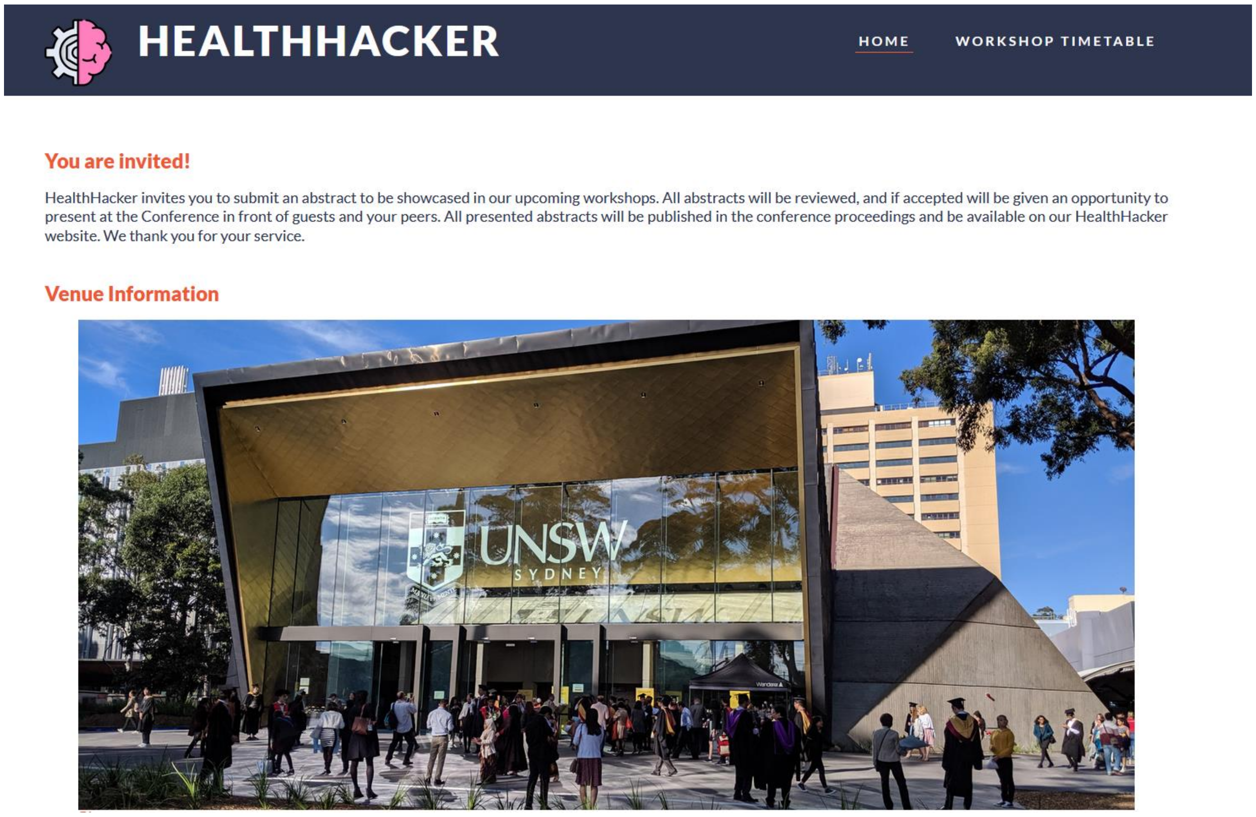
Client-Side Form Validation using JavaScript

Design Choices

- Went with a navy and orange colour combo.
  - Wanted a contrast between the colours and thought these two colours looked good together.
    - Further went with the contrast in the validation form, as well as the hyperlinks at the top, as well.
- Logo went with something simple, from flaticons. Versatile to be used.
- Heading were orange to be an eye-catcher for the user, whilst navy was for the border and a black was used for the text as standard
- The front page had a big picture to really captivate the importance of the event.



You are invited!

