

INTRODUCTION TO FRONT-END WEB DEVELOPMENT

Building your web portfolio

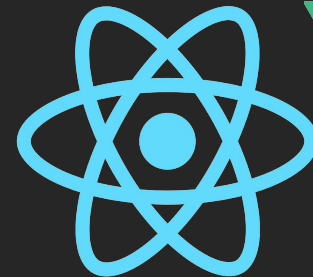
Foo Yong Qi

yongqi@u.nus.edu

HTML



CSS



SCOPE

Getting Started

HTML5 and CSS3

Basic JavaScript (ES6)

GitHub Basics

Static Site Deployment

Front-end JavaScript Frameworks (Angular, React.js, Vue.js)

ABOUT ME

ISE3 & CS

Recreational front-end web developer (PWA)

Favourite tools and technologies:

Vue.js, Vuex, Nuxt.js, Quasar, Vuetify.js, Element UI, Firebase, Node.js, Google Cloud Platform (GCP)



WHAT IS WEB DEV?

Front-end web development is the practice of converting data to a graphical interface through the use of **HTML**, **CSS** and **JavaScript**, so that users can view and interact with that data.

Some call it "client-side development". Front-end web development is the process of creating a website that viewers (clients) can interact with.


WHY WEB DEV?

Hiring season (soon)!

Would you rather submit this...

Foo Yong Qi

Block 509, Pasir Ris Street 52, #12-161, S(510509)
(+65) 9658 4080 | yongqifoo@gmail.com



Education

National University of Singapore
Undergraduate [Aug 2017 - Present]
B.Eng. (Hons) in
Industrial & Systems Engineering & Management,
Computer Science
Cumulative Average Point: 4.52 / Honours (Highest Distinction)
Expected year of graduation: 2020
Dean's List, AY18/19 Sem 1

Employment

Uncage
Co-founder, President [Apr 2018 - Present]

- Conceptualized and executed the development of the organisation
- Partnered with existing agencies to provide innovative solutions

Sqkii
Head of Merchandising [Aug 2018 - Present]

- Leading company's merchandising vertical

Business Development [May 2018 - Aug 2018]

- Pitched to potential partners for collaborations and developed projects to achieve company targets

National University of Singapore
Undergraduate Tutor [Aug 2018 - Present]

- Teaching CS1010E and CS2030 (Programming Methodology I & II) in the School of Computing
- Number of semesters taught: 2

Projects

Live Firing Systems Optimization
Developer [Jul 2016 - Mar 2017]

- Spearheaded and developed spreadsheet (MS Excel) for use in Singapore Armed Forces Multi-Mission Range Complex (MMRC) to optimize the live firing process
- Estimated annual cost savings from program: S\$75,000

#HuntTheMouse Singapore 2018
Team Member [Dec 2018 - Present]

- Part of the team executing and strategising for Sqkii's island-wide campaign where S\$100,000 was hidden somewhere in Singapore.
- Developed a program to automate the generation of strategic

About Me

I like ducks and I'm horrible at sales.

Technical Skills

- MS Office (Excel, Powerpoint, Word)
- Programming (Java, C, Python, HTML, CSS, JavaScript, R)
- Tableau (Data Visualization)
- Stella Architect (Systems Design)

Awards

Singapore Armed Forces

- BMTC '2' Coy Platoon Best Recruit
- SCS 'L' Coy Platoon Best Soldier
- 3AMB Commanding Officer's Coin
- 3AMB Base Sergeant Major's Coin
- Letter of Commendation

Things I've Tried To Do But Failed Miserably

A music production company
Producer

I tried to start my own production company since I love music. Though some of my products were featured on Spotify, I failed to scale it into a proper business.

A hair products company
Co-Founder

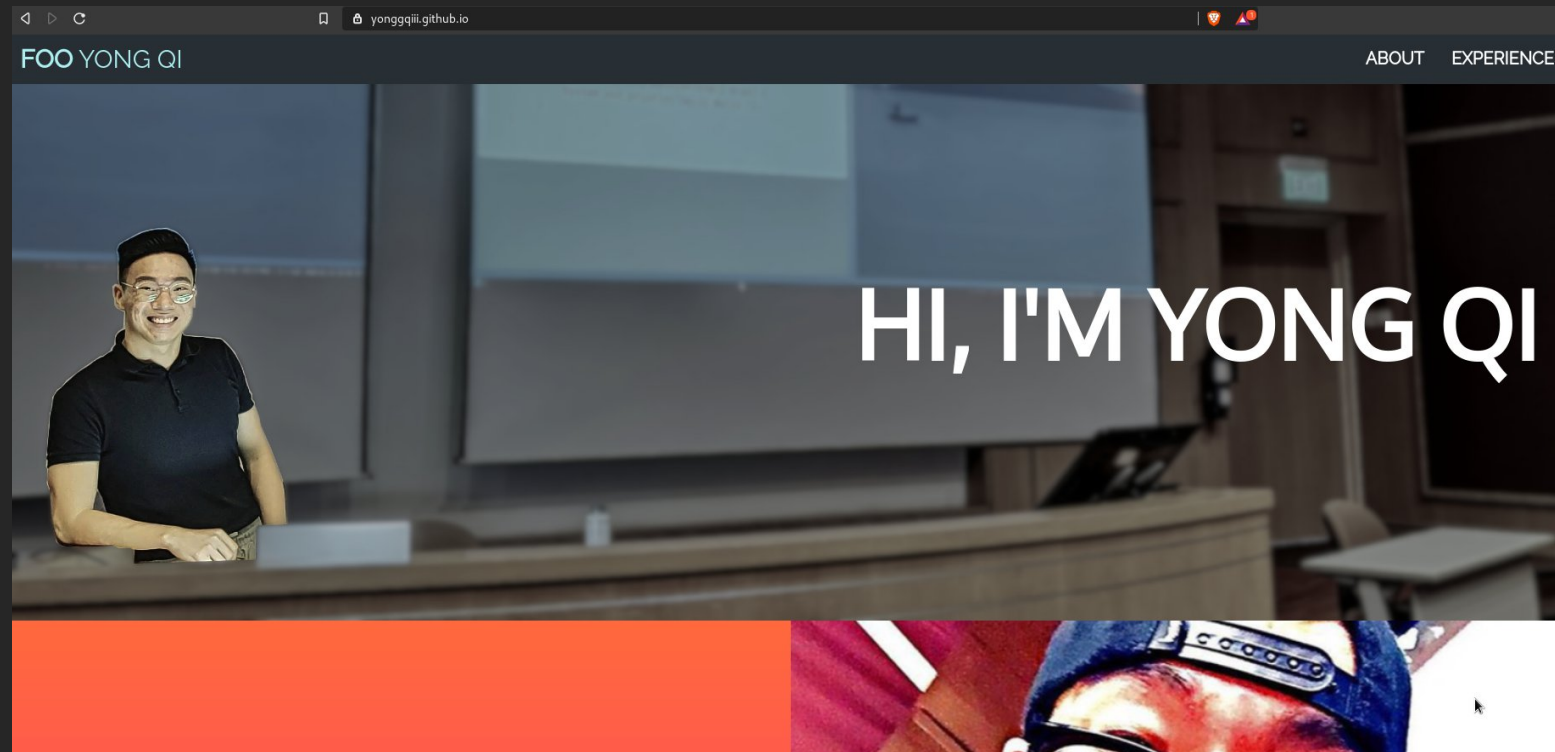
I started making my own hair products since I was broke and I couldn't afford to keep buying pomades. I tried selling these products but was unable to due to a lack of funds and me not being resourceful.

