

Bsc/HND Computing Portfolio

Building a Health-Based Website with JavaScript and jQuery

Web & Mobile Development, Assignment One.

ACCA5031

Dewi Morgan Griffiths

(GRI19 845 230)

Produced for Paul John of Gower College Swansea.

Due upon 30th of January in 2023.

Assignment is 50% of the module.

Outline of Document

Introduction of The Product.....	3	The Use of jQuery for User-Interface.....	14
Development Tools.....	3	Conclusion.....	15
Workflow Outline.....	4	Further Development.....	15
The Product.....	5	A Blog Using PHP and JavaScript.....	15
Demonstration.....	5	A Fully-Functioning Contact Form.....	15
JavaScript – Fitness Calculator Webpage		Improvements.....	16
(calc.js).....	5	A Complete Redesign without Bootstrap	
Defining User Input.....	5		16
Event Listeners and Functions.....	5	Altering Equations to Yield Further	
Alerts and Conditional Operators.....	8	Accurate Results.....	16
JavaScript – Contact Form Webpage		Fin.....	16
(contact.js).....	9	Appendix.....	18
Defining User Input.....	9	HTML Production.....	18
General Form Validation.....	9	Webpage Scripts.....	28
E-Mail Validation and Filtering Function		calc.js.....	28
.....	11	contact.js.....	38

Introduction of The Product

This website is directed toward individuals who wish to gain knowledge upon their health, they can browse our articles, use calculators to estimate how many calories are burned through activities. We write articles on people who have lost a lot of weight, overcome unhealthy habits, or have completed their milestone in sport on apps like Strava. [1] Other articles will relate to apps and services that can help more users understand the importance of health, mentally and physically.

Development Tools

For the website development workflow we are writing the project in Visual Studio Code or Vi Improved with the Windows Linux Subsystem for writing the entire website. The tmux package will be used to create multiple terminal instances (windows) for writing into multiple files at once. [2]The purpose of using a terminal-based IDE is that the development team prefer their tools being all in one place; and a command a way.

The project file and directory information is below.

Document File Type Name	Extension	Purpose	Location
Hypertext Preprocessor	.PHP	The purpose of creating a website in this file format as opposed to HTML, it reduces the need for editing every single file when the navigation bar changes or when links are removed. This reduces the amount of time wasted	Root directory, pages/, and includes/
Cascading Style Sheets	.CSS	The purpose of this is to style the webpage elements using the features provided that web browsers can interpret and style for the end-user.	CSS/
JavaScript	.JS	JavaScript will be used to create functions on the webpage, like calculators, animations and features to improve the user-experience.	JS/
Framework: Bootstrap and jQuery.	.JS	The purpose of using this framework as opposed to Bootstrap is that this framework supports use in old browsers, increasing the website's reach to more users.[3]	FW/JQ/

The web project will be tested on a web server known as the Windows, Apache, Mysql and PHP, or more commonly know as WAMP. This allows us to see how the website renders on a web server, which makes creating a better user experience straightforward as we can test it along the way during development. [4]

Workflow Outline

Before any web-based code is written, webpages that are created must have a purpose for their existence. This provides us with a goal and a reason to create a webpage in a specific way, and more importantly, what each webpage requires to function in the way we desire.

Webpage Directory	Webpage File	Webpage Additional Files	Webpage Purpose
/root	index.html	Every webpage will contain header.php or pheader.php, and footer.php	This webpage is the user's homepage, it will present them with information relating to the importance of health, useful apps and our calculators for estimating values that they wish to find out. Other information relate to articles written by us.
/includes/	header.php and pheader.php	headfoot.css style.css	This webpage is presented on every other website page, this includes navigation to the main pages. This renders a navigation bar for the user.
/includes/	footer.php and pfooter.php	features.js jquery-3.6.1.min.js	This webpage contains information about the website.
/pages/	calc.php	Yes. calc.js forms.css	This webpage allows the user to input values and calculate a result.
/pages/	contact.php	Yes. forms.css	This webpage contains a form to send the web developers a message.
/JS/	calc.js	None.	Contains JavaScript code to create a functioning calculator through DOM manipulation.
/JS/	contact.js	None.	Contains JavaScript code that validates and filters information the user sends to the mail server.
/FW/BS/css	bootstrap.min.css	None.	Bootstrap is used to make the webpage look pretty without writing too much CSS so the functionality can be focused on.
/FW/BS/js	bootstrap.min.js	None.	Bootstrap's script file that assists its css component in making the webpage look pretty.
/FW/JQ/	jquery-3.6.1.min.js	None.	The jQuery framework, used to create features on the webpage.
/CSS/	forms.css	None.	Used to edit the look of forms on webpages which contain them.
/CSS/	headfoot.css	None.	Contains information to style the navigation bar and footer of the document. This is used on every webpage.
/CSS/	style.css	None.	This adjusts the default styles shown on a webpage for a consistent user experience which looks nice, on every webpage.

Note: 'yes' refers to pre-established files that are present in every other page, repetition isn't necessary.

After the directories and their content are created, necessary files are related to each other within the appropriate documents (CSS files into header.php, Etc.).

All of the webpages will be styled with Bootstrap first after the bare-bones content is complete. The webpage functionality will be written through JavaScript and jQuery after the layout is complete. After everything functions properly, CSS will be applied.

The Product

The following content describes the product in a state of completion in regard to its scripts. See Further Development for more details referring to the product's future.

Demonstration

Images convey more than words will ever describe – a video-based demonstration will show off all of the features this website has to offer the end user. [5]

JavaScript – Fitness Calculator Webpage (calc.js)

JavaScript has been used for calculating results based on user input and filtering questionable inputs that are undesirable, especially when sending over to our email server.

Defining User Input

Every user input and element which is used to present data is defined in calc.js.

Due to the calculators being all available within one page when loaded, JavaScript and jQuery is required to hide and show elements based on user interaction. [See Appendix: calc.js]

Event Listeners and Functions

Functions that are placed into the event listeners in JavaScript normally return TRUE or FALSE.

```
function validateBodyFat() {  
    if (inpCalBodyFat.value == null | inpCalBodyFat.value == "" | inpCalBodyFat.value > 40 |  
inpCalBodyFat.value < 3){  
        return btnResCal.disabled = true;  
    }  
    if (inpCalBodyFat.value <= 40 && inpCalBodyFat.value >= 3) { // remember to place .value  
after the input, to avoid the common error!  
        return btnResCal.disabled = false;  
    }  
}
```

This is deliberate as when it comes to processing information, the input needs to be accepted and accurate. Ambiguous rules lead to bad validation and undesired information to process and store.

```
inpCalWeight.addEventListener('input',function() {  
    validateBodyFat();  
    validateWeight(btnResCal,inpCalWeight,metricCal);  
});
```

Every function related to validation returns TRUE or FALSE which is carried out through a final check to decide whether or not to throw the values in to be computed.

```
btnResFat.addEventListener('click',function() {
  let check1 = validateFatGender();
  let check2 = validateFatAge();
  // let check3 = validateWeight(btnResFat,inpFatWeight,metricFat); // double check the results for
comparison
  // Deprecated check3 due to not needing weight for the equation as BMI is needed for this
already.
  let check4 = validateFatBMI();
  if (check1 == true | check2 == true /*check3 == true*/ | check4 == true) { // If one of the
results are true (meaning wrong)
    resultFat.style.display = "none";
    attempt++;
    // Prompt a message specifying which fields need to be appropriated.
    alert(`Attempt ${attempt}#\nYou have invalid data!\nPlease observe:
    ${check1 == true ? "✗ Gender" : "✓ Gender"}
    ${check2 == true ? "✗ Age" : "✓ Age"}
    ${check4 == true ? "✗ BMI" : "✓ BMI"}`);
  }
  if (check1 == false && check2 == false && /*check3 == false*/ check4 == false) {
    let attempt = 0;
    let eGender = document.querySelector(`input[name="gender"]:checked`).value == 1 ? 1 :
0; // 1 male, 0 female.
    // ^^ equation gender value.
    let BFP = inpFatAge.value < 17 ? (1.2 * inpFatBMI.value) + (0.23 * inpFatAge.value) - (10.8
* eGender) - 0.54 : (1.51 * inpFatBMI.value) - (0.7 * inpFatAge.value) - (3.6 * eGender) - 0.54;
    // The equation changes from child to adult as there are different numbers based on age.
    // 1.2 * inpFatBMI.value + 0.23 * inpFatAge.value - 10.8 * eGender - 0.54
    fatBod.textContent = `Your body fat percentage is ${BFP.toFixed(2)}%`;
    // Add bodyfat% into calorie calculator...
    inpCalBodyFat.value = `${BFP.toFixed(2)}`;

    resultFat.style.display = "block";
  }
  //let inpFatGender = document.querySelector(`input[name="gender"]:checked`);
  //console.log(`You are ${inpFatGender.value}`);
});
```

Every function that uses parameters return values for calculating results from the user's input. This is common throughout all of the JavaScript-based code in this project as this method is effective and reduces many lines of code.