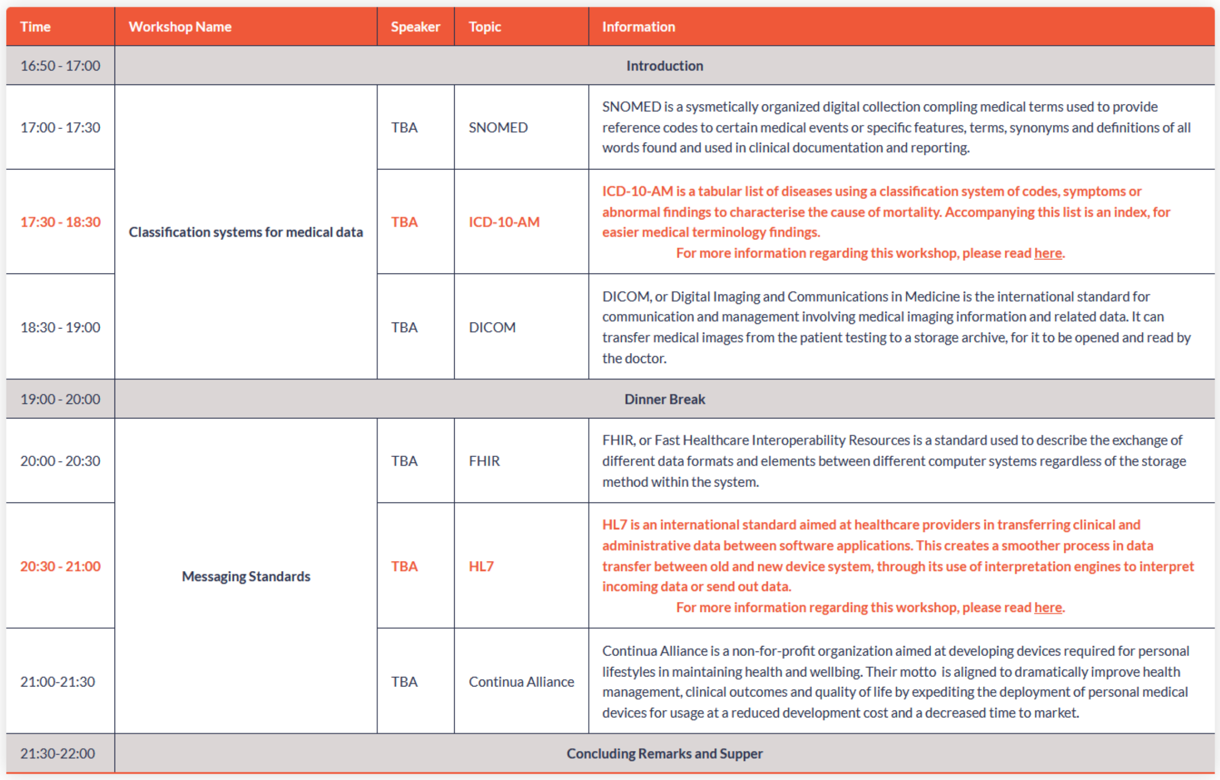
HealthHacker invites you to submit an abstract to be showcased in our upcoming workshops. All abstracts will be reviewed, and if accepted will be given an opportunity to present at the Conference in front of guests and your peers. All presented abstracts will be published in the conference proceedings and be available on our HealthHacker website. We thank you for your service.

Venue Information

Front-page, home of the webpage.

The timetable wanted to play around with the color palette.

- Grey was used to establish break or not important bits and pieces
- Introductions were given to each, as simplified information of the topics at hand.
- Orange was used to highlight mandatory workshops with their timings and topics highlighted as well
- Other major headings were bolded for ease of seeing it.