Interests

- Music Production
- Programming
- Acting smart

WHY WEB DEV?

Or this? (Not that my portfolio is particularly nice)



DO I REALLY NEED THIS?

Not really. There are many fantastic website builders available (you need to pay for those though).

But if you're not willing to pay \$100+/year, give web development a try!

THINGS YOU'LL NEED

A Text Editor (Visual Studio Code)

A Web Browser (Google Chrome, Chromium, Brave etc.)

Your resume (or you can create one right now!)

Some Git client (Git BASH for Windows, git for MacOS/Linux)

A GitHub account

VISUAL STUDIO CODE

Visual Studio Code is a robust and powerful **text editor**.

It is popular among developers, especially among front-end web developers.

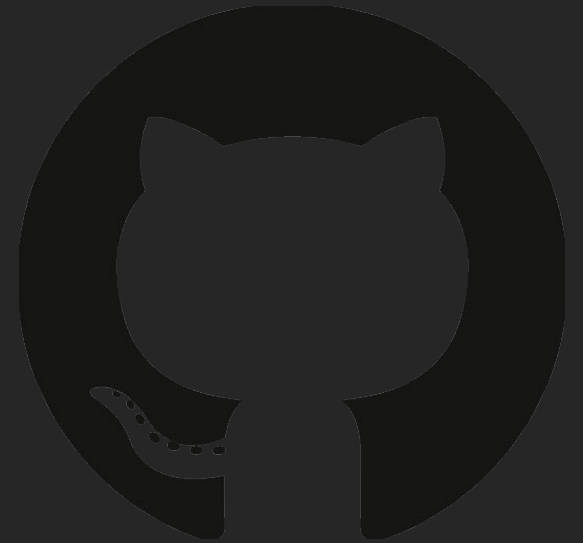
<https://code.visualstudio.com/download>



GIT / GITHUB

Git is a **V**ersion **C**ontrol **S**oftware (VCS). It helps us keep track of the changes we've made to software.

More specifically, we are using **GitHub** to host our site to the web!



GETTING STARTED

Create a folder in your Desktop/Home directory, called
`<github_username>.github.io`

e.g.

`yongggqiii.github.io`

Open that folder in Visual Studio Code to get started.

HTML & CSS BASICS

WHAT IS HTML

HTML stands for **H**yper**T**ext **M**arkup **L**anguage. It is the standard **markup language** for Web pages.

A markup language is designed for the **processing, definition** and **presentation** of text.



WHAT IS HTML

HTML files contain **elements**, or tags, which define different areas of a web page.

Think of writing your HTML as creating an ugly Microsoft Word document.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <!-- Importing google fonts. You may use any other webfonts that you like -->
    <link
      href="https://fonts.googleapis.com/css?family=Didact+Gothic|Kosugi|Oxyc
      rel="stylesheet"
    />
    <!-- Do change the title as you wish. -->
    <title>My Portfolio</title>
    <style>
      html {
        scroll-behavior: smooth;
      }

      /* Setting default margins, padding, height and width for html and body */
      html,
      body {
        margin: 0;
        padding: 0;
        background: #272f35;
      }

      .links {
        color: white;
        text-decoration: none;
        font-weight: bold;
        font-family: Raleway;
      }

      .app-header {
        position: fixed;
        transform: translate3d(0, -2px, 0);
        opacity: 1;
        top: 0;
        left: 0;
        height: 60px;
        width: 100%;
        -webkit-transition: opacity 0.3s ease-out, transform 0.3s ease-out;
        -moz-transition: opacity 0.3s ease-out, transform 0.3s ease-out;
        -o-transition: opacity 0.3s ease-out, transform 0.3s ease-out;
        transition: opacity 0.3s ease-out, transform 0.3s ease-out;
      }
```

HTML ELEMENT

A HTML page contains one html element.

```
<html>
```

```
</html>
```

You should write everything you'd like to be in your page in this element.

HTML TAGS

HTML tags define the beginning and end points of an element.

```
<some-cool-element>
```

```
  Stuff in this element
```

```
</some-cool-element>
```

The first tag marks the beginning of `some-cool-element`, and the second tag ends `some-cool-element`.

HTML TAGS

There are several pre-defined elements that are available to us, for instance, the paragraph element.

```
<p>
```

A cool paragraph.

```
</p>
```

HTML TAGS

However, not all elements require closing tags. For example, line breaks and image elements do not require closing. These are called **singletons**.

```
<br> <img>
```

HTML COMMENTS

To write comments in html, we use: `<!--` and `-->`

```
<!--  
  I am a really  
  cool  
  comment!  
-->
```

HTML HELLO WORLD

Now that we know how tags work, let's create our first Hello World webpage! Write the following in `index.html` in VSCode and start it on a Live Server.

```
<html>  
  Hello World!  
</html>
```

**THANK YOU FOR YOUR
ATTENTION**

HTML HELLO WORLD

We are not done yet. Let's add a doctype declaration, to inform the web browser that we are writing in HTML5.

```
<!DOCTYPE html>  
<html lang="en">  
  Hello World!  
</html>
```

HTML HEAD

The head element in a html document provides important information / imports that describe it.

