1712540
Web Development Project
Deliverable 2
Words: 2352

PC
2448895-800
2449022-800
MAC
250883
251014

Content

Introduction

The Slice of Life

(excerpt from deliverable 1)

Target Market: Students

Persona/User: A student who likes to order their food quickly and hassle free.

Scenario: Quick lunch in between lectures (2 hours free time)

The persona needs to order their food quickly, because time is limited in between lectures. We assume that they do not have enough time to cook something big, but enough time to order some fast delivered food. Students also operate on a low budget, so the food must either be low cost or the website should offer a variety of discounts. The persona also fits into a variety of different dietary options, variety of pizza/toppings should also be available.

Synopsis of needs:

- Easy and fast navigation
- The least number of clicks from start of website to checkout
- Loyalty system or discount deals
- A wide variety of pizza/sides
- Payment by cash on delivery (quicker than entering card details)

Note, not all needs highlighted in deliverable 1 were fulfilled due to time reasons. However, some aspects were greatly improved (i.e. speed of navigation)

Development

Homepage

I started off by adding the logo and banner ads using <img>

```
<img class="headerLogo" src="resources/slideoflife.png"
alt="Slice of Life logo"/>
<img id="advertBanner"  src="resources/mobileBanner.png"
alt="mobileBanner"/>
```

I gave the logo a class instead of an id by mistake (I needed an id for future JavaScript) however, this was corrected once I began implementing the script to determine the device type.

I initially did all the CSS using the id tag. However, once the device checking script was implemented I used the id as a unique identifier and the classes to hold the CSS. This allowed me to switch between classes depending on the device.

```
.headerLogo {
    width: 75%;
    max-width: 800px;
    }
```

```
#advertBanner{
    z-index: -1;
    width: 100%;
    position: absolute;
    top: 75%;
    left: 50%;
    transform: translate(-50%, -50%);
    }
```

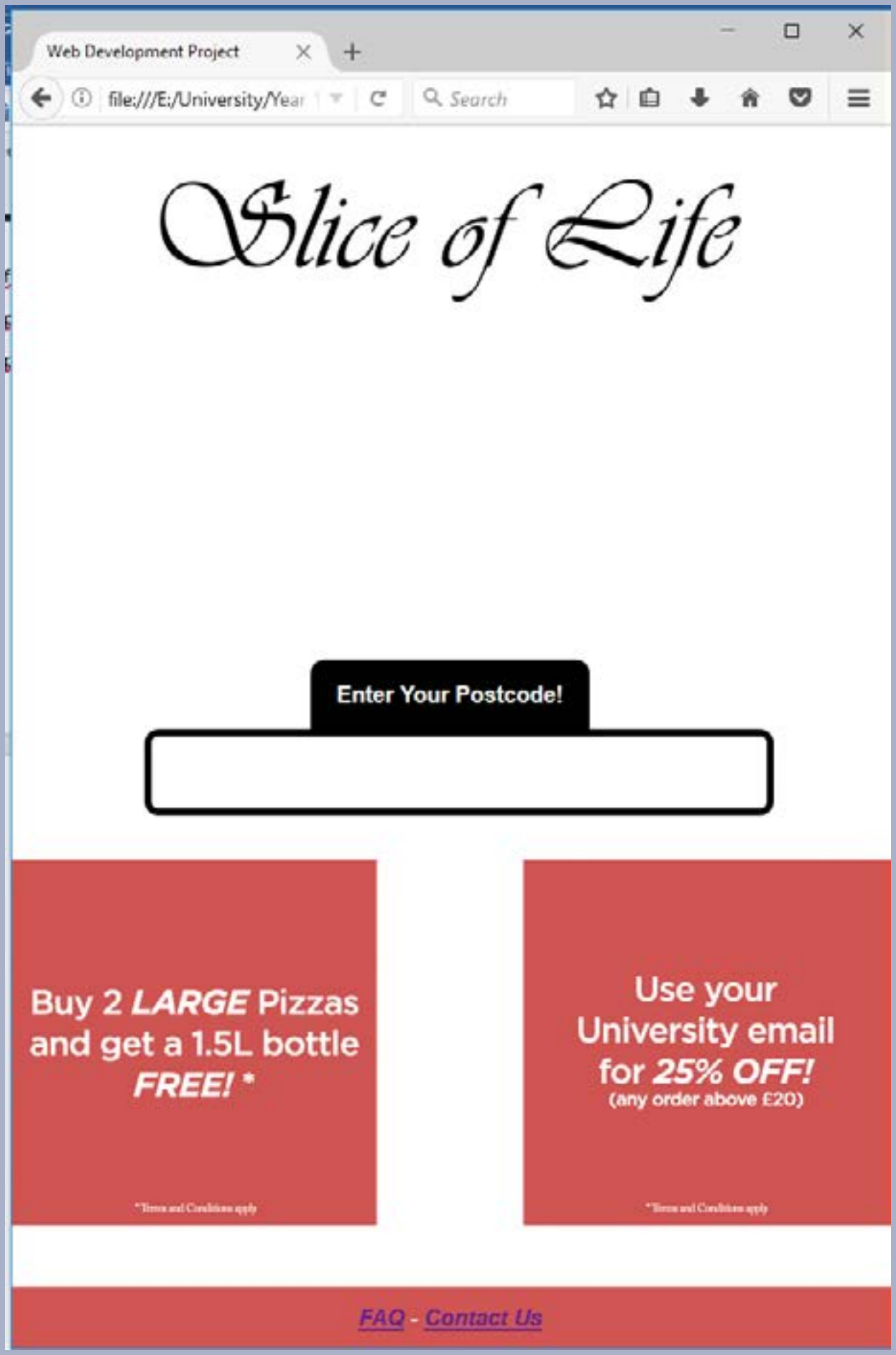
Once completed, I added a postcode form where the user can input their postcode to go to the next screen. JavaScript is used to check whether the postcode is correct.

```
.formParent{
    width: 50vh;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    text-align: center;
    }
.postcodeForm{
    display: inline-block;
    width: 100%;
    color: grey;
    border-radius: 10px;
    border-color: black;
    border-width: 5px;
    box-shadow: none;
    border-style: solid;
    height: 50px;
    max-height: 50px;
    max-width: 500px;
```

```
<div class="formParent">
    <div class="postcodeInfo">
        <strong>Enter Your Postcode!</strong>
    </div>
    <form>
        <input class="postcodeForm" type="text"
        name="postcode">
    </form>
</div>
```

```
<script> //Checking form
    function formCheck(){
        var postcode = document.getElementById("postcodeForm").elements[0].value;
        if(postcode.length == 6 || postcode.length == 7 || postcode.length == 8){
            window.open("menu.html");
            window.close();
        } else{
            alert("'"+postcode+"' invalid. Try again.");
        }
    }
</script>
```