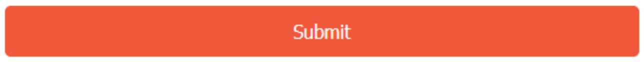


Time	Workshop Name	Speaker	Topic	Information
16:50 - 17:00	Introduction			
17:00 - 17:30	Classification systems for medical data	TBA	SNOMED	SNOMED is a sysmetically organized digital collection compling medical terms used to provide reference codes to certain medical events or specific features, terms, synonyms and definitions of all words found and used in clinical documentation and reporting.
17:30 - 18:30		TBA	ICD-10-AM	ICD-10-AM is a tabular list of diseases using a classification system of codes, symptoms or abnormal findings to characterise the cause of mortality. Accompanying this list is an index, for easier medical terminology findings. For more information regarding this workshop, please read here.
18:30 - 19:00		TBA	DICOM	DICOM, or Digital Imaging and Communications in Medicine is the international standard for communication and management involving medical imaging information and related data. It can transfer medical images from the patient testing to a storage archive, for it to be opened and read by the doctor.
19:00 - 20:00	Dinner Break			
20:00 - 20:30	Messaging Standards	TBA	FHIR	FHIR, or Fast Healthcare Interoperability Resources is a standard used to describe the exchange of different data formats and elements between different computer systems regardless of the storage method within the system.
20:30 - 21:00		TBA	HL7	HL7 is an international standard aimed at healthcare providers in transferring clinical and administrative data between software applications. This creates a smoother process in data transfer between old and new device system, through its use of interpretation engines to interpret incoming data or send out data. For more information regarding this workshop, please read here.
21:00-21:30		TBA	Continua Alliance	Continua Alliance is a non-for-profit organization aimed at developing devices required for personal lifestyles in maintaining health and wellbing. Their motto is aligned to dramatically improve health management, clinical outcomes and quality of life by expediting the deployment of personal medical devices for usage at a reduced development cost and a decreased time to market.
21:30-22:00	Concluding Remarks and Supper			

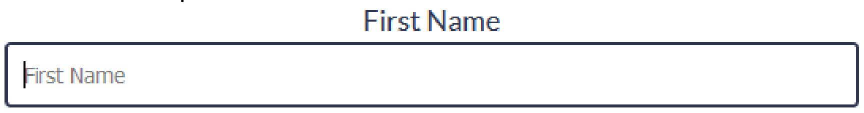
Timetable preview from webpage.

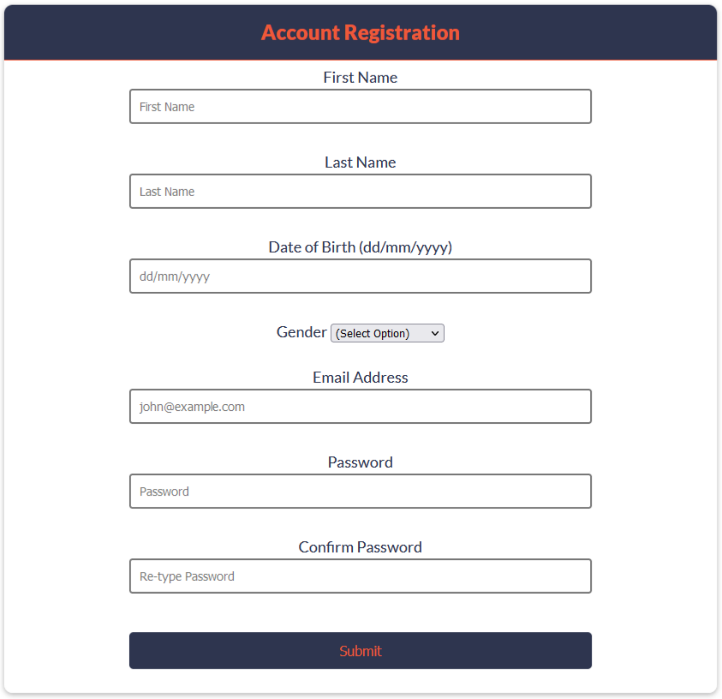
Validation of Form

- To further use the colour palette, the validation was also done in similar palette
  - The difference here is:
    - when hovered over submit button, there would be a change of colours, as main-focus and cursor would change, so its easier for the user to look at



    - when the input field is being clicked in, the input border changes, such that it's the main focus by highlight is navy. A placeholder was placed, so it was to make sure the instructions, if not clear, was double checked. Some included examples of what the acceptable format should be.