```
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <meta http-equiv="X-UA-Compatible" content="ie=edge">  
  <title>Document</title>  
</head>
```

HTML TEMPLATE

As a shortcut, type in "!" in an empty html document and press enter. A commonly used HTML5 template will appear.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
</body>
</html>
```

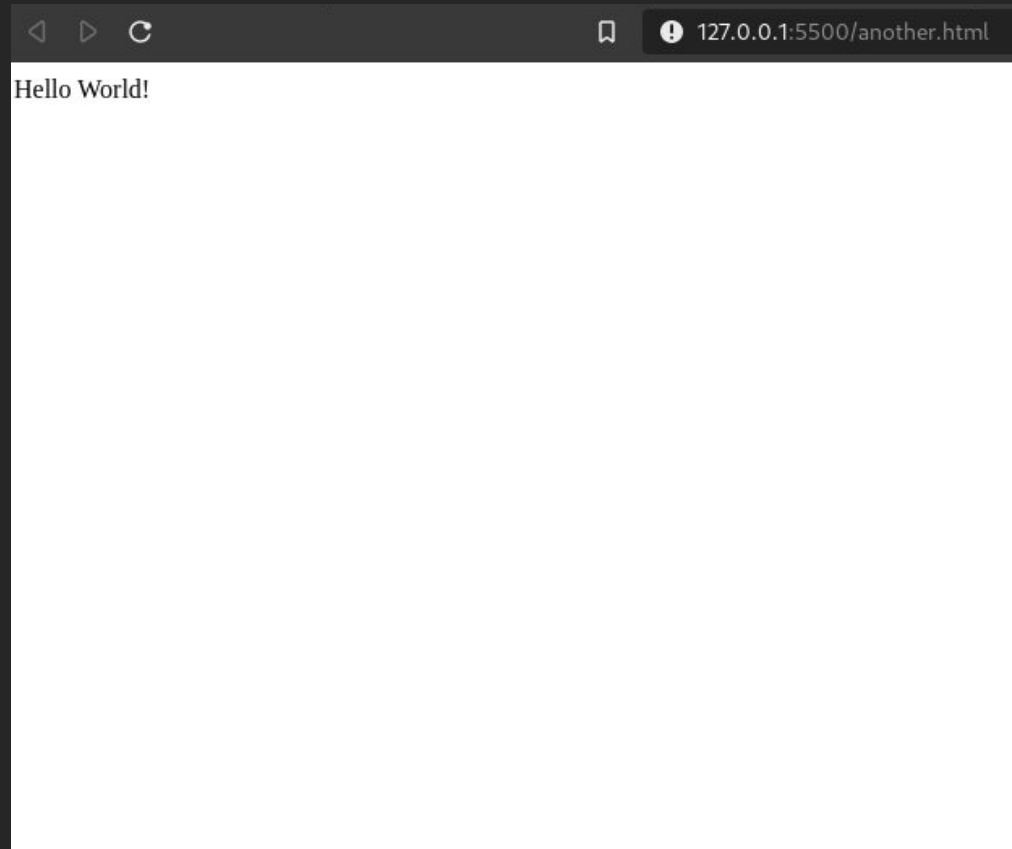

HTML HELLO WORLD

Now, type in Hello World in the body element.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  Hello World!
</body>
</html>
```

HTML HELLO WORLD

Your page should have been rendered on your browser with the live server!



HTML TITLE

Let's also change the title of our page. You may give it a cool title (your name, "My Cool Portfolio", etc.)

You will see that the new title is reflected in the browser.



HTML BODY

Typically, we write all our markup in the body element.

The body element will be where we can start using html elements to render our content.

BASIC HTML ELEMENTS

Strong, emphatic and underline elements do exactly that.

```
<body>  
  <strong>Strong text</strong>  
  <em>Emphatic(italicised) text</em>  
  <u>Underlined text</u>  
  <pre>Preformatted text</pre>  
</body>
```

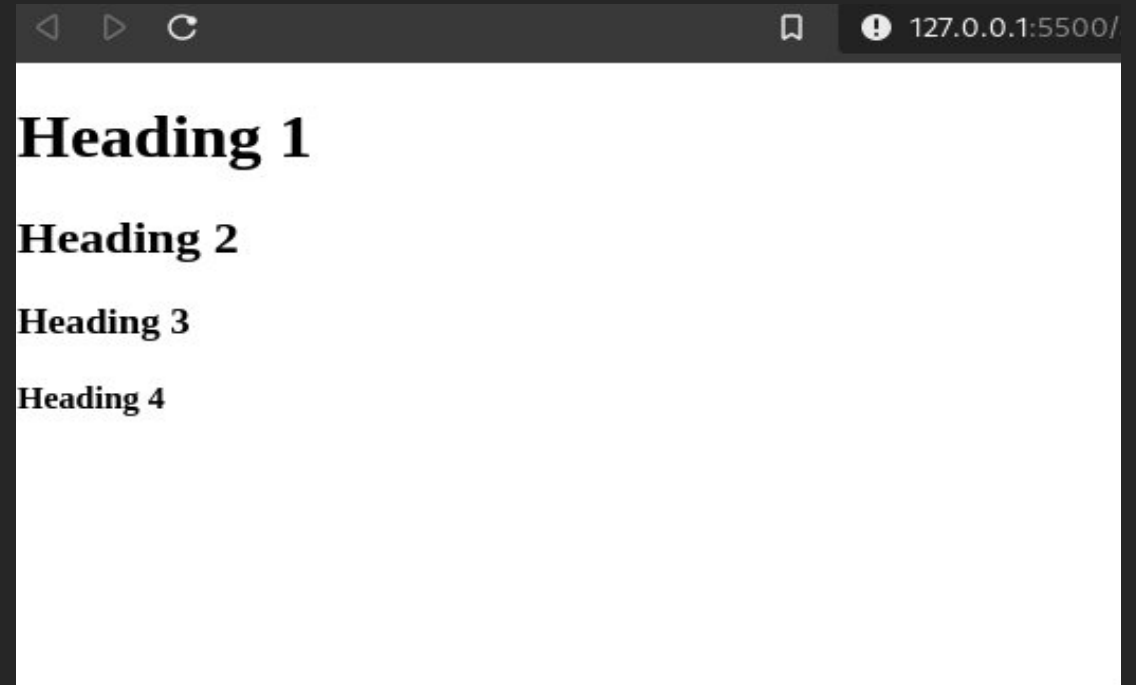
Strong text *Emphatic(italicised) text* Underlined text
Preformatted text

BASIC HTML ELEMENTS

Heading elements define headings (obviously).

Give these a try!