Before implementing the device checking script, I exclusively tested my website for mobile phones (specifically iPhone 6 at 375x667).
All viewport information gained from http://mediag.com/news/popular-screen-resolutions-designing-for-all/
All browsers resized with http://resizemybrowser.com/



The resolution checking script was written using JavaScript and is always the last thing to be activated on site load. This allows any elements that need to be resized/changed to initialise. If the script was loaded at the top of the body, the website would not react to the resolution because any elements that should be effected have not been created yet.

```
var width = window.outerWidth;
var height = window.outerHeight;
//document.write(width, "x", height);
if(width > height){
    var landscape = true;
    var w = height;
    var h = width;
} else{
    var landscape = false;
    var w = width;
    var h = height;
}
if(w < 500 && h < 800){
    //document.write("MOBILE");
    mobileSite();
} else if(w < 1100 && h < 1400){
    //document.write("TABLET");
    tabletSite();
} else{
    //document.write("DESKTOP");
    desktopSite();
}
function mobileSite(){
    document.getElementById("headerLogo").className = "origHeaderLogo";
    document.getElementById("advertBanner").className = "mobileBanner";
    if(landscape == true){
        document.getElementById("advertBanner").style.display = "none";
    }
}
function tabletSite(){
    document.getElementById("headerLogo").className = "smallHeaderLogo";
    document.getElementById("advertBanner").className = "tabletBanner";
    if(landscape == true){
        document.getElementById("advertBanner").style.display = "none";
    }
}
function desktopSite(){
    document.getElementById("advertBanner").style.display = "none";
}
```