Account Registration

First Name

First Name

Last Name

Last Name

Date of Birth (dd/mm/yyyy)

dd/mm/yyyy

Gender (Select Option)

Email Address

john@example.com

Password

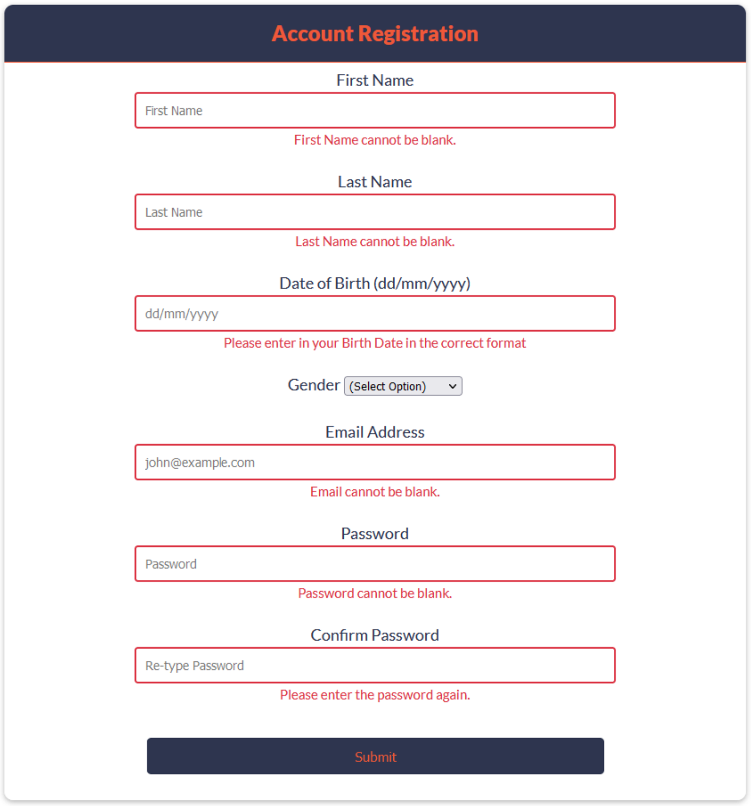
Password

Confirm Password

Re-type Password

Submit

Validation form, unanswered.


Account Registration

First Name

First Name

First Name cannot be blank.

Last Name

Last Name

Last Name cannot be blank.

Date of Birth (dd/mm/yyyy)

dd/mm/yyyy

Please enter in your Birth Date in the correct format

Gender (Select Option)

Email Address

john@example.com

Email cannot be blank.

Password

Password

Password cannot be blank.