For example:

```
function convertHeight(num,decision) {
  const F2M = 0.3048;
  /*switch (decision){
    case 1:
      return parseFloat(num);
    case 2:
      return parseFloat(num * F2M);
    default:
      return "Error";
  }*/
  // For some reason the SWitch statement likes to return "NaN". :/
  if (decision == 1) {
    return parseFloat(num);
  }
  if (decision == 2) {
    return parseFloat(num * F2M);
  }
}
function convertWeight(num,decision) {
  const LBMass = 0.45359237;
  const StoneMass = 6.350293; // This is used to convert values multiple times throughout
after validation.
  // Convert Lbs to Kilograms.
  if (decision == 1) {
    metricBMIWeight.value = 1;
    metricCal.value = 1; // For changing this for the user, in case they forget to when their
    // metricFat.value = 1; Answer carries over from one another.
    return parseFloat(num * LBMass); // Switch statement due to only requiring one line of
code.
  }
  if (decision == 2) {
    metricBMIWeight.value = 2;
    metricCal.value = 2;
    // metricFat.value = 2;
    return parseFloat(num);
  }
  if (decision == 3) {
    metricBMIWeight.value = 3;
    metricCal.value = 3;
    // metricFat.value = 3;
    return parseFloat(num * StoneMass); // Converts Stone to KG.
  }
}
```

Alerts and Conditional Operators

When users click a button to calculate their results, if there is anything returns TRUE in its checks, an alert will show up and let the user know of their mistakes. This is very useful for somebody who may not understand their values are absurd or tried giving the script bad information on purpose.

Conditional operators are used to assume this information, this is shown throughout the scripts written in this project. Conditional operators are very necessary to reduce five to six lines of code that allow the project to be easier to read. Ternaries are nested into each other if there are three possibilities: NULL, FALSE and TRUE.

True is invalid input.

Null is either invalid or excusable.

False is valid input.

The decision behind FALSE being valid, is that it says there is no issue. The threat is false.

JavaScript – Contact Form Webpage (contact.js)

JavaScript has been used for calculating results based on user input and filtering questionable inputs that are undesirable, especially when sending over to our email server. [See Appendix: contact.js]

Defining User Input

Before the code on the webpage is developed, user input and their elements are taken into account along with where the data is being sent. It is important to define the inputs to process.

```
// Define page elements.  
let btnSend = document.getElementById('btn-send'); // The button.  
let inpEMail = document.getElementById('email'); // The email box.  
let inpName = document.getElementById('name'); // the user's name  
let inpText = document.getElementById('write2us'); // The text area.  
let inpPron = document.getElementById('select-pronouns'); // Pronouns selection.
```

The button and user inputs are defined with the JavaScript's 'let' to declare elements into one variable.

General Form Validation

The form validation carried out is very similar to how validation in calc.js was achieved.

Boolean is used to judge whether a user has given appropriate information. The user needs to provide their pronouns, their name, their email and their message which is under 1,500 characters and above 100. The character limit on their name is a generous 50 characters. There is no limit on the email as different email providers have different limits for what they are willing to accept as an address.

All of these checks are within functions that are ran as soon as the user presses a button, once the button is clicked they have to confirm if they want to send the message, unless they have issues with their data.

```
function validateText () {  
  if (inpText.value.length >= 1500) {  
    return true;  
  }  
  if (inpText.value.length < 100) {  
    return null; // null if there is no text worth sending.  
  }  
  else {  
    return false; // false if there is no issue.  
  }  
}  
function validateName () {  
  // get the length of the text's value and compare it against the inbetween values that are accepted.  
  if (inpName.value == null | inpName.value == "" | inpName.value.length > 50 |  
  inpName.value.length <= 2) {  
    return true;  
  }  
}
```

```

    if (inpName.value.length <= 50 && inpName.value.length > 2) {
        return false;
    }
    function validatePronouns() {
        if (inpPron.value == 0) {
            return true;           // Validation TRUE means it's incorrect.
        }
        if (inpPron.value <= 6) {
            return false;         // No issues so FALSE.
        }
    }
}

```

The providers that are accepted in this form are stored in an array which is used in E-Mail Validation and Filtering Function section. This is to get rid of the possibility of a temporary email service being used to flood our email server.

```

let emailProvider =
['gmail.com','outlook.com','aol.com','yahoo.com','icloud.com','protonmail.com','yandex.com','mail.c
om','hotmail.com'];

```

E-Mail Validation and Filtering Function

Contact forms on the internet are easy to give false information to and flood the email server with requests to store messages written by anyone who browses the website. We had been careful and considered many ways at which we can prevent a bombing to the address. [6]

Similarly to validation-related functions in calc.js, the same techniques are used in this file.

To prevent emails from temporary email accounts that anyone can use for free, we have limited the email providers that can be used to send us mail. The email providers chosen are based off popular clients recommended in articles. [7]

The function responsible for email validation is “function validateEmail(){...” This is used for when the user clicks the button to send their message.

The idea is to break down an e-mail into multiple variables that are either true, or false before filtering or email syntax validation.

```
let atSymbol = inpEMail.value.split('\u0040').pop() != inpEMail.value ? '\u0040' : false;
let atDot = inpEMail.value.split('\u002E').pop() != inpEMail.value ? '\u002E' : false;
// ^^^ The value must be different if input is given, but incorrectly.
// This prevents triggering success code with bad / undesired values.
// The last part of the email to solve is the extension after the dot.
let emailName = false;
let atExtension = false; // default values used to break down the email.
let atDomain = false; // Less confusion when validating the integrity of the email.
// If the atSymbol (@) is true.
```

As the comments describe, these variables are used to find and store provided information. The validation does not begin until the two email-specific characters are given. “@” and “.” or ‘\u0040’ and ‘\u002E’. Escape sequences are used in case of any Unicode-related errors not accepting user’s text input. A specific value is being compared to the value of the user’s input when broken down.

```
function validateEmail () {
  // https://www.utf8-chartable.de/
  // @ in unicode is equal to 0040, using the unicode escape sequence.

  // If a split can be made, it is true. If not, it is false.
  let atSymbol = inpEMail.value.split('\u0040').pop() != inpEMail.value ? '\u0040' : false;
  let atDot = inpEMail.value.split('\u002E').pop() != inpEMail.value ? '\u002E' : false;
  // ^^^ The value must be different if input is given, but incorrectly.
  // This prevents triggering success code with bad / undesired values.
```

After the two symbols are present, the first checks the function does is provide values to the previously defined variables. This is deliberate to avoid any errors from there being no input and nothing to compare from the start.

The Email element is checked for an address name, the provider and the extension. These are broken up using the pre-defined JavaScript functions, “toLowerCase()”, “split()”, “reverse()” and “pop()”. The methods allowed for individual pieces of the email to be compared. If anything is missing, the validation would return true and make the user provide a valid email.

```

emailName = inpEMail.value.toLowerCase().split("\u0040").reverse().pop();
atExtension = inpEMail.value.toLowerCase().split("\u002E").pop();
atDomain = inpEMail.value.toLowerCase().split("\u002E").reverse().pop().split("\u0040").pop();
// ^^^ Requires multiple splits to get values inbetween @ (\u0040) and . (\u002E)
// Get the email extension.
}
if (emailName == false | atExtension == false |
    atDomain == false | atSymbol == false | atDot == false) {
    return true;    // return true if not correct syntax or existence.
}

```

After all the information is shown to be present and not false, the next step is to filter the email provider to avoid a flood. This was done using a DO loop since we are comparing one value to an array of them.

```

do {
    --i;
    if (emailProvider[i] != atDomain+atDot+atExtension) {
        validMail = false; // If nothing found in loop, it is false.
    }
    else {
        validMail = true; // Once true, give the valid mail truth status.
    }
    // console.log(`Reading ${i}. ${atDomain+atDot+atExtension} isn't equal to $
    {emailProvider[i]}`);
}while(emailProvider[i] != (atDomain+atDot+atExtension) && i != 0)
    // Loop through the array until the domain and extension are equal.