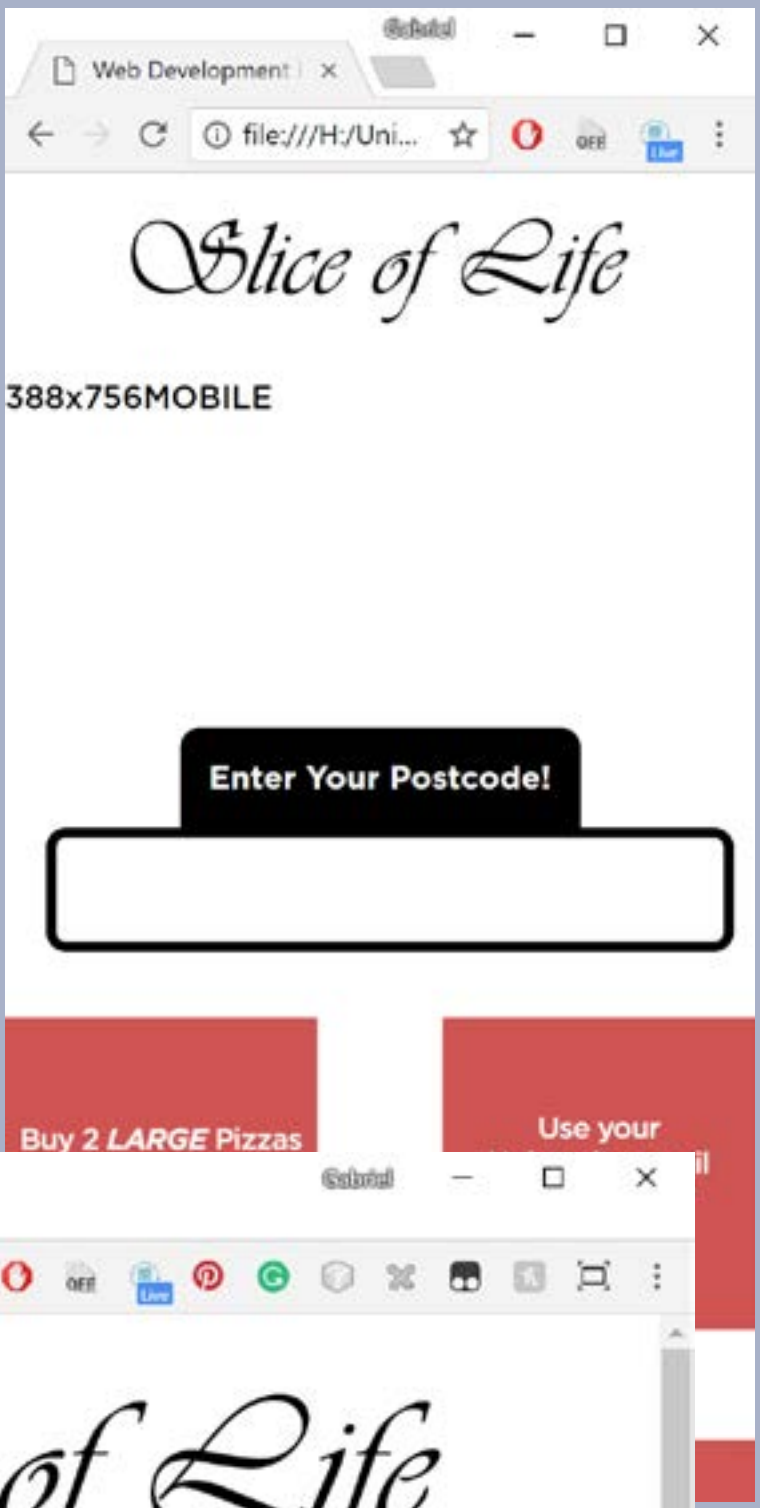
The script gets the width and height of the browser and then compares the two to see if the browser is landscape or portrait.

The variables **w** and **h** are then assigned depending on the previous factor. **w** and **h** correspond to the width and height of the device; in landscape mode the width of the browser is the height of the phone; **h = width;**.

After this, the variables **w** and **h** are compared against *x* and *y* resolution values (i.e. 500 and 800 respectively) to determine the screen size and therefore, the device type.

The device checker working using a "mobile phone" resolution.

The printed resolution and device type were only for testing purposes (using document.write) and are not visible in the final website.



Menu

Using a tutorial from https://www.w3schools.com/howto/howto_js_slideshow.asp I created a "slideshow" style pizza selection. I originally wanted it to look like a wheel that rotates, I settled for the fade in/out animation that the tutorial suggested because a rotating picture wheel was out of my skill set for the time being.

```
<div class="pizzaImage fade"> <img src="resources/pizzas/custom.png"
style="width:100%">
  <button class="pizzaButton">Create Your Own</button>
</div>
```

Each pizza image is added as an individual element and cycled through using two buttons and JavaScript.