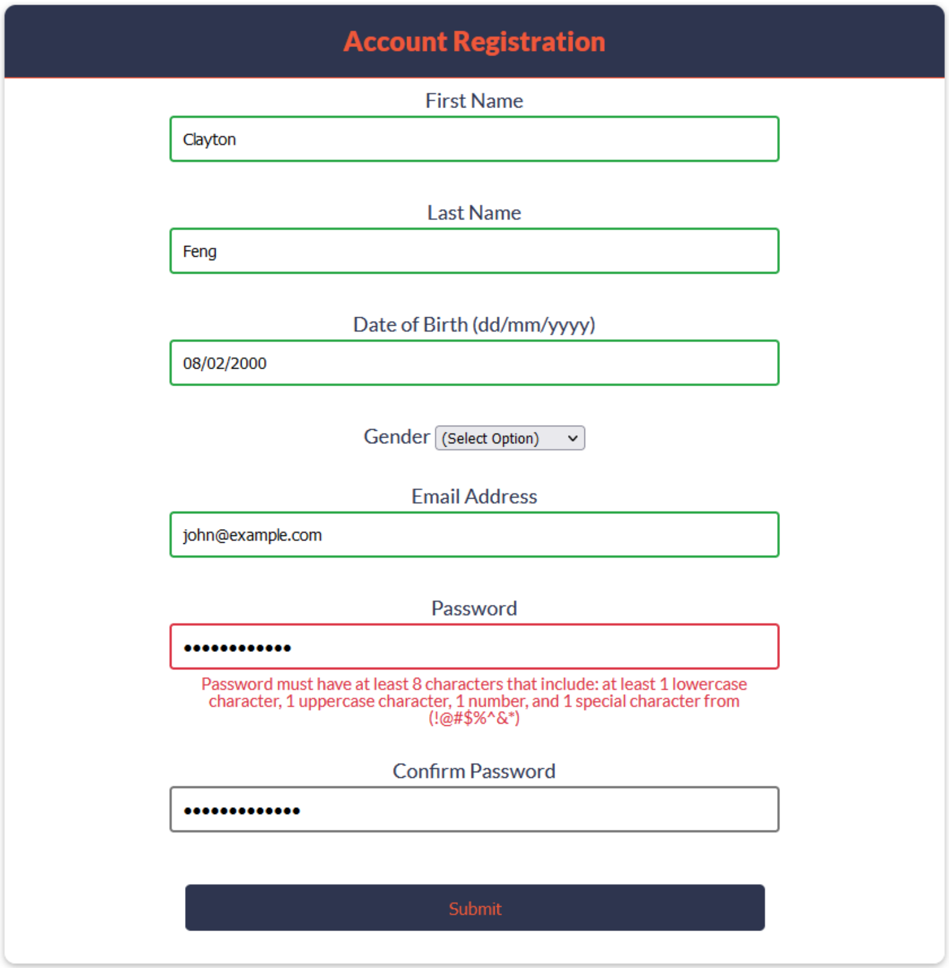
Confirm Password

Re-type Password

Please enter the password again.

Submit

Validation form, unanswered with preview of all errors.


Account Registration

First Name

Clayton

Last Name

Feng

Date of Birth (dd/mm/yyyy)

08/02/2000

Gender (Select Option)

Email Address

john@example.com

Password

••••••••••••

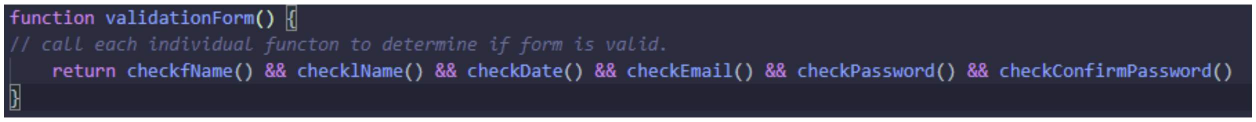
Password must have at least 8 characters that include: at least 1 lowercase character, 1 uppercase character, 1 number, and 1 special character from (!@#$%^&*)

Confirm Password

•••••••••••••

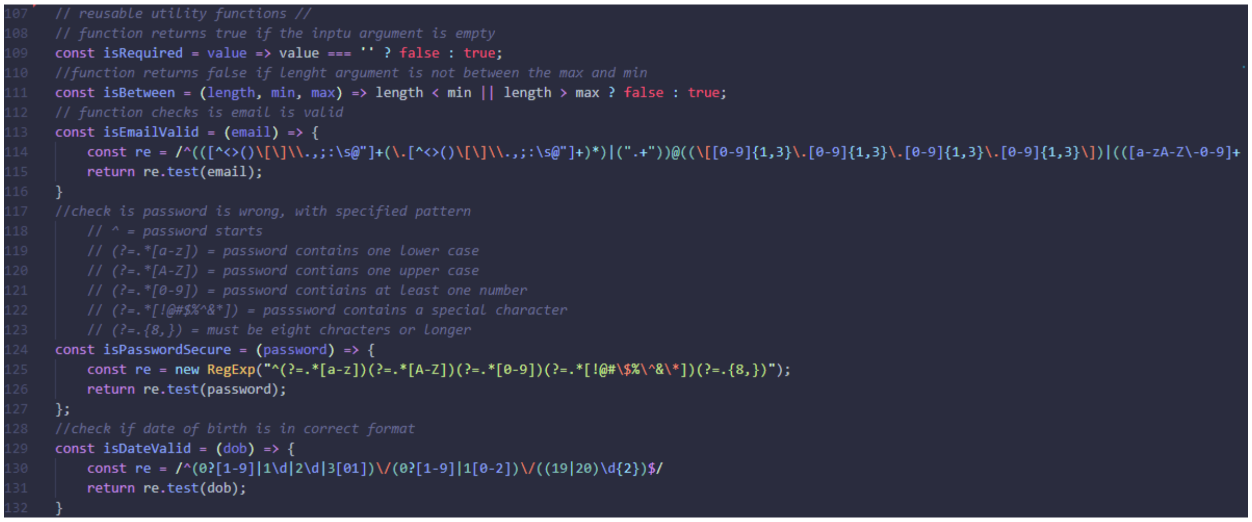
Submit

Password has greater restrictions for more security. Normally standard for most websites. Note: Account registration validation works FROM TOP → DOWN. This is best, to confirm via code that all validations are met per input field.


```
function validationForm() {
// call each individual functon to determine if form is valid.
    return checkfName() && checklName() && checkDate() && checkEmail() && checkPassword() && checkConfirmPassword()
}
```

