Click -> Inspect
- See the elements in the Elements sub-tab

See all the elements the browser "assumed" for you

YOU DO NOT WANT TO RELY ON THIS!

It will fail you later

Your second HTML file

Edit `index.html`:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Second HTML File</title>
</head>
<body>
  <p>Hello Again, World</p>
  Here
  Is
  More     Text
</body>
</html>
```

HTML Basics

HTML elements may **nest** but may not overlap

```
<div>
  <p> valid </p>
</div>
```

```
<div>
  <p>invalid
  </div>
</p>
```

Whitespace visually collapses to one space

- whitespace = spaces, tabs, new lines
- Newlines in content will be a space!

Real HTML Case

Imagine a chat application

- A list of users
- A list of messages (text, sender, avatar)
- Somewhere to type
- A button to send

Do not think in terms of how it will look

"Semantic" is about what it is and what it means

- NOT what it looks like

Chat - High Level

HTML is a series of nested and/or sibling containers

Page (Document)

- List of Users
- List of Messages
- Typing Area

Chat - some details

Page (Document)

- List of Users
- List of Messages
    - Each Message
        - Avatar
        - Username
        - Text
- Typing Area
    - Input area for message to send
    - Send Button

Chat - structural bones

(contents of `<body>`)

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

Why a base `<div>` for the app at all, why not just put contents in `<body>`?

Why are some `<ol>` and `<ul>` (ordered/unordered lists) and some `<div>`?

Why these elements?

Why base `<div>` and not just contents in `<body>`?

- Allows contents to be managed as a unit
    - Formatting
    - Add to page (controls, non-app details, ads, etc)

Why are some `<ol>`, `<ul>` (lists) and some `<div>`?

- Semantics
    - `<ul>` contents are related to each other
    - `<div>` contains unrelated contents

How to decide on elements`

Why `<ol>` vs `<ul>`?

- Does order matter?

Why `<div>` and not `<p>` or `<span>`?

- `<p>` is a paragraph
- `<span>` is a portion of text
- `<div>` is very generic - be specific when you can, but you often can't

MDN is your friend. Google: `MDN ul`

Semantics are arguable

Adding Flesh to the bones

Still need more details

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

Fleshing out User list

```
<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>
```

But Why

```
<li>
  <div class="user">
    <span class="username">Amit</span>
  </div>
</li>
```

Could make `<li class="user">`

- How to have a "user" block outside a list?

Could skip the `<span>`

- What if add more to user
    - avatar? last active? status message?

See how the semantics give options

Fleshing out Message List

```
<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>
      </div>
      <p class="message-text">You up?</p>
    </div>
  </li>
</ol>
```

Arguable, but what arguments?

- `<div class="message">` not just `<li>`?
- `<div class="sender">`?
- `<img class="avatar" .../>` not in a `<div>`?
- `<span class="username">` not a `<p>`
- `<p class="message-text">` a `<p>` and not a `<div>`?
- `message-text` and not `text`?

Fleshing out the outgoing

```
<div id="outgoing">
  <form action="/chat">
    <input
      class="to-send"
      value=""
      placeholder="Enter message to send"
    />
    <button type="submit">Send</button>
  </form>
</div>
```

But Why - Outgoing

```
<form action="/chat">
```

- We'll cover HTML Forms separately

```
<input class="to-send" .../>
```

- Classes for interact data can be hard to name

```
<button type="submit">Send</button>
```

- Might want a class
    - Let wait to minimize complexity
- `foo-button` is NOT a great class name
    - But naming is hard - no better choice?

Seeing it in action

Now we have Semantic HTML

Let's look at an example

- Amit
- Bao

1. Amit

   You up?

2. Bao

   Yeah, still working on this INFO6250 work, but I keep getting distracted by cat videos

Enter message to send Send

That looks terrible

- Semantics ALLOW for flexible styling
    - Mostly from the CSS
        - Which we don't have yet
- Writing **Semantic HTML**
    - Makes it easier to create a certain look
    - And adjust to new needs in the future
- Writing HTML to look a certain way
    - Will look better at first
    - Difficult to make changes to
    - May break on different devices/platforms

Summary

- Browsers are tolerant
    - Don't rely on the tolerance
- HTML whitespace will collapse to single space
- HTML whitespace is for humans (99% of the time)
- Be Semantic without considering appearance
    - Semantic is always better
- Think about the data when considering structure
- Be as specific as you can
    - Sometimes you that's not very specific
    - Semantics take work

Summary - Part 2

Requirements for this Course:

- Tag names, attributes in **kebab-case**
- HTML attribute values with no space around `=`
- Attribute values quoted with double quotes (`"`)
- Class names are **kebab-case** (or BEM-style)
    - All lowercase, with hyphens (`kebab-case`)
    - NOT `camelCase`, `MixedCase`, or `snake_case`
- Name classes for what the element represents
    - NOT what it will look like
    - **Semantic** class names