I added a **Customise** or **Create Your Own** button depending on the pizza selected. If the selected pizza is pre-made (e.g. Pepperoni) the button says **Customise** and if they selected the custom pizza (a greyed out image with a question mark) they can create their pizza from scratch with the **Create Your Own** button.

```
//PIZZA SELECTOR
var pizzaIndex = 1;
showPizza(pizzaIndex);

function changePizza(n) {
    customisationOff("switcher");
    showPizza(pizzaIndex += n);
}

function showPizza(n) {
    var i;
    var pizzas = document.getElementsByClassName("pizzaImage");
    if (n > pizzas.length) {
        pizzaIndex = 1
    }
    if (n < 1) {
        pizzaIndex = pizzas.length
    }
    for (i = 0; i < pizzas.length; i++) {
        pizzas[i].style.display = "none";
    }
    pizzas[pizzaIndex - 1].style.display = "block";
    pizzaTypeSaver(n); //Going to basketHandler.js
}
```

The function names are self-explanatory:

showPizza displays the pizza image according to it's location in the pizzas array (all elements with the **pizzaImage** class).

changePizza shows the image next/before the current one in the array.

The original tutorial also included dots and an extra function to display text. I did not need these for my website so I did not include them in my code.

My original plans included an extra page for customisation. I felt that it would be much quicker to navigate the website if I included it on the same page as a pop-in element when the **Customise/Create Your Own** button was clicked.

```
if (selectedButton.id == "sizeOne" || selectedButton.id == "crustOne" || sel
    try {
        selectedButton.classList.remove("choiceButton");
        selectedButton.className = "chosenButton";
        optionTwo.classList.remove("chosenButton");
        optionTwo.className = "choiceButton";
        optionThree.classList.remove("chosenButton");
        optionThree.className = "choiceButton";
        optionFour.classList.remove("chosenButton");
        optionFour.className = "choiceButton";
        optionFive.classList.remove("chosenButton");
        optionFive.className = "choiceButton";
    } catch (err) {
        //alert(err);
    }
    switch(selectedButton.id){
        case "sizeOne":
            switch(sizeSelected){
                case "med":
                    subtotal = subtotal-5;
                    break;
                case "lar":
                    subtotal = subtotal-10;
                    break;
                case "xlar":
                    subtotal = subtotal-15;
                    break;
            }
            sizeSelected = "small";
            document.getElementById("subtotalPrice").innerHTML = "£"+subtota
            break;
    }
    option = 1;
```

When one of the white buttons is clicked, it turns black. The JavaScript checks what type of button was clicked, and if any other buttons of that category are already selected. It then changes the price and deselects any currently selected options. This makes sure the user doesn't have a pizza with multiple size, crust or sauce values. The toppings can be selected simultaneously in case they want to add multiple extra toppings.

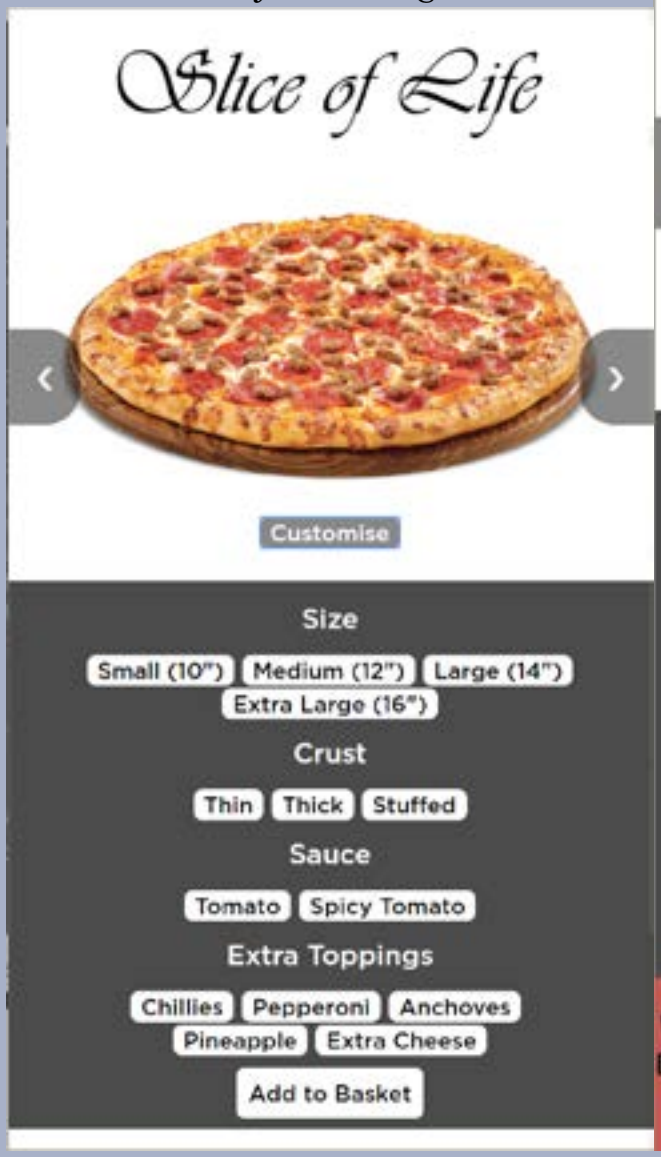
I used try/catch statements to catch any errors and display them using an alert. This was purely to streamline bug testing and is not present in the final website.