```
<body>  
  <h1>Heading 1</h1>  
  <h2>Heading 2</h2>  
  <h3>Heading 3</h3>  
  <h4>Heading 4</h4>  
</body>
```



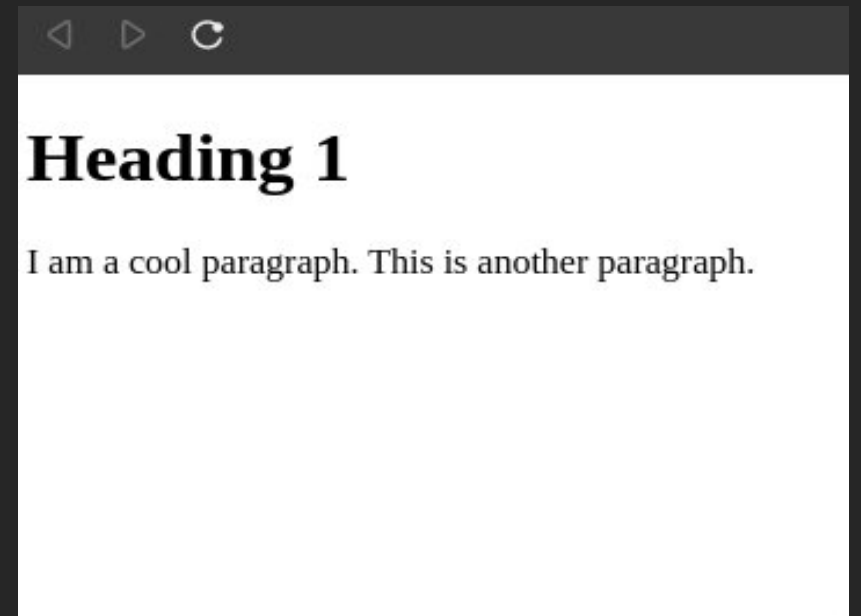
BASIC HTML ELEMENTS

Paragraph elements define paragraphs (obviously).

Give this a try!

```
<body>
  <h1>Heading 1</h1>
  <p>
    I am a cool paragraph.

    This is another paragraph.
  </p>
</body>
```



BASIC HTML ELEMENTS

Notice how there is **no line break** shown? Instead, we should use the line-break tag!

Give this a try!

```
<body>
  <h1>Heading 1</h1>
  <p>
    I am a cool paragraph.
    <br>
    This is another paragraph.
  </p>
</body>
```

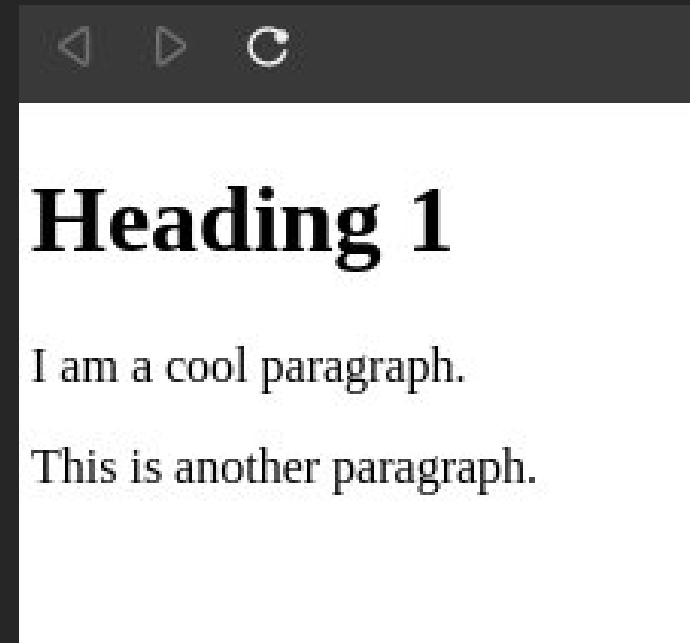
Heading 1

I am a cool paragraph.
This is another paragraph.

BASIC HTML ELEMENTS

Instead, we should be just using another paragraph.

```
<body>
  <h1>Heading 1</h1>
  <p>
    I am a cool paragraph.
  </p>
  <p>
    This is another paragraph.
  </p>
</body>
```



ELEMENT ATTRIBUTES

In HTML, we can pass in values to element attributes to provide extra definitions/behaviour/appearances to them. You will see how to use them shortly.

```
<h1 attr1="value1" attr2="value2">  
    stuff...  
</h1>
```

BASIC HTML ELEMENTS

The anchor element allows us to provide links to areas in our webpage/other websites.

```
<a href="https://www.google.com">Take me to Google!</a>
```

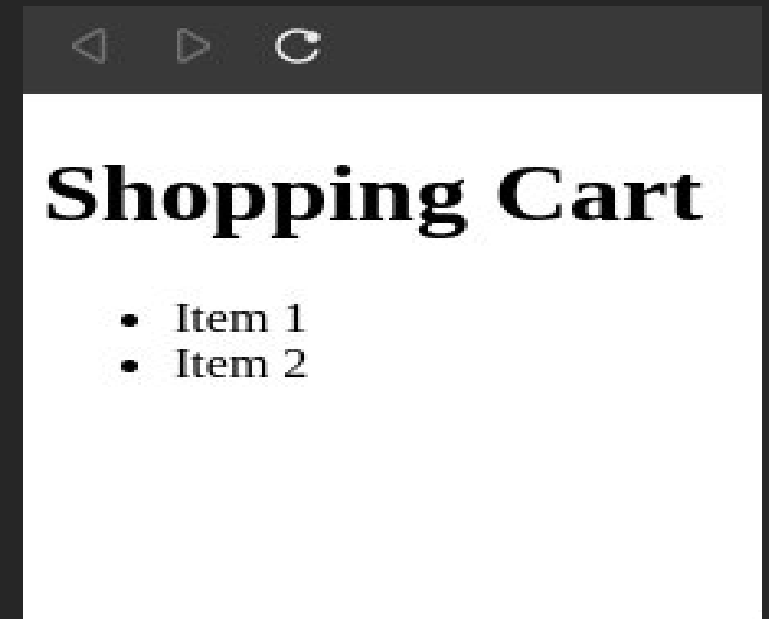
The **href** attribute defines the target reference (link). When the user clicks on the element, they will be taken to that reference.

BASIC HTML ELEMENTS

We can also have ordered or unordered lists. Each item in the list is a list-item element.

Let's try an unordered list.

```
<h1>Shopping Cart</h1>
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```



BASIC HTML ELEMENTS

What about ordered lists?

```
<h1>Shopping Cart</h1>
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
</ol>
```

Shopping Cart

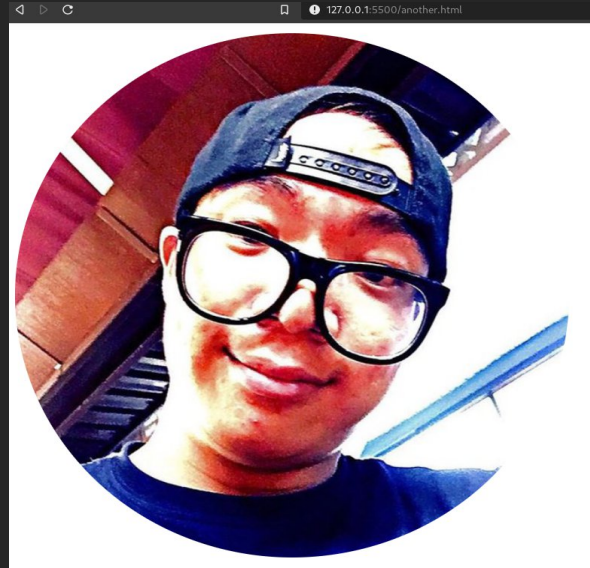
1. Item 1
2. Item 2

BASIC HTML ELEMENTS

So far we've only dealt with plain text, which is boring. With the image element, we can display images as well!

```