```

If there is a matching provider to the information the user has shown, a variable called “validMail” is true. This is the one time TRUE is used positively in the code, as it describes the integrity of the email to be truthful.

If no provider was found, the user would be told that it isn’t supported through returning NULL.

```

if (!validMail) {
    return null; // Tell the user their email provider is not supported.
}

```

If all of the variables are false and validMail is not false, a FALSE is returned and the user has successfully passed the email check.

```

if (emailName != false && atDomain != false && atExtension != false && validMail != false)
{ // This solves the no text with symbols issue.
    //alert(`Success  ${emailName}${atSymbol}${atDomain}${atDot}${atExtension}`);
    return false;
}

```

This function was a lot more complicated than the other checks due to the syntax of an email account and the desire to prevent any attacks or issues that could occur and break the page on behalf of the user.

It may be unethical to limit the user to what email provider they can use to message us as the internet should be a place of free expression and communication. This decision was made out of respect for our equipment, we are not catering to the 1% of people who do not use a mainstream email provider.

The Use of jQuery for User-Interface

jQuery has been used in the calculator webpage (calc.php), specifically effects that provide a smoother experience than an instant cut to the content.

jQuery is written at the beginning of the document, but was the last to be implemented. This is due to needing everything to work before user-experience is taken into account.

```
$(document).ready(function(){
    $("#btnBMI").click(function(){
        $(conBMI).fadeIn(750);
        $(conCal).hide();    // Hide the other elements if they are present.
        $(conFat).hide();
    });
    $("#btnCal").click(function(){
        $(conBMI).hide();
        $(conCal).fadeIn(750);    // Show the selected element based on the button.
        $(conFat).hide();
    });
    $("#btnFat").click(function(){
        $(conBMI).hide();
        $(conCal).hide();
        $(conFat).fadeIn(750);
    });
});
```

The jQuery code is read and initialised last in the document due to the statement:

```
$(document).ready(function(){
```

This tells the JavaScript interpreter that it requires the entire webpage and file to be read before jQuery is ready to be used.

Built-in jQuery functions that were used:

```
.hide();
.fadeIn();
```

“.hide();” tells the interpreter through jQuery’s framework that the content needs to be hidden. The content is selected through raw JavaScript and stored in a variable below jQuery.

“.fadeIn(x);” tells the interpreter to fade in the content, milliseconds are used to measure the time for the effect to take place.

Originally the content was to fade in and out, it felt and looked better for content to hide as soon as the next button is pressed, giving the illusion that the selected fades over it.

Conclusion

The project which has been designed and given functionality do have plans for the future which may change how the site looks or functions.

Further Development

This website has had many changes from its original idea to how it works today, some of those ideas are on halt for user-feedback and further testing. These are the ideas which will one day be implemented.

A Blog Using PHP and JavaScript

We plan to provide content to whomever visits our website giving the latest trends on health, nutrition, workout plans, intermittent fasting and scientific studies in regard to weight loss.

The site would use PHP and MySQL to store and retrieve articles from the database so that the user can browse a lot more than what they can now.

A Fully-Functioning Contact Form

The users as of now cannot submit an actual message to our mail server since we are still developing the website.

The idea is to send the message to our database which will forward it to another email provider that can interpret the message which the user wants read.

This will build a connection between the site developers and their daily users.

Improvements

This website is far from polished, finished or accurate. Major improvements can be made.

A Complete Redesign without Bootstrap

The website uses Bootstrap temporarily to present users with content that is easy to read and navigate regardless of their screen size. Raw CSS will be used along with jQueryUI to give the user a different browsing experience when compared to other health calculator sites.

Altering Equations to Yield Further Accurate Results

Our equations are very rough, rough to the point that they are considered possibly inaccurate depending on who you ask. This is due to the outdated material studied to work out the user's health information. [8],[9]

Fin.

The website that has been produced; is useful for those who seek information about their weight / health without the need of visiting a doctor. The scripts used on the webpages are well-thought out and contain many solutions to issues that were found in testing.

Many of the solutions can be made for re-use to avoid copy and pasting in many areas of the code. Some if statements are not required because they can be replaced with conditional operators in one line instead.

The website will be publicly available after its successful and solid development cycle is to a higher standard.

Bibliography

1: Strava, Inc., Strava | Run and Cycling Tracking on the Social Network for Athletes, 2009, <https://www.strava.com/>
2: Microsoft, What is the Windows Subsystem for Linux?, N/A, <https://learn.microsoft.com/en-us/windows/wsl/about>
3: OpenJS Foundation, Browser Support | jQuery UI, , <https://jqueryui.com/browser-support/>
4: Romain Bourdon, WampServer, la plateforme de développement Web sous Windows - Apache, MySQL, PHP, N/A, <https://www.wampserver.com/en/>

5: gri19845230, Demonstration of JavaScript., 2022-01-30, <https://ufile.io/wz5h7nrw>
6: Josh Hendrickson, How Email Bombing Uses Spam to Hide an Attack, 2020-09-18
7: Vaneezae Malli, 14 BEST Email Service & Account Providers in 2023 [Free + Paid], 2022-12-26, <https://www.mailmunch.com/blog/best-email-service-providers>
8: , BMR Calculator, N/A, <https://www.bmi-calculator.net/bmr-calculator/>
9: Moose & Doc, Body Fat Percentage: Find out yours, N/A, <https://halls.md/body-fat-percentage-formula/>

Appendix

HTML Production

This section provides more information for Workflow Outline.

After I had established where each file will be located and which ones would be accessing each other, I had to start the production of the website. Visualising where information would be once styled, what input boxes are needed, the equations, Etc.

Since I plan to use JavaScript and PHP to produce this website there isn't much of an issue with creating consistency across each page, this gives me time to focus more on the content.



- [Homepage](#)
- [Articles](#)
- [Calculators](#)
- [Contact](#)

Welcome to Fitness

Say goodbye to fat!

Yes, you have heard it here first! Fitness is awesome and you shouldn't take your bodies foregranted; like some many of us have already! If you are overwe

We'll tell you straight, being fat is no laughing matter. Getting in shape doesn't mean you will look good, it is you working for a healthier body and a more cl
will no longer be a chore.

I know, that doesn't sound like the average you! But if you are serious about become healthier, open your mind to the idea you are going to change your life

What is Healthy?

Being skinny and fit?

Many of you probably think of unhealthy as being chubby, obese, too skinny or bald! In reality when medical professionals refer to a healthy body, they look

Getting into these greathabits will not only improve your physical health, your mental health will change. A happy body is a happy mind, do not forget that

Salad Vs. Vegan

There's a difference?

Believe it or not, vegan products are not that good for you. Eating healthy isn't just giving up natural foods, it is giving up unhealthy foods.

Just because the packaging is green, it doesn't make it healthy, packaging that say "Only 104 calories per [enter value]", they are compensating for how harr

To eat healthy you need a variety in your diet.. we don't mean a variety of foods that are greasy, fried or smoked. We mean a variety of nutrients that each p

A healthy diet consists of protein, fruits, vegetables and of course salad. Vegan products are not made of that, they contain harmful ingredients.

Where Can I Find Good Food?

Is it safe?

Believe it or not, there is not much options for what you could eat when compared to unhealthy foods, if you are attempting to only eat healthy.

Regardless of this fact, this makes it easier for finding healthy foods as you know what you are looking for. A lot of grocery stores have a produce section.

But if you have the option, go to your local farmer's market and pick out locally grown produce, before you eat remember to wash them.

Figure 1: index.php before any styling is applied.

My website features many <div> tags within each other, to assist myself in visualising where content would be, and which elements should be alongside each other.