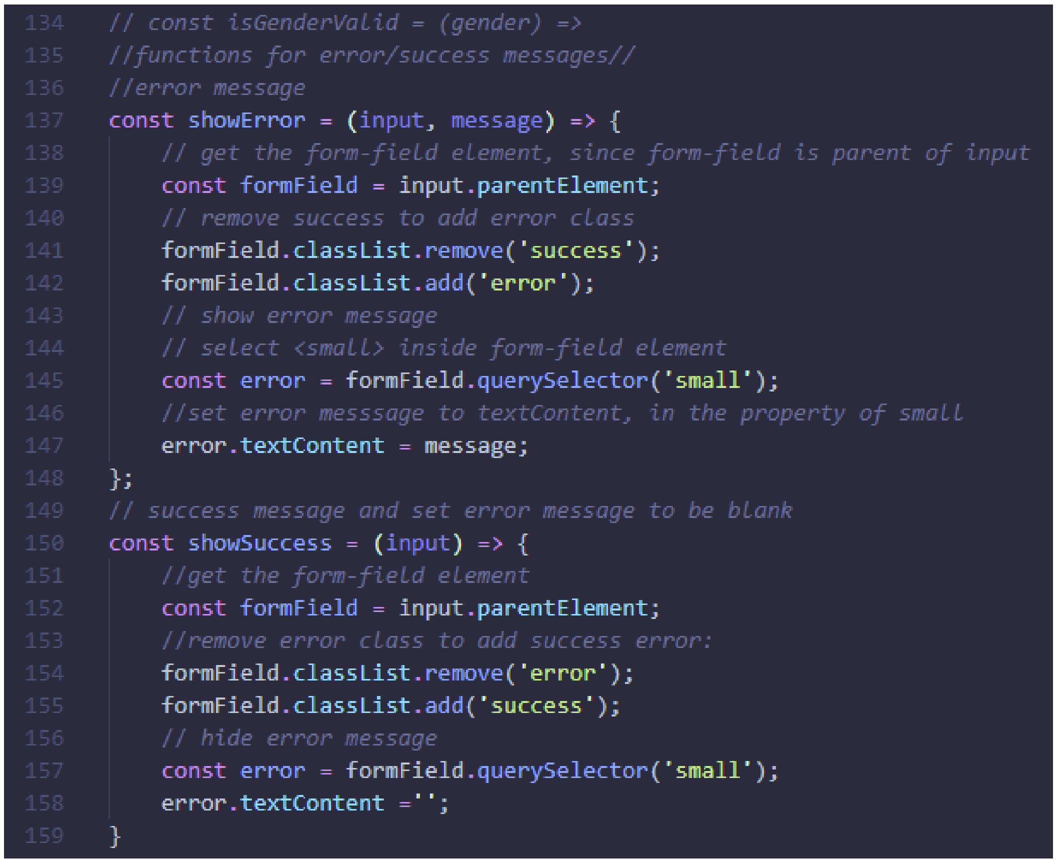
Coding elements:

Important reusable utility functions. These functions are used repeated, since a lot of input fields have crossovers between each one, eg. If null, return valid = false, rather than true. So instead of having to constantly rewrite between const, its best to create the same const to be reused, as important functions that won't be changed.