```



BASIC HTML ELEMENTS

Here are some more elements you might find useful, namely `header`, `nav` and `footer`

```
<header>
  <nav>
    <ul>
      <li><a href="#about">ABOUT</a></li>
      <li><a href="#experience">EXPERIENCE</a></li>
    </ul>
  </nav>
</header>
```

BASIC HTML ELEMENTS

Even more:

`button, form, table, input, textarea, textarea`

IMPORTANT ELEMENTS

To me (personally), there are the two most important elements in the world when dealing with text.

```
<span>I am a span</span>
```

```
<div>
```

```
  I am a div
```

```
</div>
```

IMPORTANT ELEMENTS

`span` is an inline element. This means that it does not cause a line break to occur.

```
<span>I am a span</span>
```

IMPORTANT ELEMENTS

Meanwhile, `div` is a block element. It is "an entire block" of stuff, which means newlines occur.

```
<div>  
  I am a div  
</div>
```

IMPORTANT ELEMENTS

These elements do not really do anything. That's why they're my favourite; because we can add our own styles from scratch.

```
<span>I am a span</span>
```

```
<div>
```

```
  I am a div
```

```
</div>
```

CLASS & ID

We want to be able to group and identify elements because they have similar/unique behaviour, styles, appearances and so on.

In all elements, you can define its class and id via the `class` and `id` attributes.

```
<some-el class="some-class" id="some-id"></some-el>
```

CLASS & ID

A class is like a group.

Multiple elements can have the same class. The same element can also have multiple classes (separated by spaces).

```
<some-el class="class1 class2"></some-el>
```

```
<some-el class="class1"></some-el>
```

```
<some-el class="class2"></some-el>
```

CLASS & ID

An id is a unique identifier.

Only one element can have a particular id. An element can only have one id (of course).

```
<some-el id="id1"></some-el>
```

CLASS & ID

Of course, all elements can have classes and ids.

```
<some-el class="some-class" id="some-id"></some-el>
```

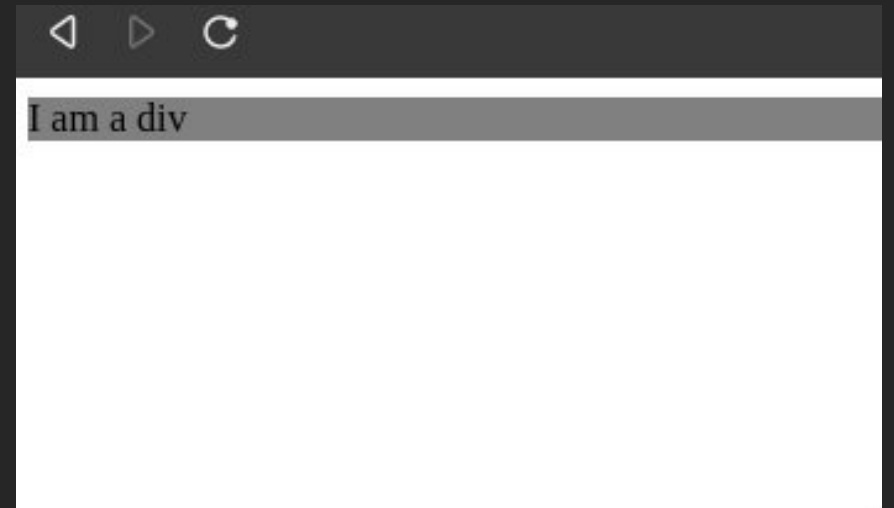

CLASS & ID

Currently, classes and ids don't really do much. However, we can define nice styles to beautify our document, which is made significantly easier with classes and ids.

INLINE STYLES

For now, we can define the styling of elements with the style attribute. Let's try this:

```
<div style="background: grey;">  
  I am a div  
</div>
```



STYLING

We should not be using inline styling very often (unless we want to define the style of **one unique element**). Instead, we can make use of **stylesheets** to style our document efficiently.

```
<div style="background: grey;">  
  I am a div  
</div>  
  
<div style="background: grey;">  
  I am another div  
</div>
```

HANDS-ON EXERCISE

<https://yongggqiii.github.io/basicmarkup.html>

BASIC CSS

WHAT IS CSS?

Cascading **S**tyle **S**heets describe how HTML elements are displayed on screen.

It saves a lot of work by styling groups of elements with the same rules, and even style multiple web pages at once.

Being familiar with CSS is all about **knowledge** (you will be bombarded with information).



BASIC CSS

Let's start by creating a `main.css` file in a new `css` subfolder, then importing it into `index.html` with this statement in the `head` element:

```
<link rel="stylesheet" type="text/css" href="css/main.css">
```

SELECTORS

To define styles for a certain group, we use selectors.

```
selector {  
    attribute1: value1;  
    attribute2: value2;  
}
```

This means that **all elements** identified by the selector will adopt the styles in the curly braces.

SELECTORS

Let's start with some basics.

```
html {  
    background: grey;  
}
```

Now, all html elements (you should only have one) will have a grey background.

SELECTORS

To style a class, we identify it with a `.` at the start.

```
.class1 {  
    background: grey;  
}
```

Now, all elements with `class="class1"` will have a grey background.

SELECTORS

To style an element with an id, we identify it with a # at the start.

```
#id1 {  
    background: grey;  
}
```

Now, all elements with `id="id1"` will have a grey background.

SELECTORS

Suppose we have a `div class="c11"` inside a `div class="c12"`. To style it, we do

```
.c12 .c11 {  
    background: grey;  
}
```

Now, elements with `class="c11"` inside elements with `class="c12"` will have a grey background.

SELECTORS

Suppose we have a `div class="c11 c12"`. To style it, we do

```
.c11.c12 {  
    background: grey;  
}
```

Now, elements with `class="c11 c12"` will have a grey background.

SELECTORS

Suppose we have a `div class="c11 c12"`. To style it, we do

```
.c11.c12 {  
    background: grey;  
}
```

Now, elements with `class="c11 c12"` will have a grey background.

COLORS

Let's try adding colors to our elements.

```
html {  
    background: grey;  
    color: #232147;  
}
```

What does it do?

COLORS

Colors are denoted by their RGB HEX values.

RR GG BB

Where RR, GG and BB are hexadecimal values from 00 to FF (0 to 255 in hexadecimal).

The google color picker is a great tool.