Source Code of index.php

```
<?php include "includes/connect.php";
include "includes/header.php";

?>

<div>
    <div>
        <h1>Welcome to Fitness</h1>
        <h2 class="sub">Say goodbye to fat!</h2>
        <div>
            <p>Yes, you have heard it here first! Fitness is awesome and you shouldn't
take your bodies foregranted; like some many of us have already! If you are overweight, do not
be ashamed because you are in the right place.</p>
            <p>We'll tell you straight, being fat is no laughing matter. Getting in
shape doesn't mean you will look good, it is you working for a healthier body and a more clear
mind. When you get out of the habits which eat up your lifespan, you will feel more connected
to yourself and people around you. You learn to enjoy nature and going outside will no longer
be a chore.</p>
            <p>I know, that doesn't sound like the average you! But if you are serious
about become healthier, open your mind to the idea you are going to change your life through
habits you pick up and turning your current ones into dead habits, null and void.</p>
        </div>
    </div>
    <div>
        <h1>What is Healthy?</h1>
        <h2 class="sub">Being skinny and fit?</h2>
        <div>
            <p>Many of you probably think of unhealthy as being chubby, obese, too
skinny or bald! In reality when medical professionals refer to a healthy body, they look at
how your organs function, how your blood flows through your body, and more importantly, how it
affects you on a daily basis.</p>
            <p>Getting into these <b>great</b> habits will not only improve your
physical health, your mental health will change. A happy body is a happy mind, do not forget
that when you next go and order fried foods and a side of extra salt and cheese. A balanced
diet is required for this transformation journey.</p>
        </div>
    </div>
    <div>
        <h1>Salad Vs. Vegan</h1>
        <h2>There's a difference?</h2>
        <div>
            <div>
                <p>Believe it or not, vegan products are not that good for you.
Eating healthy isn't just giving up natural foods, it is giving up unhealthy foods.</p>
                <p>Just because the packaging is green, it doesn't make it healthy,
packaging that say "Only 104 calories per [enter value]", they are compensating for how
harmful their product is.</p>
            </div>
            <div>
                <p>To eat healthy you need a variety in your diet.. we don't mean a
variety of foods that are greasy, fried or smoked. We mean a variety of nutrients that each
provides you.</p>
                <p>A healthy diet consists of protein, fruits, vegetables and of
course salad. Vegan products are not made of that, they contain harmful ingredients.</p>
            </div>
        </div>
    </div>
    <div>
        <h1>Where Can I Find Good Food?</h1>
```

Source Code of index.php

```
<h2>Is it safe?</h2>
<div>
    <p>Believe it or not, there is not much options for what you could eat
when compared to unhealthy foods, if you are attempting to only eat healthy.</p>
    <p>Regardless of this fact, this makes it easier for finding healthy foods
as you know what you are looking for. A lot of grocery stores have a produce section.</p>
    <p>But if you have the option, go to your local farmer's market and pick
out locally grown produce, before you eat remember to wash them.</p>
</div>
</div>
<div>
    <h1>Explore Your Health and Fitness</h1>
    <h2>Bookmark this website!</h2>
    <div>
        <p>We write articles for free, raising awareness of health and the benefits of
many other habits that we haven't mentioned.</p>
        <p>Our goal is to empower people to explore themselves and to overcome their life
threatening minor decisions.</p>
        <p>Share us with your friends and family, do not let them suffer alone.</p>
    </div>
</div>
</body>
<footer><?php include "includes/footer.php";?></footer>
```

As you can see I have included the end of the HTML document within the <footer> tag as there is an absence of the closing page in this file. Doing this reduces the many times I would need to write it and wastes less time spent writing repetitive HTML.

Figure 2: calc.php before styling.



- [Homepage](#)
- [Articles](#)
- [Calculators](#)
- [Contact](#)

Calculate Your Health!

Use our calculators to **estimate** your physical health.

Body Mass Index

Check how healthy your BMI truly is.

Enter Height	⬆ ⬇ ⬆	Milimetres (mm)	▼
Enter Weight	⬆ ⬇ ⬆	Pounds (Lbs)	▼
Calculate			

Your Results!

Your BMI is calculated to be:

Enter text about being healthy, too unhealthy or just right.

What is a Body Mass Index?

The Body Mass Index (BMI) is the measurement of your body's fatness. A high BMI doesn't mean you are unhealthy, but it can bring on concern. If you are worried of your BMI being high or too low, seek medical advice and get your body tested for further investigation as to whether or not you are healthy. Just because your BMI is within the recommended range for your height, this does not mean you are overall healthy, if you feel there may be something wrong, consult a doctor.

Calorie Burner Calculator

Find out how many calories you'd burn from various activities!

Enter Body Fat%	⬆ ⬇ ⬆	
Enter Weight	⬆ ⬇ ⬆	Pounds (Lbs) ▼
Calculate		

Your Results!

Your BMI is calculated to be:

Enter text about being healthy, too unhealthy or just right.

Don't Know Your Body Fat Percentage?

Your body fat percentage requires you to get your weight in your preferred measurement, you will also need to provide your age and gender.

Source Code of calc.php

```
<?php include "../includes/connect.php";
include "../includes/pheader.php";

?>

<div>
    <h1>Calculate Your Health!</h1>
    <p>Use our calculators to <b>estimate</b> your physical health.</p>
</div>
<div>
    <div>
        <h2>Body Mass Index</h2>
        <p>Check how healthy your BMI truly is.</p>
        <form>
            <input type="number" name="user-height" placeholder="Enter Height">
            <select id="height">
                <option name="user-tall" value="1">Milimetres (mm)</option>
                <option name="user-tall" value="2">Feet/Foot (ft)</option>
            </select>
            <br>
            <input type="number" name="user-weight" placeholder="Enter Weight">
            <select id="weight">
                <option name="user-metric" value="1">Pounds (Lbs)</option>
                <option name="user-metric" value="2">Kilograms (Kgs)</option>
                <option name="user-metric" value="3">Stone</option>
            </select>
            <br>
            <button type="submit" name="submit" value="Enter
JavaScript">Calculate</button>
        </form>
    </div>
    <div>
        <h3>Your Results!</h3>
        <p>Your BMI is calculated to be: </p>
        <p>Enter text about being healthy, too unhealthy or just right.</p>
    </div>
</div>
<div>
    <h1>What is a Body Mass Index?</h1>
    <p>The Body Mass Index (BMI) is the measurement of your body's fatness. A high BMI
doesn't mean you are unhealthy, but it can bring on concerning health issues which relate to
your BMI being high.</p>
    <p>If you are worried of your BMI being high or too low, seek medical advice and
get your body tested for further investigation as to whether or not you are in danger of
obtaining health issues from as you are now.</p>
    <p>Just because your BMI is within the recommended range for your height, this does
not mean you are overall healthy, if you feel there may be something wrong, do seek medical
advice.</p>
</div>
</div>
<div>
    <div>
        <h2>Calorie Burner Calculator</h2>
        <p>Find out how many calories you'd burn from various activities!</p>
        <!--Use the Katch & McArdle Method      Basal Metabolic Rate = 370 + (21.6 * (Lean
Mass = Weight - (weight * body fat%))-->
        <!-- Activity Multiplier is 1.[x] * BMR-->
        <form>
            <input type="number" name="user-bodyfat" placeholder="Enter Body Fat%">
            <br>
```

Source Code of calc.php

```
<input type="number" name="user-weight" placeholder="Enter Weight">
<select id="weight">
    <option name="user-metric" value="1">Pounds (Lbs)</option>
    <option name="user-metric" value="2">Kilograms (Kgs)</option>
    <option name="user-metric" value="3">Stone</option>
</select>
<br>
<button type="submit" name="submit" value="Enter
JavaScript">Calculate</button>
</form>
<div>
    <h3>Your Results!</h3>
    <p>Your BMI is calculated to be: </p>
    <p>Enter text about being healthy, too unhealthy or just right.</p>
</div>
</div>
<div>
    <h3>Don't Know Your Body Fat Percentage?</h3>
    <p>Your body fat percentage requires you to get your weight in your preferred
measurement, you will also need to provide your age and gender.</p>
    <p>You can get your body fat percentage with our <a href="#">body fat
calculator</a>.</p>
</div>
</div>
<div>
    <!--Equation used for the BFP= 1.2 * BMI + 0.23 * Age - 10.8 * Gender - 0.54-->
    <h2>Body Fat% Calculator</h2>
    <p>Discover your body fat percentage to find out how many calories you burn!</p>
    <form>
        <input type="radio">Male</input>
        <input type="radio">Female</input>
        <br>
        <input type="number" placeholder="Enter age" name ="user-age" min="1" max="99">
        <div>
            <input type="number" name="user-weight" placeholder="Enter Weight">
            <select id="weight">
                <option name="user-metric" value="1">Pounds (Lbs)</option>
                <option name="user-metric" value="2">Kilograms (Kgs)</option>
                <option name="user-metric" value="3">Stone</option>
            </select>
        </div>
        <div>
            <input type="number" placeholder="Enter BMI" name="user-BMI" min="15"
max="40">
        </div>
        <button type="submit" name="submit" value="Enter JavaScript">Calculate</button>
    </form>
</div>
<?php include "../includes/pfooter.php"; // for pages.??>
```

I had to place the equations for the calculations in the calc.php HTML as comments, for me to refer to later when creating a script that will carry out those calculations.