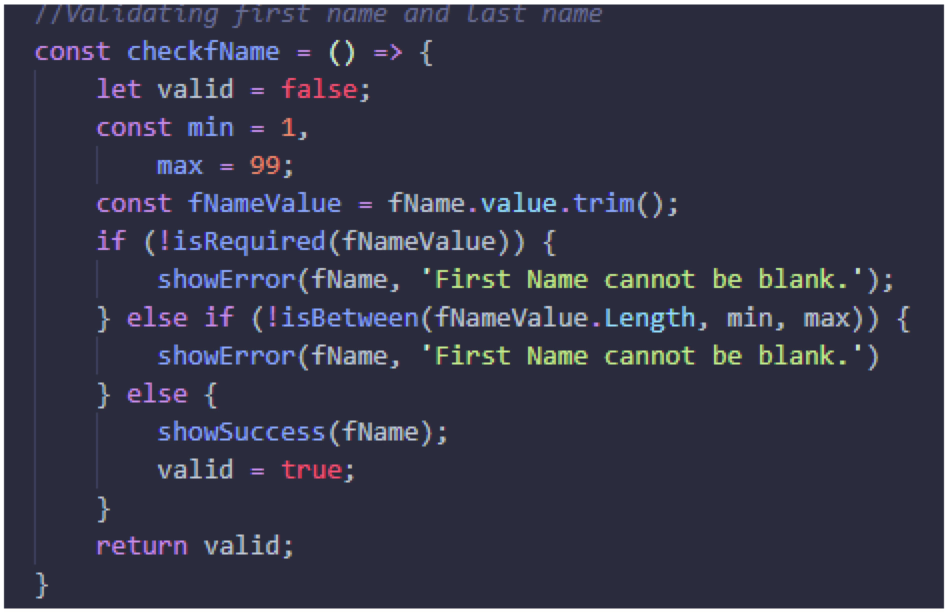
```
// reusable utility functions //
// function returns true if the inptu argument is empty
const isRequired = value => value === '' ? false : true;
//function returns false if lenght argument is not between the max and min
const isBetween = (length, min, max) => length < min || length > max ? false : true;
// function checks is email is valid
const isEmailValid = (email) => {
    const re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+
    return re.test(email);
}
//check is password is wrong, with specified pattern
    // ^ = password starts
    // (?=.*[a-z]) = password contains one lower case
    // (?=.*[A-Z]) = password contians one upper case
    // (?=.*[0-9]) = password contiains at least one number
    // (?=.*[!@#$%^&*]) = passsword contains a special character
    // (?=.{8,}) = must be eight chracters or longer
const isPasswordSecure = (password) => {
    const re = new RegExp("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#\$%\^&\*])(?=.{8,})");
    return re.test(password);
};
//check if date of birth is in correct format
const isDateValid = (dob) => {
    const re = /^(0?[1-9]|1\d|2\d|3[01])\/(0?[1-9]|1[0-2])\/((19|20)\d{2})$/
    return re.test(dob);
}
```

Error/Success messages. Similar to the above, error and success messages need to shown, where if valid = true, success message is shown, whilst if valid = false, error message will be shown.



```
// const isGenderValid = (gender) =>
//functions for error/success messages//
//error message
const showError = (input, message) => {
    // get the form-field element, since form-field is parent of input
    const formField = input.parentElement;
    // remove success to add error class
    formField.classList.remove('success');
    formField.classList.add('error');
    // show error message
    // select <small> inside form-field element
    const error = formField.querySelector('small');
    //set error messsage to textContent, in the property of small
    error.textContent = message;
};
// success message and set error message to be blank
const showSuccess = (input) => {
    //get the form-field element
    const formField = input.parentElement;
    //remove error class to add success error:
    formField.classList.remove('error');
    formField.classList.add('success');
    // hide error message
    const error = formField.querySelector('small');
    error.textContent ='';
}
```

Const layout. Below is an example of the layout of the code, I was following. I wanted to list out the conditions and outputs if valid = false, ie. when it does not follow through the required. This is because when valid = false, there is more requirements and outputs to be met, then when its true. Hence, why, else condition, is for when valid = true.



```
//Validating first name and last name
const checkfName = () => {
    let valid = false;
    const min = 1,
        max = 99;
    const fNameValue = fName.value.trim();
    if (!isRequired(fNameValue)) {
        showError(fName, 'First Name cannot be blank.');
    } else if (!isBetween(fNameValue.Length, min, max)) {
        showError(fName, 'First Name cannot be blank.')
    } else {
        showSuccess(fName);
        valid = true;
    }
    return valid;
}
```

Design motivation:

Wanted to use bold colours with clear geometry to shape around the websites. There were some aspects I could have applied better, both via code and on the page, but due to limited experience and time constraints, I had the most fun with what I have. The javascript files, go through a similar if and else statements, like some experience from MATLAB – which, honestly, I am very surprised it worked. There are some repeats in the style.css file, due to limited knowledge of what is going on and my experimenting of what works and what does not work. But by going with a simple bold design, it made my comfortable to getting used to how HTML, css and javascript work with each other.