COLORS

In CSS,

#XXX == #XXXXXX

Which means, #444 is shorthand for #444444.

FONTS

We can also change the fonts within the element.

```
html {  
    font-family: Arial, 'Times New Roman', Times, serif;  
    font-size: 16px;  
    font-weight: bolder;  
}
```

Use VSCode's autocompletion to see the possible values for these attributes.

FONTS

Often, we do not want to depend on system fonts (because they might not be installed on a viewer's machine).

Let's try importing a Google webfont in our HTML!

```
<link  
  href="https://fonts.googleapis.com/css?family=Raleway|&display=swap"  
  rel="stylesheet">
```

FONTS

Now, we are sure that the Raleway font will be rendered.

```
html {  
    font-family: Raleway;  
    font-size: 16px;  
    font-weight: bolder;  
}
```

FONTS

Sizing values come in the following syntax:

`[Number][Unit]`

`16px`

FONTS

Here are some units:

Unit	Description
px	pixels (dependent on viewing device)
rem	font-size of root font
vw	1% viewport width
vh	1% viewport height
vmin/vmax	1% viewport min/max
%	relative to parent element

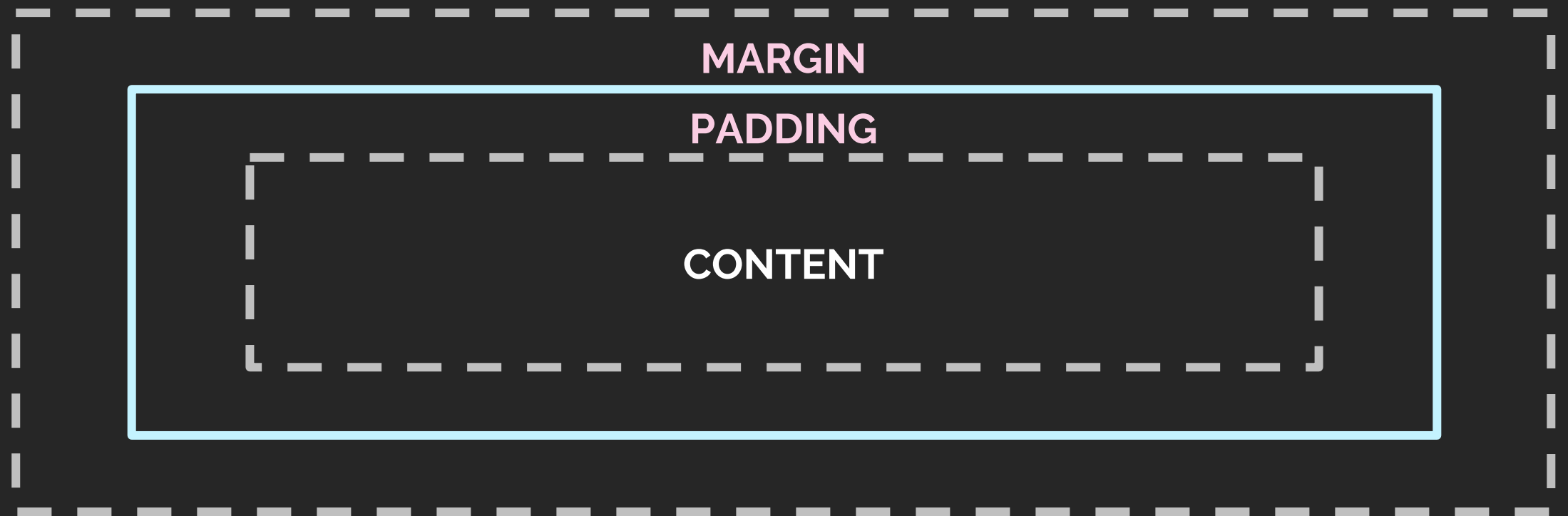
MARGIN/PADDING

Let's try this:

```
.class1 {  
    background: grey;  
    padding: 30px;  
    margin: 30px;  
}
```

MARGIN/PADDING

Margins define the space **around** the element; paddings define the space **inside** the element at the border, **around the content**.



MARGIN/PADDING

They have the following syntax:

```
margin/padding: top rt btm lt;
```

```
margin/padding: top+btm rt+lt;
```

```
margin/padding: top+rt+btm+lt;
```

BORDERS

Let's try this:

```
.class1 {  
    border: 5px solid black;  
    border-radius: 5px;  
}
```

BORDERS

You can define borders with the following syntax:

```
border: thickness style color;
```

border-radius rounds the border with some radius.

DECORATIONS

Let's try this:

```
.class1 {  
    text-decoration: underline overline blue  
    dotted;  
}  
ul {  
    list-style-type: none;  
}
```

DECORATIONS

You can text decorations with the following syntax:

```
text-decoration: line color style;  
list-style-type: type;
```

We often use this for **removing decorations from anchors and list items** (`none`).

POSITION & SIZE

Let's try this:

```
.class1 {  
    background: grey;  
    position: fixed;  
    top: 0;  
    left: 0;  
    height: 50%;  
    width: 100%;  
}
```

POSITION & SIZE

These 3 positions have different effects. Give them all a try!

```
position: fixed | absolute | relative;
```

VISIBILITY

Let's try these 3 **separately**:

```
.class1 {  
    opacity: 1;  
    visibility: hidden;  
    display: none;  
    z-index: 3;  
}
```


VISIBILITY

Attribute	Description
opacity	from 0 to 1, how opaque the element is
visibility	visible hidden, whether element is visible
display	none: element is not visible
z-index	how "forward" element is (lowest value 1)

When opacity is 0, the element can **still be interacted with**.

When visibility is set to hidden, the element **still takes up space**.

When display is set to none, the **element "doesn't exist"**.

FLEX

Let's try making this element a flex container:

```
.class1 {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

FLEX

Flex aims to provide a more efficient to lay out, align, and distribute space among items in a container, even when their size is **unknown/dynamic**.

```
.class1 {  
    display: flex;  
}
```

FLEX

Here are some attributes of a flexbox:

`flex-direction`

`flex`

`flex-wrap`

`justify-content`

`align-items`

ANIMATIONS

In CSS, we can define our own animations.

```
@keyframes fade-in {  
  0% {  
    opacity: 0;  
  }  
  100% {  
    opacity: 1;  
  }  
}
```

ANIMATIONS

We can then add these animations to our elements.

```
.class1 {  
  animation: 1s fade-in infinite linear;  
}
```

TRANSITION

We can define transitions for an element as such. Their use will become clearer with JavaScript.

```
.class1 {  
    transition: attr1 time timingfunction,  
               attr2 time timingfunction,  
               ...;  
}
```

MEDIA QUERIES

Media queries allow us to specify different styles for different media. In this case, we are setting styles for when the viewport width is 1080px or below.