The idea is to have the calculator to the left, to the right of it will be information about what the significance of the calculator is, and why the users should refer to it for an estimate of their physical health.

Each calculator will be separated through pages that are already loaded, this will reduce the need for many other pages and all of the information will be available in one place through the click of a button.



- [Homepage](#)
- [Articles](#)
- [Calculators](#)
- [Contact](#)

Write to Us!

We don't bite!

Feel free to let us know what you think of our website, calculators, current contact and what we

If you don't have any feedback and would like to let us know of your journey, feel free to contact

Enter E-Mail

Enter Name

He/Him

Write to us...

Browse... No file selected.

Send

Webpage completely developed by Dawi.

Frameworks used are jQuery and jQueryUI.

Any enquiries, e-mail me at GRI19845230@stu.gcs.ac.uk

Figure 3: contact.php before styling.

The contact.php webpage features multiple inputs for the user, we require the e-mail for the web server to send us their information so they can be contacted. The file is optional to send over, this is just in case people decide they want to send in photographs, videos or a collection of them for us to see and possibly write articles on.

Source Code for contact.php

```
<?php include "../includes/connect.php";
include "../includes/pheader.php";

?>

<div>
    <h1>Write to Us!</h1>
    <h2>We don't bite!</h2>
    <div>
        <p>Feel free to let us know what you think of our website, calculators,
current contact and what we could change for more accuracy, visitors or anything
more!</p>
        <p>If you don't have any feedback and would like to let us know of your
journey, feel free to contact. We love talking to people on their journey, and posting
about their accomplishments if given permission.</p>
    </div>
</div>
<div>
    <form>
        <br>
        <input type="email" value="" name="email" autofocus placeholder="Enter E-
Mail" required></input>
        <br>
        <input type="text" value="" name="users-name" placeholder="Enter Name"
required></input>
        <select>
            <option name="user-pronouns" value="1">He/Him</option>
            <option name="user-pronouns" value="2">She/Him</option>
            <option name="user-pronouns" value="3">He/Her</option>
            <option name="user-pronouns" value="4">She/Her</option>
            <option name="user-pronouns" value="5">They/Them</option>
        </select>
        <br>
        <textarea placeholder="Write to us..." rows="5" cols="60"
required></textarea>
        <br>
        <input type="file" id="file" name="file">
        <input type="button" value="Send" name="submit"></input>
    </form>
</div>

<?php include "../includes/pfooter.php"; // for pages.
?>
```

The required information is the e-mail, username and of course the text content. We decided to give them pronouns to choose from as we are attempting to attract a large user-base, including younger individuals who have an interest in improving their health.

The following is the code for the included files so far in every webpage.

Source code for header.php

```
<?php include "includes/connect.php";

?>
<head>
    <link rel="stylesheet" type="text/css" href="CSS/headfoot.css">
    <link rel="stylesheet" type="text/css" href="CSS/style.css">
    <link rel="stylesheet" type="text/css" href="FW/JQUI/jquery-ui.min.css">
</head>
<body>

<div>
    <nav>
        <ul>
            <a href="index.php"><li>Homepage</li></a>
            <a href="pages/feed.php?from=home"><li>Articles</li></a>
            <a href="pages/calc.php?from=home"><li>Calculators</li></a>
            <a href="pages/contact.php?from=home"><li>Contact</li></a>
        </ul>
    </nav>
</div>
```

Source code for pheader.php

```
<?php include "../includes/connect.php";

?>
<head>
    <link rel="stylesheet" type="text/css" href="../CSS/headfoot.css">
    <link rel="stylesheet" type="text/css" href="../CSS/style.css">
    <link rel="stylesheet" type="text/css" href="../FW/JQUI/jquery-ui.min.css">
</head>
<body>

<div>
    <nav>
        <ul>
            <a href="../index.php"><li>Homepage</li></a>
            <a href="../pages/feed.php"><li>Articles</li></a>
            <a href="../pages/calc.php"><li>Calculators</li></a>
            <a href="../pages/contact.php"><li>Contact</li></a>
        </ul>
    </nav>
</div>
```

Source code for footer.php

```
<?php
    // Insert any information about the creation of these webpages.
?>
    <p>Webpage completely developed by Dawi.</p>
    <p>Frameworks used are jQuery and jQueryUI.</p>
    <p>Any enquiries, e-mail me at GRI19845230@stu.gcs.ac.uk</p>
    <!--Scripts used for the content in every page.-->
    <script type="text/javascript" src="FW/JQUI/jquery-ui.min.js"></script>
    <script type="text/javascript" src="FW/JQ/jquery-3.6.1.min.js"></script>
    <script type="text/javascript" src="JS/style.js"></script>
</html>
```

Source code for footer.php

```
<?php
    // Insert any information about the creation of these webpages.
?>
    <p>Webpage completely developed by Dawi.</p>
    <p>Frameworks used are jQuery and jQueryUI.</p>
    <p>Any enquiries, e-mail me at GRI19845230@stu.gcs.ac.uk</p>
    <!--Scripts used for the content in every page.-->
    <script type="text/javascript" src="../FW/JQUI/jquery-ui.min.js"></script>
    <script type="text/javascript" src="../FW/JQ/jquery-3.6.1.min.js"></script>
    <script type="text/javascript" src="../JS/style.js"></script>
</html>
```

The reason for there being two header and footer files is because they are for pages after the homepage (index.php).

The reason for the inclusion of a connect.php file, that is there for when the back-end of the project is being carried out. This refers to presenting posts in feed.php and directing the user toward the page with the information, the contact page will function by storing the details of the user for the admins to later respond to.

Webpage Scripts

This covers the production of the client-side scripts.

