COMP6080: Web Front-End Programming

Tutorial 2

Sidney Pham
sidney.pham@unsw.edu.au

UNSW

Week 2, Term 3 2020

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.

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.

I would *discourage* the use of W3Schools, not because it's terrible anymore (see: W3Fools), but because MDN is almost always *better*.

CSS Units

I had an email last week asking plenty of great questions, and I thought I'd share my advice on choosing CSS units (e.g. px vs em vs rem) with you all.

CSS Units

I had an email last week asking plenty of great questions, and I thought I'd share my advice on choosing CSS units (e.g. px vs em vs rem) with you all.

Disclaimer: This is just my advice, it isn't necessarily endorsed by the course. Take it with a grain of salt.

CSS Units

I had an email last week asking plenty of great questions, and I thought I'd share my advice on choosing CSS units (e.g. px vs em vs rem) with you all.

Disclaimer: This is just my advice, it isn't necessarily endorsed by the course. Take it with a grain of salt.

I honestly use px (over em / rem) most of the time, because I was once told that it's mostly fine, and I haven't run into any terrible issues. It's conceptually very simple, which has a lot of advantages, and after reading these three articles (one, two, three) which propose different viewpoints, I'm relatively certain that primarily using px is mostly fine.

CSS Units (cont.)

To summarise, using relative units for text allows a developer to 'easily' scale up text, e.g. if they want to change the design; but the StackOverflow post says that this is somewhat overhyped a benefit.

CSS Units (cont.)

To summarise, using relative units for text allows a developer to 'easily' scale up text, e.g. if they want to change the design; but the StackOverflow post says that this is somewhat overhyped a benefit.

A lot of resources I've found have discouraged em in favour of rem, if you need to use relative units (except for media queries, which you don't have to worry about yet), e.g. see top comment in the StackOverflow post.

CSS Units (cont.)

To summarise, using relative units for text allows a developer to 'easily' scale up text, e.g. if they want to change the design; but the StackOverflow post says that this is somewhat overhyped a benefit.

A lot of resources I've found have discouraged em in favour of rem, if you need to use relative units (except for media queries, which you don't have to worry about yet), e.g. see top comment in the StackOverflow post.

Also, I know Hayden recommends pt in the lectures for font sizes, but honestly, I just use px (which is used more commonly than pt, according to many sources, e.g. MDN).

Exercise: Sorry, Richard...

Take a look at `http://www.cse.unsw.edu.au/~richardb/email.html` for review. Try to find all the things that could be improved.

Some Obvious Things

A good start is to use an HTML Validator.

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;
2. The CSS file is `style.html`;

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;
2. The CSS file is `style.html`;
3. Inconsistent capitalisation and indentation;

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;
2. The CSS file is `style.html`;
3. Inconsistent capitalisation and indentation;
4. The keyword meta tag isn't terribly useful (see: this article);

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;
2. The CSS file is `style.html`;
3. Inconsistent capitalisation and indentation;
4. The keyword meta tag isn't terribly useful (see: this article);
5. Unclosed HTML tags;

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;
2. The CSS file is `style.html`;
3. Inconsistent capitalisation and indentation;
4. The keyword meta tag isn't terribly useful (see: this article);
5. Unclosed HTML tags;
6. Use of <`blockquote`> and other tags to describe style, instead of CSS;

Some Obvious Things

A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;
2. The CSS file is `style.html`;
3. Inconsistent capitalisation and indentation;
4. The keyword meta tag isn't terribly useful (see: this article);
5. Unclosed HTML tags;
6. Use of `<blockquote>` and other tags to describe style, instead of CSS;
7. Use of HTML attributes (e.g. `bgcolor`) instead of CSS;

Some Obvious Things

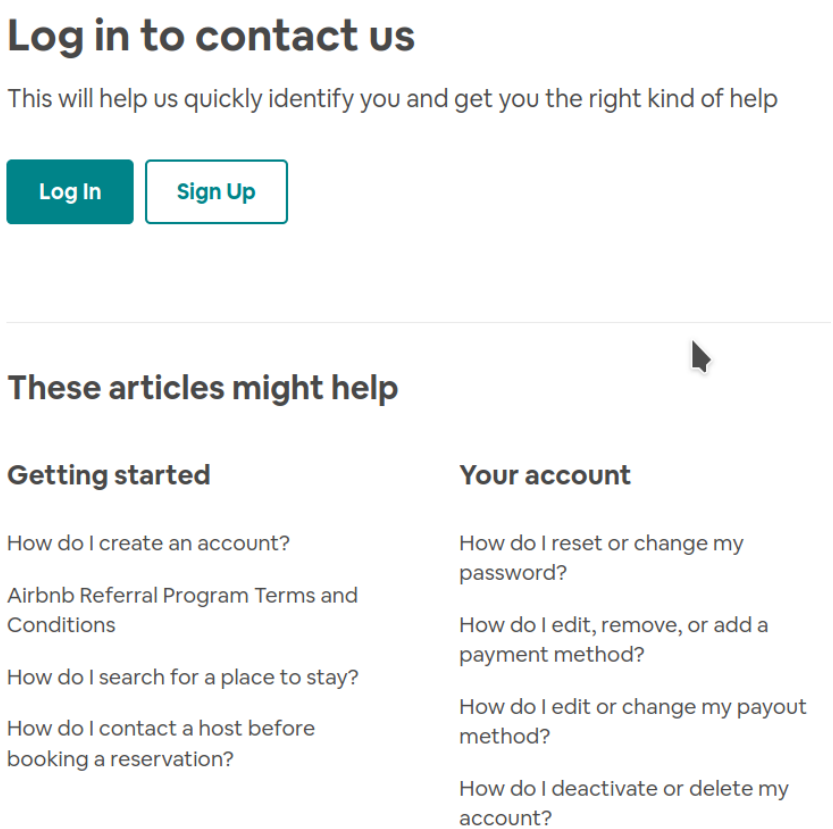
A good start is to use an HTML Validator. Not everything it tells you matters *that* much, so take it with a grain of salt.

