COMP6080: Web Front-End Programming

Tutorial 1

Sidney Pham
sidney.pham@unsw.edu.au

UNSW

Week 1, Term 3 2020

Who am I?

I'm Sidney Pham, a computer science student in my third year. Sadly, it's my final term.

Who am I?

I'm Sidney Pham, a computer science student in my third year. Sadly, it's my final term.
I've been tutoring COMP1531 Software Engineering Fundamentals since 18S2, and I've also tutored COMP4141 Theory of Computation.

Who am I?

I'm Sidney Pham, a computer science student in my third year. Sadly, it's my final term.
I've been tutoring COMP1531 Software Engineering Fundamentals since 18S2, and I've also tutored COMP4141 Theory of Computation.
I've been doing web development since around 2012, and I've made plenty of stupid web apps since. I'll be a graduate front-end engineer at Canva next year.

Who am I?

I'm Sidney Pham, a computer science student in my third year. Sadly, it's my final term.
I've been tutoring COMP1531 Software Engineering Fundamentals since 18S2, and I've also tutored COMP4141 Theory of Computation.
I've been doing web development since around 2012, and I've made plenty of stupid web apps since. I'll be a graduate front-end engineer at Canva next year. I don't think I can stress strongly enough how much I *don't* know.

Who am I?

I'm Sidney Pham, a computer science student in my third year. Sadly, it's my final term.
I've been tutoring COMP1531 Software Engineering Fundamentals since 18S2, and I've also tutored COMP4141 Theory of Computation.
I've been doing web development since around 2012, and I've made plenty of stupid web apps since. I'll be a graduate front-end engineer at Canva next year. I don't think I can stress strongly enough how much I *don't* know.

I've also interned at AWS as an associate solutions architect in 2018–19 and was at Atlassian as a software developer intern last summer. Feel free to ask me about any of those roles.

Who am I?

I'm Sidney Pham, a computer science student in my third year. Sadly, it's my final term.
I've been tutoring COMP1531 Software Engineering Fundamentals since 18S2, and I've also tutored COMP4141 Theory of Computation.
I've been doing web development since around 2012, and I've made plenty of stupid web apps since. I'll be a graduate front-end engineer at Canva next year. I don't think I can stress strongly enough how much I *don't* know.

I've also interned at AWS as an associate solutions architect in 2018–19 and was at Atlassian as a software developer intern last summer. Feel free to ask me about any of those roles.

You can email me at `sidney.pham@unsw.edu.au`. I'm always happy to be messaged about anything.

Getting to Know Each Other

I'd encourage you to turn your webcams on, if you're comfortable.

Getting to Know Each Other

I'd encourage you to turn your webcams on, if you're comfortable. Please, I'm lonely.

Getting to Know Each Other

I'd encourage you to turn your webcams on, if you're comfortable. Please, I'm lonely.

I want to find out a few things about you all:

1. Name
2. How much experience do you have with front-end development?
3. Why'd you take this course?

About the Course

https://webcms3.cse.unsw.edu.au/COMP6080/20T3/outline

About the Course

`https://webcms3.cse.unsw.edu.au/COMP6080/20T3/outline`

Any questions?

My Approach to Tutorials

Our class is on Tuesday, which is pretty early, so I'll try my best to not make too many assumptions about you having watched the current week's lectures.

My Approach to Tutorials

Our class is on Tuesday, which is pretty early, so I'll try my best to not make too many assumptions about you having watched the current week's lectures. Please remind me if I'm going too fast for you; it's important to remember that this course is relatively introductory.

My Approach to Tutorials

Our class is on Tuesday, which is pretty early, so I'll try my best to not make too many assumptions about you having watched the current week's lectures. Please remind me if I'm going too fast for you; it's important to remember that this course is relatively introductory.

Additionally, I'll often stray from what the tutorial sheet indicates, in order to give you a broader and (IMO) more relevant understanding of the content.

Okay, Maybe Even Further Back

I'm supposed to teach you CSS right away, but in case anyone's lost, here's how it fits into the picture.

Okay, Maybe Even Further Back

I'm supposed to teach you CSS right away, but in case anyone's lost, here's how it fits into the picture.

Webpages are made out of HTML pages that define their structure and content. By itself, that doesn't really make it look any particular way (*a tricky point that I'll get back to later*).

Okay, Maybe Even Further Back

I'm supposed to teach you CSS right away, but in case anyone's lost, here's how it fits into the picture.

Webpages are made out of HTML pages that define their structure and content. By itself, that doesn't really make it look any particular way (*a tricky point that I'll get back to later*). CSS is a language that we can use to make our HTML look pretty.

Let's See an Example

In `box.html`, from the tutorial sheet, can you make a box that:

- is 100px wide;
- is 50px high;
- has 3px of padding;
- has a 1px border that is solid and of colour #333333;
- has 5px of margin on the left and right, and 10px of margin on the top and bottom; and
- has a background colour of `rgb(255, 255, 0)`.