calc.js

The following is all of the code and comments for this script featured in the calc.php webpage.

```
// Initilise when webpoage is loaded.
$(document).ready(function(){
    $("#btnBMI").click(function(){
        $(conBMI).fadeIn(750);
        $(conCal).hide();    // Hide the other elements if they are present.
        $(conFat).hide();
    });
    $("#btnCal").click(function(){
        $(conBMI).hide();
        $(conCal).fadeIn(750);    // Show the selected element based on the button.
        $(conFat).hide();
    });
    $("#btnFat").click(function(){
        $(conBMI).hide();
        $(conCal).hide();
        $(conFat).fadeIn(750);
    });

    /*if (button == 1) {
        $(conBMI).fadeIn(1500);
        $(conCal).fadeOut(1500);
        $(conFat).fadeOut(1500);
    }
    if (button == 2) {
        $(conBMI).fadeOut(1500);
        $(conCal).fadeIn(1500);
        $(conFat).fadeOut(1500);
    }
    if (button == 3) {
        $(conBMI).fadeOut(1500);
        $(conCal).fadeOut(1500);
        $(conFat).fadeIn(1500);
    }
    */
});
// Script for the functioning of calc.php

// Default appearance of the elements for when the user enters the website.
let conBMI = document.getElementById("bmi");
let conCal = document.getElementById("calories");
let conFat = document.getElementById("bodyfat");
conCal.style.display = "none";
conFat.style.display = "none";
```

```

// Get the buttons which will trigger the elements.
// This is the UI part of the code.
let btnBMI = document.querySelector("#btnBMI");
let btnCal = document.querySelector("#btnCal");
let btnFat = document.querySelector("#btnFat");
let decision = 1;

btnBMI.addEventListener('click', function() {
    /*conBMI.style.display = "flex";
    conCal.style.display = "none";
    conFat.style.display = "none";*/
    validateHeight(btnResBMI,inpBMIHeight,metricBMIHeight); // validate Height and Weight.
    validateWeight(btnResBMI,inpBMIWeight,metricBMIWeight);
});
btnCal.addEventListener('click', function() {
    /* conBMI.style.display = "none";
    conCal.style.display = "flex";
    conFat.style.display = "none"; */
    validateBodyFat();
    validateWeight(btnResCal,inpCalWeight,metricCal);
});
btnFat.addEventListener('click', function() {
    /* conBMI.style.display = "none";
    conCal.style.display = "none";
    conFat.style.display = "flex"; */
    validateFatGender();
    validateFatAge();
    validateWeight(btnResFat,inpFatWeight,metricFat);
    validateFatBMI();
});

// This is the text part of the code.
let resultBMI = document.getElementById("#resBMI");
let resultCal = document.querySelector("#resCal");
let resultFat = document.querySelector("#resFat");
resultBMI.style.display = "none";
resultCal.style.display = "none";
resultFat.style.display = "none";

// Show the result when the button is pressed.
let btnResBMI = document.querySelector("#btnResBMI");
let btnResCal = document.querySelector("#btnResCal"); // Name of the result buttons.
let btnResFat = document.querySelector("#btnResFat");

// Define all controls where values are taken from.
//inpBMIHeight = document.getElementsByName("user-height");
//inpBMIWeight = document.getElementsByName("user-height");

```

```

    // BMI text inputs.
    let inpBMIHeight = document.getElementById("bmi-user-height");
    let inpBMIWeight = document.getElementById("bmi-user-weight"); // The name of text inputs.
    let metricBMIHeight = document.querySelector("#bmi-height"); // The drop down.
    let metricBMIWeight = document.querySelector("#bmi-weight");

    // Calorie text inputs.
    let inpCalBodyFat = document.getElementById("cal-bodyfat");
    let inpCalWeight = document.getElementById("cal-weight");
    let metricCal = document.getElementById("cal-metric"); // The drop down

    // Fat text inputs.
    let inpFatAge = document.getElementById("fat-age"); // Name of the text inputs.
    let inpFatWeight = document.getElementById("fat-weight");
    let inpFatBMI = document.getElementById("fat-bmi");
    let metricFat = document.getElementById("fat-metric"); // The Drop Down.

    // For placing the answers into the webpage.
    let bmiAnswer = document.getElementById("bmi-answer");
    // Calories Calculator answers.
    let calFatAnswer = document.getElementById("cal-fat"); // The results shown.
    let calLeanAnswer = document.getElementById("cal-lean"); // The results shown.
    let calSultAnswer = document.getElementById("cal-res"); // The results shown.
    // Body Fat% Calculator has multiple answers to be shown at once.
    let fatMR = document.getElementById("fat-met");
    let fatBF = document.getElementById("fat-bf");
    let fatBod = document.getElementById("fat-bod");

    // Reset all inputs upon rendering of the page.
    inpBMIHeight.value = null;
    inpBMIWeight.value = null;
    inpCalBodyFat.value = null;
    inpCalWeight.value = null; // The value is null when refreshed ( the page ).
    inpFatAge.value = null;
    inpFatWeight.value = null;
    inpFatBMI.value = null;
    // Doesn't matter if the gender radio buttons are checked, user can adjust easily.

    // Enter all of the validation for the calculator here.
    // Reusable Solutions for validation & conversion.

    function validateWeight(thebutton, theweight, metric) { // Refactored for the other calculator tabs
        // The drop down will determine the validation.
        // The format is, if less 20lbs or greater than 500lbs, button is disabled.
        if (theweight.value == null | metric.value == 1 && theweight.value > 500 | theweight.value <
30) {
            return thebutton.disabled = true;

```

```

    }
    if (theweight.value == null | metric.value == 2 && theweight.value > 227 | theweight.value <
9) {
        return thebutton.disabled = true;
    }
    if ( theweight.value == null | metric.value == 3 && theweight.value > 35 | theweight.value <
2) {
        return thebutton.disabled = true;
    }
    if (metric.value == 1 && theweight.value < 500 && theweight.value > 20) {
        return thebutton.disabled = false;
    }
    if (metric.value == 2 && theweight.value < 227 && theweight.value > 9 ) {
        return thebutton.disabled = false;
    }
    if (metric.value == 3 && theweight.value < 35 && theweight.value > 2) {
        return thebutton.disabled = false;
    }
}

// Reusable code for height in case of future uses, similar to validateWeight
function validateHeight(thebutton, theHeight, metric) { // Validates height, the same as weight.
    // The drop down will determine the validation.
    // The format is, if less 20lbs or greater than 500lbs, button is disabled.
    //console.log(thebutton);console.log(theHeight.value);console.log(metric.value);
    if (theHeight.value == null | metric.value == 1 && theHeight.value > 2.41 | theHeight.value
< 0.6) {
        return thebutton.disabled = true; // Be careful with copy and pasting, spent 1.5 hours fixing
an issue, which was due to a stay theight.value between | and |.
    } // Using metres and feet to measure height.
    if (theHeight.value == null | metric.value == 2 && theHeight.value > 8 | theHeight.value < 2)
{
        return thebutton.disabled = true;
    }
    if (metric.value == 1 && theHeight.value < 2.41 && theHeight.value > 0.6) {
        return thebutton.disabled = false;
    }
    if (metric.value == 2 && theHeight.value < 8 && theHeight.value > 2 ) { // If between these
two values.
        return thebutton.disabled = false;
    }
}

// This section of validation is the conversion for Lbs and Stone to Kgs and Feet to Metres.
function convertWeight(num,decision) {
    const LBMass = 0.45359237;
    const StoneMass = 6.350293; // This is used to convert values multiple times throughout
after validation.
    // Convert Lbs to Kilograms.
    if (decision == 1) {
        metricBMIWeight.value = 1;

```

```

        metricCal.value = 1;    // For changing this for the user, in case they forget to when their
// metricFat.value = 1;    Answer carries over from one another.
        return parseFloat(num * LBMass); // Switch statement due to only requiring one line of
code.
    }
    if (decision == 2) {
        metricBMIWeight.value = 2;
        metricCal.value = 2;
        // metricFat.value = 2;
        return parseFloat(num);
    }
    if (decision == 3) {
        metricBMIWeight.value = 3;
        metricCal.value = 3;
        // metricFat.value = 3;
        return parseFloat(num * StoneMass); // Converts Stone to KG.
    }
    /* switch (decision){
        case 1:
            return parseFloat(num * LBMass); // Switch statement due to only requiring one line of
code.
        case 2:
            return parseFloat(num);
        case 3:
            return parseFloat(num * StoneMass); // Converts Stone to KG.
        default:
            return "Error";
    }*/
} // Do conversions on the 25th when you can.

function convertHeight(num,decision) {
    const F2M = 0.3048;
    /*switch (decision){
        case 1:
            return parseFloat(num);
        case 2:
            return parseFloat(num * F2M);
        default:
            return "Error";
    }*/
    // For some reason the SWitch statement likes to return "NaN". ./
    if (decision == 1) {
        return parseFloat(num);
    }
    if (decision == 2) {
        return parseFloat(num * F2M);
    }
}
}