```
@media (max-width: 1080px) {  
  .app-header li {  
    display: none;  
  }  
}
```


MEDIA QUERIES

We can even change the style when the orientation is changed.

```
@media (orientation: landscape) {  
  .app-header li {  
    display: none;  
  }  
}
```

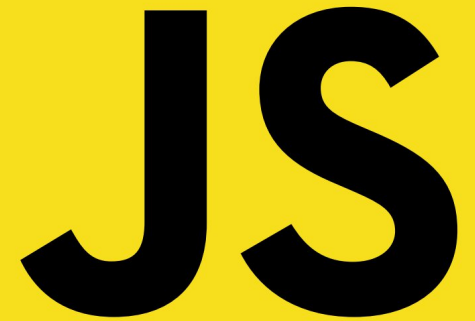
HANDS-ON EXERCISE

Design your portfolio!

WHAT IS JAVASCRIPT

JavaScript (**not Java!**) is the programming language for the web.

On the front end, while HTML is used for content and CSS is used for design, JavaScript is used to define **behaviour**.



WHAT IS JAVASCRIPT

JavaScript is a high-level, single threaded, interpreted (or JIT compiled), dynamically typed, multi-paradigm, prototype-inheritance based, general purpose programming language with a non blocking event loop concurrency model.

For starters, let's copy my template into a blank new `javascript.html` file for learning purposes.



JAVASCRIPT SYNTAX

Assignments:

```
var | let | const x = 10
```

JAVASCRIPT SYNTAX

Mathematical Operators:

1 + 2 - 3 * 4 / 5

JAVASCRIPT SYNTAX

Console output:

```
console.log('Hello World!')
```

JAVASCRIPT SYNTAX

Template literals:

```
var name = 'Bob'
```

```
console.log(`Hello ${name}`)
```


JAVASCRIPT SYNTAX

Logical Operators:

```
true && false // and  
false || true  // or  
!false         // not
```

JAVASCRIPT SYNTAX

Control structures:

```
if (name === 'Bob') {  
    stuff()  
} else if (name === 'Jack') {  
    otherStuff()  
} else {  
    moreStuff()  
}
```

JAVASCRIPT SYNTAX

Iteration:

```
for (var i = 0; i < 10; ++i) {  
    stuff()  
}  
while (i == 10) {  
    console.log('i')  
}  
seq.forEach(callbackFunction)
```

JAVASCRIPT SYNTAX

Functions:

```
function stuff(a, b) {  
    return a + b  
}
```

```
var stuff2 = function(a, b) {  
    return a - b  
}
```

JAVASCRIPT SYNTAX

Arrow functions (ES6):

```
var arrowFunction = (x, y) => x + y
```

JAVASCRIPT SYNTAX

Arrays:

```
var seq = [1, 2, 3]
```

```
var i = seq[2]
```

JAVASCRIPT SYNTAX

Objects:

```
var obj = {  
  'name' : 'Bob',  
  'age' : 25,  
  'hello' : () => console.log('hello')  
}  
var n = obj.name  
var age = obj['age']  
obj.hello()
```

JAVASCRIPT EXERCISE

- The Fair-Play Society has a unique way in selecting its committee members. During the election meeting, all n members present will form a circle, numbered clockwise consecutively, with the current chairperson in position number 1. The chairperson then decides the size of the next committee s .

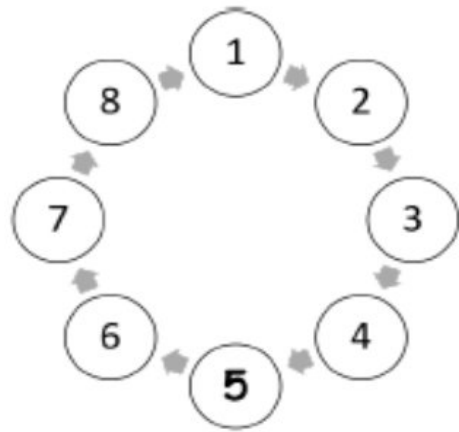
JAVASCRIPT EXERCISE

To form the new committee, the person numbered $s + 1$ will be dropped out from the circle of selection. With one less member, they then re-number the rest of the members present in the circle, with the member that was at position $s + 1$ now at position 1, and the member to his left holding position 2, and so on, till the member to his right holding position $n - 1$.

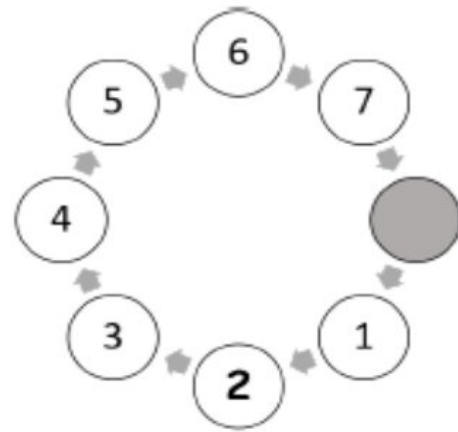
JAVASCRIPT EXERCISE

As an example, consider $n = 8$ and $s = 2$. In particular, the person numbered 5 in Figure (a) will be re-numbered to 2 in Figure (b). Note that the member coloured grey has been dropped out and no longer in the circle of selection. are given an input string. Print the index of the matching closing bracket of the first opening bracket. If there aren't any, print -1.

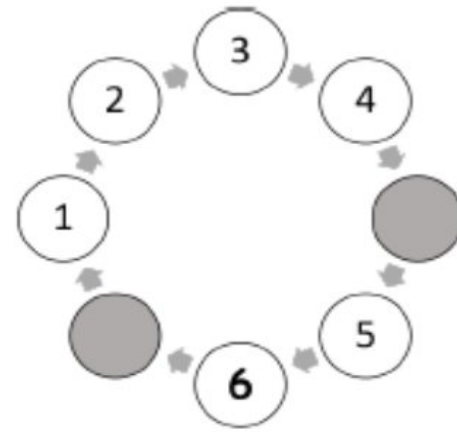
JAVASCRIPT EXERCISE



(a)



(b)



(c)

JAVASCRIPT EXERCISE

The process then continues to eliminate the member at position $s + 1$ again. Continuing from figure (b), the next process results in figure (c), with the person of interest (in bold) now given the number 6. The entire selection process will be repeated until there are exactly s members left in the circle, and these members then form the next committee.

JAVASCRIPT EXERCISE

Task:

Write a program that takes as input the total number of members present n , the size of the new committee s ($2 \leq s < n$), and the position you are in k ($1 \leq k \leq n$). The program then determines if you will become one of the new committee members.

Also print an array of new positions you are in every iteration.

SOLUTION

```
var positions = [];  
while (a > b && c > 0) {  
    a -= 1;  
    c -= b + 1;  
    if (c < 0) {  
        c += a + 1;  
    }  
    positions.push(c);  
}
```

```
console.log(positions);  
if (c == 0) {  
    console.log("Sorry, you are not selected");  
} else {  
    console.log("Congratulations, you are in!");  
}
```

GETTING ELEMENTS

That was the lame part. Let's get into some fun stuff.

In JavaScript, we can get elements from our page.