Let's See an Example

In `box.html`, from the tutorial sheet, can you make a box that:

- is 100px wide;
- is 50px high;
- has 3px of padding;
- has a 1px border that is solid and of colour #333333;
- has 5px of margin on the left and right, and 10px of margin on the top and bottom; and
- has a background colour of `rgb(255, 255, 0)`.

(Sidney: explain browser stylesheets.)

Class vs ID

We can add `class` and `id` attributes to HTML elements, e.g. `<h1 class="big-and-bold">...</h1>`.

Class vs ID

We can add `class` and `id` attributes to HTML elements, e.g. `<h1 class="big-and-bold">...</h1>`. These allow us to refer to particular elements in the page (not only from within the HTML document, but in CSS and JS as well).

Class vs ID

We can add `class` and `id` attributes to HTML elements, e.g. `<h1 class="big-and-bold">...</h1>`. These allow us to refer to particular elements in the page (not only from within the HTML document, but in CSS and JS as well). So what's the difference between the two?

Class vs ID

We can add `class` and `id` attributes to HTML elements, e.g. `<h1 class="big-and-bold">...</h1>`. These allow us to refer to particular elements in the page (not only from within the HTML document, but in CSS and JS as well). So what's the difference between the two?

tl;dr — you can only have one element of a particular ID, but as many elements of the same class as you want.

Class vs ID

We can add `class` and `id` attributes to HTML elements, e.g. `<h1 class="big-and-bold">...</h1>`. These allow us to refer to particular elements in the page (not only from within the HTML document, but in CSS and JS as well). So what's the difference between the two?

tl;dr — you can only have one element of a particular ID, but as many elements of the same class as you want.

Browsers will probably still display your page if you disobey, because they're very flexible, but it's still invalid HTML.

Class vs ID

We can add `class` and `id` attributes to HTML elements, e.g. `<h1 class="big-and-bold">...</h1>`. These allow us to refer to particular elements in the page (not only from within the HTML document, but in CSS and JS as well). So what's the difference between the two?

tl;dr — you can only have one element of a particular ID, but as many elements of the same class as you want.

Browsers will probably still display your page if you disobey, because they're very flexible, but it's still invalid HTML. Also, IDs can be accessed using anchor links in URLs;

Class vs ID

We can add `class` and `id` attributes to HTML elements, e.g. `<h1 class="big-and-bold">...</h1>`. These allow us to refer to particular elements in the page (not only from within the HTML document, but in CSS and JS as well). So what's the difference between the two?

tl;dr — you can only have one element of a particular ID, but as many elements of the same class as you want.

Browsers will probably still display your page if you disobey, because they're very flexible, but it's still invalid HTML. Also, IDs can be accessed using anchor links in URLs; and you'll end up with weird behaviour in JavaScript (e.g. `getElementById`) if there are two elements with the same ID.

Semantic HTML

If there's one thing you take away from this tutorial, it's that you should try to write Semantic HTML. That is, your HTML should express the meaning and structure of the webpage (rather than its styling).

Semantic HTML

If there's one thing you take away from this tutorial, it's that you should try to write Semantic HTML. That is, your HTML should express the meaning and structure of the webpage (rather than its styling).

This means using new semantic elements in HTML5 such as `section`, `article`, `footer`, `progress`, `nav`, `aside`, `mark`, and `time`.

Semantic HTML

If there's one thing you take away from this tutorial, it's that you should try to write Semantic HTML. That is, your HTML should express the meaning and structure of the webpage (rather than its styling).

This means using new semantic elements in HTML5 such as `section`, `article`, `footer`, `progress`, `nav`, `aside`, `mark`, and `time`. The overuse of `div` tags is called *div-itis*, and it makes me very sad.

Semantic HTML

If there's one thing you take away from this tutorial, it's that you should try to write Semantic HTML. That is, your HTML should express the meaning and structure of the webpage (rather than its styling).

This means using new semantic elements in HTML5 such as `section`, `article`, `footer`, `progress`, `nav`, `aside`, `mark`, and `time`. The overuse of `div` tags is called *div-itis*, and it makes me very sad.

There are *many* benefits for differently-abled users of your websites (e.g. increased accesibility for screen-readers), as well as non-browser user agents (e.g. crawlers).

Semantic HTML

If there's one thing you take away from this tutorial, it's that you should try to write Semantic HTML. That is, your HTML should express the meaning and structure of the webpage (rather than its styling).

This means using new semantic elements in HTML5 such as `section`, `article`, `footer`, `progress`, `nav`, `aside`, `mark`, and `time`. The overuse of `div` tags is called *div-itis*, and it makes me very sad.

There are *many* benefits for differently-abled users of your websites (e.g. increased accesibility for screen-readers), as well as non-browser user agents (e.g. crawlers). This isn't even to mention separation of concerns and clearer DOM structure, which are themselves very important.

What Does This Mean?

This means, for example, that you should try not to rely on your browser stylesheet to make an `h2` tag look bolder than a `p` tag. Or use
 to create separate paragraphs of text (when you can use multiple <p> tags instead).