```

```

// BMI Calculator Validation
inpBMIWeight.addEventListener('input',function() { // When focusing in on the last input, check.
    validateHeight(btnResBMI,inpBMIHeight,metricBMIHeight); // validate Height and Weight.
    validateWeight(btnResBMI,inpBMIWeight,metricBMIWeight);
});
btnResBMI.addEventListener('click',function(){
    attempt++;
    let check1 = validateHeight(btnResBMI,inpBMIHeight,metricBMIHeight); // validate Height
and Weight.
    let check2 = validateWeight(btnResBMI,inpBMIWeight,metricBMIWeight);

    if (check1 == true | check2 == true){
        resultCal.style.display = "none";
        alert(`Attempt ${attempt}#\nYou have invalid data!\nPlease observe:
        ${check1 == true ? "✗ Height" : "✓ Height"}
        ${check2 == true ? "✗ Weight" : "✓ Weight"}`);
    }
    if (check1 == false && check2 == false) {
        attempt = 0; // Fix this when you can, it gives inaccurate result.
        let BMI = convertWeight(inpBMIWeight.value,metricBMIWeight.value) /
(convertHeight(inpBMIHeight.value,metricBMIHeight.value)*convertHeight(inpBMIHeight.value,
metricBMIHeight.value));
        bmiAnswer.textContent = `Your BMI is calculated to be: ${BMI.toFixed(2)}`;
        // Give the weight to the calorie calculator...
        inpCalWeight.value = inpBMIWeight.value;
        //inpCalWeight.value = `${inpBMIWeight.value}`;
        // Put the BMI as a value for bodyfat%...
        inpFatBMI.value = `${BMI.toFixed(2)}`;
        resultBMI.style.display = "block";
    }
});

// Calories Calculator Validation
function validateBodyFat() {
    if (inpCalBodyFat.value == null | inpCalBodyFat.value == "" | inpCalBodyFat.value > 40 |
inpCalBodyFat.value < 3){
        return btnResCal.disabled = true;
    }
    if (inpCalBodyFat.value <= 40 && inpCalBodyFat.value >= 3) { // remember to place .value
after the input, to avoid the common error!
        return btnResCal.disabled = false;
    }
}

// Links
let linkBFP = document.getElementById(`bf-link`);
linkBFP.addEventListener('click',function(){conCal.style.display = "none";conFat.style.display =
"flex";});

```

```

// This is in case of the fields already containing values from other inputs.
btnBMI.addEventListener('onhover',function() {
    validateBodyFat();
    validateWeight(btnResCal,inpCalWeight,metricCal);
});
inpCalWeight.addEventListener('input',function() {
    validateBodyFat();
    validateWeight(btnResCal,inpCalWeight,metricCal);
});
// The submit button for getting the answers.
btnResCal.addEventListener('click', function() {
    attempt++;
    let check1 = validateBodyFat();
    let check2 = validateWeight(btnResCal,inpCalWeight,metricCal);
    // console.log(`Option 1# - ${check1}\nOption 2# - ${check2}`);
    if (check1 == true | check2 == true) {
        alert(`Attempt ${attempt}#\nYou have invalid data!\nPlease observe:
            ${check1 == true ? "✗ Body Fat" : "✓ BodyFat%"}
            ${check2 == true ? "✗ Weight" : "✓ Weight"}`);
    }
    if (check1 == false && check2 == false) {

        // Equation for Calories Burned.
        // Katch and McArdle equation.
        //
        //  $370 + (21.6 * \text{LBM (Lean Body Mass) [kg] = weight - (weight * bodyfat\%)})$ 
        let fatMass = (inpCalBodyFat.value/100) *
convertWeight(inpCalWeight.value,metricCal.value); // Equation to use in lean mass.
        let leanMass = convertWeight(inpCalWeight.value,metricCal.value) - fatMass; // Used
for the end result.
        let calSult = 370 + (21.6 * leanMass); // Work out the basal metabolic rate.
        calFatAnswer.textContent = `Your Fat Mass is ${fatMass.toFixed(2)}`;
        calLeanAnswer.textContent = `Your Lean Body Mass is ${leanMass.toFixed(2)}`
        calSultAnswer.textContent = `Your BMR is ${calSult.toFixed(2)}`;
        resultCal.style.display = "block";
        inpBMIWeight.value == inpCalWeight.value;
    }
});

// BodyFat% Validation
function validateFatGender() { // Validate the gender
    if (document.querySelector(`input[name="gender"]:checked`) == null) { // Disable button if
gender isn't selected.
        return btnResFat.disabled = true;
    }
    if (document.querySelector(`input[name="gender"]:checked`) != null) {
        return btnResFat.disabled = false; // Enable button if there is a selected gender.
    }
}
function validateFatAge() {

```

```

    if (inpFatAge.value == null | inpFatAge.value == "" | inpFatAge.value > 115 | inpFatAge.value < 1 ) {
        return btnResFat.disabled = true;
        //console.log("boo!");
    }
    if (inpFatAge.value >= 1 && inpFatAge.value <= 115 && inpFatAge.value != null) {
        return btnResFat.disabled = false;
        //console.log("Congrats");
    }
}

function validateFatBMI(){
    if (inpFatBMI.value == null | inpFatBMI.value > 40 | inpFatBMI.value < 15) {
        return btnResFat.disabled = true;
    }
    if (inpFatBMI.value < 40 && inpFatBMI.value >= 15) {
        return btnResFat.disabled = false;
    }
}

// Disabling the buttons to prevent premature activation.
btnResFat.disabled = true;
btnResCal.disabled = true;
btnResBMI.disabled = true;
btnResFat.addEventListener('mouseover',function() { // On mouse over of the button, check the
radio buttons.
    validateFatGender();
    validateFatAge();
    validateWeight(btnResFat,inpFatWeight,metricFat);
    validateFatBMI();
    // console.log("Checked.");
});
// this is in case of the BMI already being entered from the previous calculator.
inpFatAge.addEventListener('input',function() { // Input to for those who use keyboard only to
browse the web.
    validateFatGender();
    validateFatAge();
    validateWeight(btnResFat,inpFatWeight,metricFat);
    validateFatBMI();
});
    // If the user enters a value in the BMI section.
inpFatBMI.addEventListener('input',function() { // Input to for those who use keyboard only to
browse the web.
    validateFatGender();
    validateFatAge();
    validateWeight(btnResFat,inpFatWeight,metricFat);
    validateFatBMI();
});

// If the input is allowed but other values have not been entered.
let attempt = 0;

```

```

btnResFat.addEventListener('click',function() {
  let check1 = validateFatGender();
  let check2 = validateFatAge();
  // let check3 = validateWeight(btnResFat,inpFatWeight,metricFat); // double check the results for
  comparison
  // Deprecated check3 due to not needing weight for the equation as BMI is needed for this
  already.
  let check4 = validateFatBMI();
  if (check1 == true | check2 == true /*check3 == true*/ | check4 == true) { // If one of the
  results are true (meaning wrong)
    resultFat.style.display = "none";
    attempt++;
    // Prompt a message specifying which fields need to be appropriated.
    alert(`Attempt ${attempt}#\nYou have invalid data!\nPlease observe:
    ${check1 == true ? "✗ Gender" : "✓ Gender"}
    ${check2 == true ? "✗ Age" : "✓ Age"}
    ${check4 == true ? "✗ BMI" : "✓ BMI"}`);
  }
  if (check1 == false && check2 == false && /*check3 == false*/ check4 == false) {
    let attempt = 0;
    let eGender = document.querySelector(`input[name="gender"]:checked`).value == 1 ? 1 :
    0; // 1 male, 0 female.
    // ^^ equation gender value.
    let BFP = inpFatAge.value < 17 ? (1.2 * inpFatBMI.value) + (0.23 * inpFatAge.value) - (10.8
    * eGender) - 0.54 : (1.51 * inpFatBMI.value) - (0.7 * inpFatAge.value) - (3.6 * eGender) - 0.54;
    // The equation changes from child to adult as there are different numbers based on age.
    // 1.2 * inpFatBMI.value + 0.23 * inpFatAge.value - 10.8 * eGender - 0.54
    fatBod.textContent = `Your body fat percentage is ${BFP.toFixed(2)}%`;
    // Add bodyfat% into calorie calculator...
    inpCalBodyFat.value = `${BFP.toFixed(2)}`;

    resultFat.style.display = "block";
  }
  //let inpFatGender = document.querySelector(`input[name="gender"]:checked`);
  //console.log(`You are ${inpFatGender.value}`);
});

// Components that are not needed...
inpFatWeight.style.display = "none";
metricFat.style.display = "none";
fatMR.style.display = "none";
fatBF.style.display = "none";

```

// Below is a previous solution which didn't work as it should have in my design / head.

```

/** btnBMI.addEventListener('click', function() {decision = 1});
btnCal.addEventListener('click', function() {decision = 2});
btnFat.addEventListener('click', function() {decision = 3});
if (decision == 1) {

```

```

conBMI.style.display = "block";
console.log("First");
conCal.style.display = "none";
conFat.style.display = "none";
}
if (decision == 2) {
    console.log("Second");
    conBMI.display = "none";
    conCal.display = "block";
    conFat.display = "none";
}
if (decision == 3) {
    console.log("Third");
    conBMI.display = "none";
    conCal.display = "none";
    conFat.display = "block";
}
/** btnBMI.onclick = decision = 1;
btnCal.onclick = decision = 2;
btnFat.onclick = decision = 3;
*/

/**switch(decision) {
    case '1':
        btnBMI.display = "block";
        btnCal.display = "none";
        btnFat.display = "none";
        break;
    case '2':
        btnBMI.display = "none";
        btnCal.display = "block";
        btnFat.display = "none";
        break;
    case '3':
        btnBMI.display = "none";
        btnCal.display = "none";
        btnFat.display = "block";
        break;
    default:
        btnBMI.display = "block";
        btnCal.display = "none";
        btnFat.display = "none";
} */
// If statements for changing the content shown.