```
var target = document.getElementById('target')  
var divs = document.getElementsByTagName('div')  
var stuff = document.getElementsByClassName('class1')
```

GETTING ELEMENTS

We can then amend the element as we wish!

```
target.innerHTML = 'I am cool'  
target.className = 'class1 class2'
```


EXERCISE

By only amending index.js, try making the target object disappear.

Hint: the .hidden class has attribute display: none

SOLUTION

```
var TARGET = document.getElementById("target");  
TARGET.className += "hidden";
```

EVENT LISTENERS

We cannot predict when a viewer will interact with elements of a webpage. But, we can define what should be done when an interaction occurs. This is called **event-driven programming**.

We can attach functions to anticipated events. These are called **event listeners**.

EVENT LISTENERS

Let's try adding this in index.js, then resize the window:

```
function hello(e) {  
  console.log(e);  
}  
window.addEventListener("resize", hello);
```

EVENT LISTENERS

We can add event listeners to elements in the page. Let's try an exercise.

Try changing the text in the `target` element to "`Weee!`" when the `click-me` button is pressed.

Hint: You can use the 'click' event trigger.

EVENT LISTENERS

```
document.getElementById("click-me").addEventListener("click",  
() => {  
    document.getElementById("target").innerHTML = "Weee!";  
});
```

Alternatively, you could add `onclick="changeTarget()"` as an attribute and value to the button element in the html.

TRANSITIONS

Now that we know how to amend our page based on triggers, let's try using CSS transitions too!

Try to make the `click-me` button as a toggle for `target` to fade out/in.

You should amend the html / css!

TIMEOUT/INTERVAL

Occasionally, you'd want to wait several seconds to do/repeat something. In this case, we use:

```
function hello() {  
    console.log("hello");  
}  
var once = setTimeout(hello, 1000);  
var repeat = setInterval(hello, 2000);
```


TIMEOUT/INTERVAL

To clear the timeout or interval, use the following:

```
clearTimeout(once)  
clearInterval(repeat)
```

HANDS-ON EXERCISE

Try to implement what you've learnt!

GITHUB BASICS

Now that you've created your awesome webpage, let's host it online. We'll be using GitHub Pages as our static site host.

You should have a GitHub account, Git BASH (Windows) / Git (MacOS / Linux) installed.

REMOTE REPO

In your GitHub account, create an empty repository called

`githubusername.github.io`

for example:

`yongggqiii.github.io`

LOCAL REPO

In Git (BASH), navigate to the directory you've been working in. Then, type

```
git init .  
git config --global user.name "your-username"  
git config --global user.email "your-email"  
git remote add origin https://github.com/your-  
username/your-repo.git
```

STAGE/COMMIT/PUSH

We've just initialised your repositories and configurations. Let's learn the default git workflow.

- 1) Stage
- 2) Commit
- 3) Push

STAGE

Git is simply a version control software. As such, it tracks changes to the files we've made.

To save and deploy our new page, we need to add all the changed files to the commit stage.

```
git add .
```

COMMIT

Once we've added the files we want to stage for commit, it's time to commit (save) those changes.

```
git commit -m "A super cool change!"
```


PUSH

Once we've committed all our changes, we're ready to push our changes to the remote repository.

```
git push origin master
```

You should be prompted to enter your GitHub login credentials as well.

GITHUB-PAGES

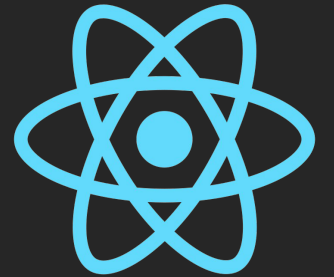
And... you're done, your web portfolio will be live at
<https://your-username.github.io/>

Congrats!

JAVASCRIPT FRAMEWORKS

You've now learnt the building blocks of web pages. What's next?

Enter JavaScript frameworks.



JAVASCRIPT FRAMEWORKS

A framework is a **collection of libraries** that work seamlessly together. In a sense, they are **schools of thought** towards building complex websites.

Currently, the top 3 JavaScript frameworks in use right now are **Angular**, **React.js** and **Vue.js**

ANGULAR

Angular (Google-backed) is a rewrite of AngularJS.

Has a steep learning curve, and requires knowledge of **TypeScript**

Very heavyweight but robust

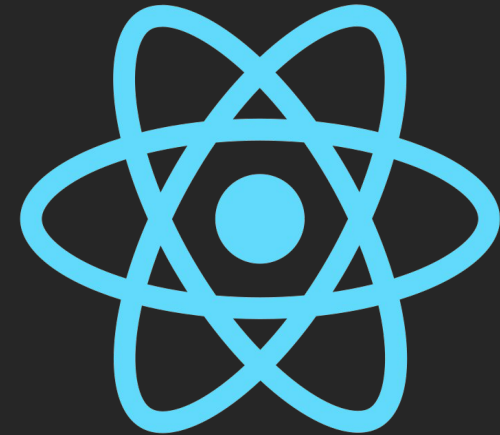


REACT

React (Facebook-backed) is the most popular framework (technically a library).

Very lightweight and modular

Fantastic community and job opportunities



VUE

Vue (open-source, community developed) is the most hippie framework (growing in popularity too)

Modularity falls in between Angular and React

Easiest to pick up due to similarity to HTML, CSS and JavaScript

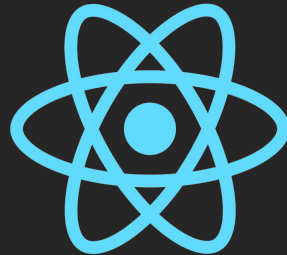


JAVASCRIPT FRAMEWORKS

These frameworks are being used by companies all over the world.



- The Guardian
- PayPal
- Lego
- Netflix



- Facebook
- Instagram
- Khan Academy
- New York Times



- Grammarly
- Adobe
- Nintendo
- GitLab

MY FAVOURITES

I personally use Nuxt.js and Quasar, which are frameworks on top of Vue.js.



Quasar
Framework 

WHAT'S NEXT

Finish up your web portfolio-this should give you a strong conceptual foundation of exploring front-end JavaScript frameworks.

You might also want to explore back-end development, where you deal with servers and databases.

THAT'S ALL!

Hope your new portfolio will land you more jobs this hiring season :)