What Does This Mean?

This means, for example, that you should try not to rely on your browser stylesheet to make an `h2` tag look bolder than a `p` tag. Or use
 to create separate paragraphs of text (when you can use multiple <p> tags instead).

An `h2` tag is only *less important* than an h1 tag, not necessarily smaller, and if you want it to be smaller, make sure your CSS (or the stylesheet of the browsers that you're targeting) reflects that.

Demo

I'll demonstrate how to use Developer Tools on this site.

Demo

I'll demonstrate how to use Developer Tools on this site.

Things I'd like to show:

1. Image height and width;
2. Border, padding and margin;
3. Modifying properties;
4. Font family;
5. Border radius;
6. `iframe` tags;
7. `:hover` states;

I Love Flexbox

Honestly, trying to align things before Flexbox was hell. For example, see the Clearfix Hack.

I Love Flexbox

Honestly, trying to align things before Flexbox was hell. For example, see the Clearfix Hack.

In a recent project, I remember trying out all the different `display` options, trying different variations of `margin: auto` and even `vertical-align: center` only to have my design still look weird. But once I remembered that Flexbox was a thing, it just worked magically, which I'm way too impressed by.

Learning Flexbox

I honestly think that you shouldn't just rely on the lectures to teach you Flexbox (because it goes too fast).

Learning Flexbox

I honestly think that you shouldn't just rely on the lectures to teach you Flexbox (because it goes too fast).

I recommend The First Search Result On Most Search Engines, Flexbox Froggy, Flexbox Defense and of course any MDN resource.

Learning Flexbox

I honestly think that you shouldn't just rely on the lectures to teach you Flexbox (because it goes too fast).

I recommend The First Search Result On Most Search Engines, Flexbox Froggy, Flexbox Defense and of course any MDN resource.

It's fine if this part of the tutorial goes too fast for you, although I'll do my best.

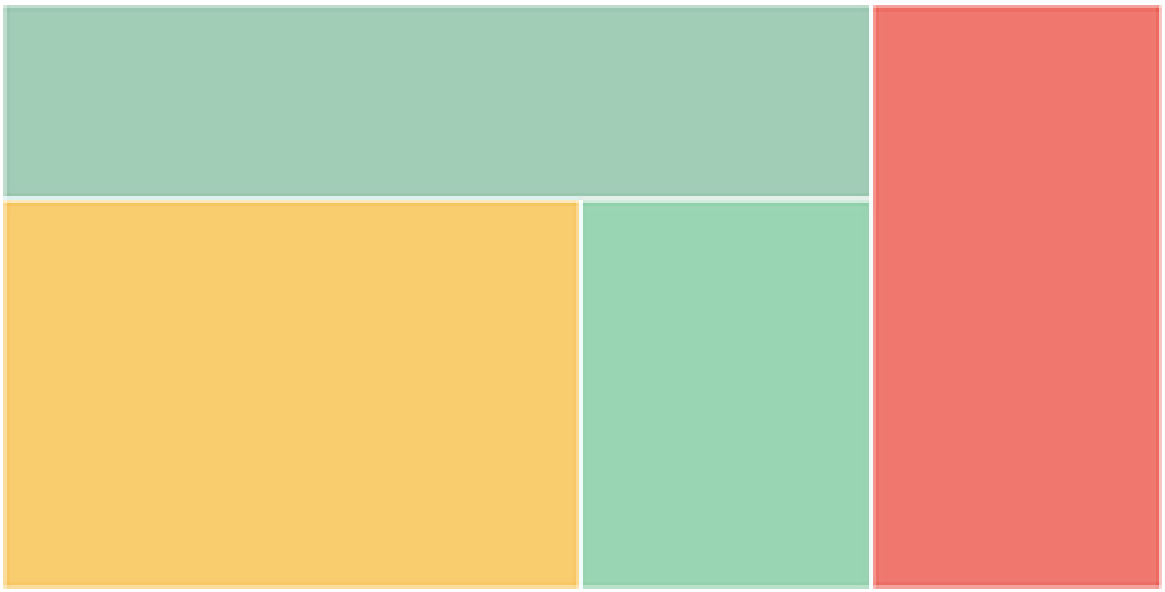
An Exercise

(Note: I don't think this exercise necessarily demonstrates what you'd typically use Flexbox for.)

An Exercise

(Note: I don't think this exercise necessarily demonstrates what you'd typically use Flexbox for.)

Use `div` tags with `flex` properties to create a page that vaguely resembles this image:



Mozilla Developer Network

One of the most useful resources you'll encounter in web development is the MDN web docs by Mozilla. I *highly* recommend using it as a resource when you can.

Tip: Use mdn.io

When looking for references on something like HTTP methods, you might want to Google that and choose the first MDN result. Instead, you can just type (something like) mdn.io/httpmethods into your browser. It behaves kind of like Google's "I'm feeling lucky" but for MDN.