```

contact.js

The following is all of the code and comments for this script featured in the contact.php webpage.

```
// This is for the contact page, validating the user input.
// Define page elements.
let btnSend = document.getElementById('btn-send'); // The button.
let inpEMail = document.getElementById('email'); // The email box.
let inpName = document.getElementById('name'); // the user's name
let inpText = document.getElementById('write2us'); // The text area.
let inpPron = document.getElementById('select-pronouns'); // Pronouns selection.
//let inpFile = document.getElementById('file');
// A list of accept email providers for a check when verifying the email.
let emailProvider =
['gmail.com','outlook.com','aol.com','yahoo.com','icloud.com','protonmail.com','yandex.com','mail.c
om','hotmail.com'];
// Validation of user's inputs.
function validatePronouns() {
  if (inpPron.value == 0) {
    return true; // Validation TRUE means it's incorrect.
  }
  if (inpPron.value <= 6) {
    return false; // No issues so FALSE.
  }
}
function validateEmail () {
  // https://www.utf8-chartable.de/
  // @ in unicode is equal to 0040, using the unicode escape sequence.

  // If a split can be made, it is true. If not, it is false.
  let atSymbol = inpEMail.value.split('\u0040').pop() != inpEMail.value ? '\u0040' : false;
  let atDot = inpEMail.value.split('\u002E').pop() != inpEMail.value ? '\u002E' : false;
  // ^^^ The value must be different if input is given, but incorrectly.
  // This prevents triggering success code with bad / undesired values.
  // The last part of the email to solve is the extension after the dot.
  let emailName = false;
  let atExtension = false; // default values used to break down the email.
  let atDomain = false; // Less confusion when validating the integrity of the email.
  // If the atSymbol (@) is true.
  if (atSymbol != false && atDot != false) { // Validating e-mail syntax.
    // Get the email name.
    // Breaking down the email for validating if they exist within the text box.
    emailName = inpEMail.value.toLowerCase().split('\u0040').reverse().pop();
    atExtension = inpEMail.value.toLowerCase().split('\u002E').pop();
    atDomain = inpEMail.value.toLowerCase().split('\u002E').reverse().pop().split('\u0040').pop();
    // ^^^ Requires multiple splits to get values inbetween @ (\u0040) and . (\u002E)
    // Get the email extension.
  }
  if (emailName == false | atExtension == false |
    atDomain == false | atSymbol == false | atDot == false) {
```

```

        return true;    // return true if not correct syntax or existence.
    }
    // Code for if there is text when there is an @ and DOT.
    // filtering email providers to avoid email spam from less true emails.
    let validMail = false;
    let i = emailProvider.length;
    for (i in emailProvider[i]) {

    }
    /* for (i; emailProvider[i].value != atExtension; i--) {
        if (atDomain != emailProvider[i].value) {
            --i;
            console.log(i);
        }
        if (atDomain == emailProvider[i].value) {
            validMail = true;
            break;
        }
        if (i < emailProvider[i].length) {
            return null; // Say the email provided is not accepted.
        }
    }*/
    // Redundant to reduce code ^^^
    // filter the email with a loop through accepted clients.

do {
    --i;
    if (emailProvider[i] != atDomain+atDot+atExtension) {
        validMail = false; // If nothing found in loop, it is false.
    }
    else {
        validMail = true; // Once true, give the valid mail truth status.
    }
    // console.log(`Reading ${i}. ${atDomain+atDot+atExtension} isn't equal to $
    {emailProvider[i]}`);
    }while(emailProvider[i] != (atDomain+atDot+atExtension) && i != 0)
    // Loop through the array until the domain and extension are equal.

    //console.log(validMail);
    if (!validMail) {
        return null; // Tell the user their email provider is not supported.
    }

    if (emailName != false && atDomain != false && atExtension != false && validMail != false) {
    // This solves the no text with symbols issue.
        //alert(`Success ${emailName}${atSymbol}${atDomain}${atDot}${atExtension}`);
        return false;
    }
}

```

```

}
function validateName () {
    // get the length of the text's value and compare it against the inbetween values that are accepted.
    if (inpName.value == null | inpName.value == "" | inpName.value.length > 50 |
inpName.value.length <= 2) {
        return true;
    }
    if (inpName.value.length <= 50 && inpName.value.length > 2) {
        return false;
    }
    // If input name is inbetween 50 and 2, and is not null, return a false.
}
function validateText () {
    if (inpText.value.length >= 1500) {
        return true;
    }
    if (inpText.value.length < 100) {
        return null; // null if there is no text worth sending.
    }
    else {
        return false; // false if there is no issue.
    }
}
}
// https://www.w3docs.com/snippets/javascript/how-to-get-file-extensions-with-javascript.html
//function validateFileType () {
    //let fileExt = inpFile.value.split('.').pop().toLowerCase();
    // ^^ the file extension value is made lowercase and split from using the .split
    // JS built-in function, the pop gets rid of the filename itself.
    //console.log(fileExt);
    /* if (fileExt != 'png' | fileExt != 'jpg' | fileExt != 'jpeg' | fileExt != 'gif' |
fileExt != 'bmp' | fileExt != 'raw') {
        return true;
    }

    // equal to null if no file, false if there is a file that is supported.
    fileExt == null | fileExt == "" | fileExt == undefined ? null : false;

    if (fileExt != "png" | fileExt != 'jpg' | fileExt != 'jpeg' |
fileExt != 'gif' | fileExt != 'bmp' | 'fileExt' != 'raw') {
        console.log("not supported.");
        return true; // Return true if the values aren't the desired types.
    }
    if (fileExt == null) {
        console.log(fileExt);
        return null; // Files are optional, so null would be checked first in case
        // of no files being given.
    }
    if (fileExt == false) {
        console.log("supported.");
        return false;
    }
}

```

```

    } */
  //}

  // This is the event which will be listened for.
  btnSend.addEventListener('click',function(){ // This is where validation will be checked.
    // Checks will be placed by here.
    let check1 = validatePronouns();
    let check2 = validateName(); // will be equal to true or false.
    let check3 = validateText();
    // let check4 = validateFileType(); // null is a possibility here due to optional.
    let check5 = validateEmail(); // Null possibility due to unsupported provider.
    if (check1 == true | check2 == true | check3 == null | check3 == true | check5 == null | check5
    == true) {
      alert(`There are issues with your message:
      ${check1 == true ? "✗ \tSelect pronouns" : "✓ \tValid Pronouns"}
      ${check2 == true ? "✗ \tInappropriate name" : "✓ \tAppropriate name"}
      ${check3 != null ? (check3 == true ? "✗ \tMessage too long" : "✓ \tAppropriate
message") : "✗ \tMessage is too short"}
      ${check5 != null ? (check5 == true ? "✗ \tEmail doesn't exist" : "✓ \tAppropriate Email") :
"✗ \tEmail provider is not supported"}`);

    }
    else {
      confirm("Do you want to send the message?","Yes","No");
    }
    // If it is null, provider not supported, if true, no email entered appropriately. If it is false,
    nothing is wrong.
    // No file is provided if the answer is null from the check4.
  });
  // Fun contact page features.

```