However, some things I'd improve are:

1. No `DOCTYPE`;
2. The CSS file is `style.html`;
3. Inconsistent capitalisation and indentation;
4. The keyword meta tag isn't terribly useful (see: this article);
5. Unclosed HTML tags;
6. Use of <`blockquote`> and other tags to describe style, instead of CSS;
7. Use of HTML attributes (e.g. `bgcolor`) instead of CSS;
8. It's not very semantic.

Exercise: AirBnB

This is a page from the AirBnB website. Create a page that uses HTML and CSS to produce a page that looks as close to the image as possible. Hint: The font-family is Circular, which you don't have to replicate. Focus on structure, spacing, and font-sizes.



What is Node.js

It's essentially an environment to run JavaScript in the terminal (cf. web browsers).

What is Node.js

It's essentially an environment to run JavaScript in the terminal (cf. web browsers). For example, you might write a front-end for the browser and a back-end/server to run in Node.js, both written in JavaScript.

What is Node.js

It's essentially an environment to run JavaScript in the terminal (cf. web browsers). For example, you might write a front-end for the browser and a back-end/server to run in Node.js, both written in JavaScript.

A Small Gotcha

Conveniently, there's a REPL (Read-Eval-Print-Loop) in Node.js (the `node` command). Much like the console in Developer Tools, this feels like a pretty straightforward JS environment at first glance. But it's not, really.

What is Node.js

It's essentially an environment to run JavaScript in the terminal (cf. web browsers). For example, you might write a front-end for the browser and a back-end/server to run in Node.js, both written in JavaScript.

A Small Gotcha

Conveniently, there's a REPL (Read-Eval-Print-Loop) in Node.js (the `node` command). Much like the console in Developer Tools, this feels like a pretty straightforward JS environment at first glance. But it's not, really. It's *convenient*, but we can't and shouldn't expect such tools to always behave stricly as a pure JS environment because that's not the purpose of these tools.

What is Node.js

It's essentially an environment to run JavaScript in the terminal (cf. web browsers). For example, you might write a front-end for the browser and a back-end/server to run in Node.js, both written in JavaScript.

A Small Gotcha

Conveniently, there's a REPL (Read-Eval-Print-Loop) in Node.js (the `node` command). Much like the console in Developer Tools, this feels like a pretty straightforward JS environment at first glance. But it's not, really. It's *convenient*, but we can't and shouldn't expect such tools to always behave stricly as a pure JS environment because that's not the purpose of these tools. (See: Ch1 of YDKJS.)

Exercise: A Simple Program

Write a program to read in a text file (one of `data1.txt`, `data2.txt`, or `data3.txt`) which contains integers on some of the lines. The program will determine the lowest and highest valued number, and print out the range (i.e. difference between max and min).

Exercise: A Simple Program

Write a program to read in a text file (one of `data1.txt`, `data2.txt`, or `data3.txt`) which contains integers on some of the lines. The program will determine the lowest and highest valued number, and print out the range (i.e. difference between max and min). Let's try to write it as simply as possible, first. (Sidney: Demo.)

Exercise: A Simple Program

Write a program to read in a text file (one of `data1.txt`, `data2.txt`, or `data3.txt`) which contains integers on some of the lines. The program will determine the lowest and highest valued number, and print out the range (i.e. difference between max and min). Let's try to write it as simply as possible, first. (Sidney: Demo.)

Now, how can we write this much more nicely? Notice how it takes quite a few lines to express a pretty simple concept?

Improvements

Typically, simple transformations can be done using functional programming, which JavaScript supports (yay!).

```
fs = require('fs');
const data = fs.readFileSync('data.txt', 'utf8');
const lines = data
                .split("\n").filter(l => l !== '')
                .map(l => parseInt(l));

console.log(
  `The range is ${Math.max(...lines) - Math.min(...lines)}`
);
```

Template Literals

Notice the string that looks like `` `string text ${expression} string text` ``? This is called a template literal or template strings in older ES spec versions.

Template Literals

Notice the string that looks like `` `string text ${expression} string text` ``? This is called a template literal or template strings in older ES spec versions.

They're super useful for string interpolation (much like Python f-strings) or multi-line strings.

Template Literals

Notice the string that looks like `` `string text ${expression} string text` ``? This is called a template literal or template strings in older ES spec versions.

They're super useful for string interpolation (much like Python f-strings) or multi-line strings. I recommend using them in favour of string concatenation, where you can.

Template Literals

Notice the string that looks like `` `string text ${expression} string text` ``? This is called a template literal or template strings in older ES spec versions.

They're super useful for string interpolation (much like Python f-strings) or multi-line strings. I recommend using them in favour of string concatenation, where you can.

There's even a more advanced form that looks like `` tag`string text ${expression} string text` ``, called tagged template literals. (See: MDN.)