I later added the footer, which hides the white bar at the bottom for every resolution that I tested (mobile, tablet, desktop; portrait, landscape).

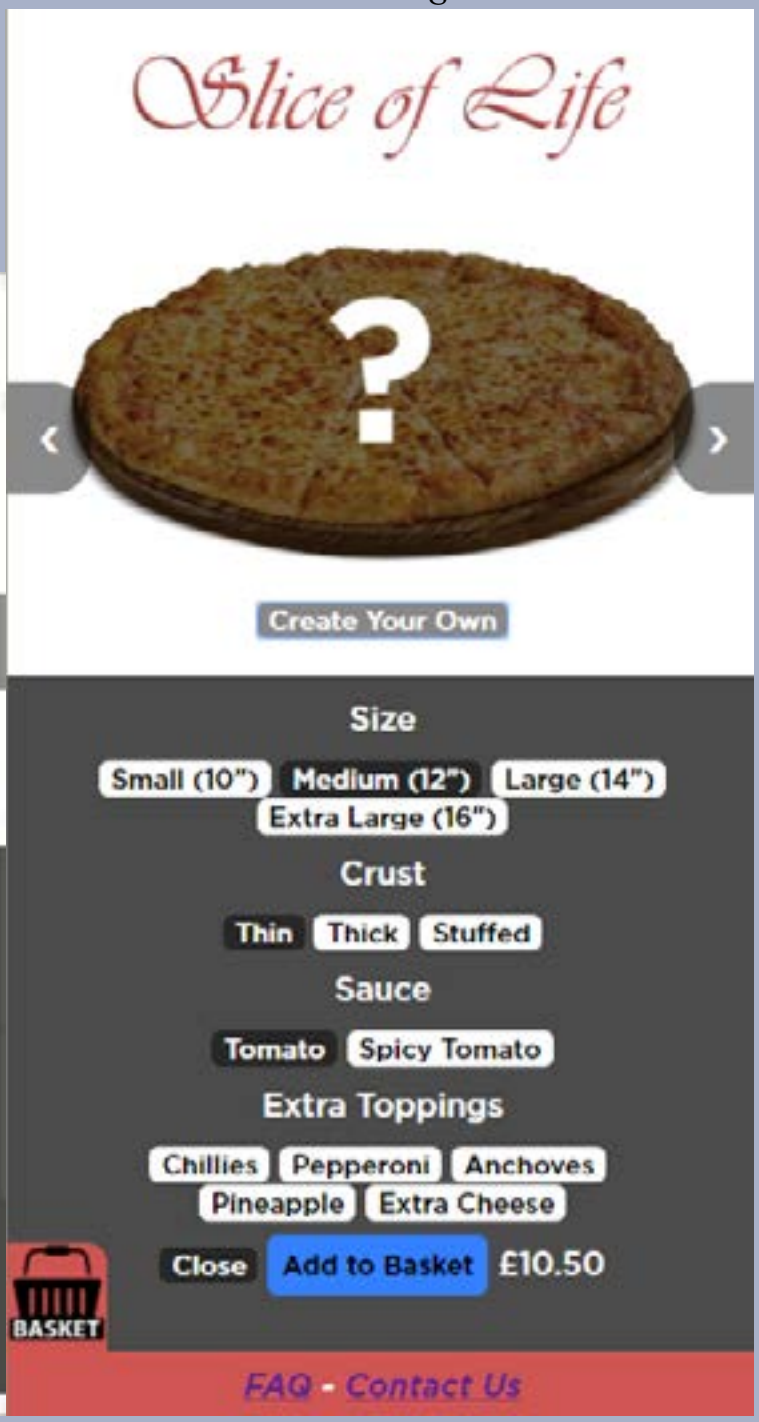
I used several fresh eyes to test my website and made small changes to reflect difficulties they faced.
Difficulties:
- No way of seeing price of current pizza
- Add to Basket button not easily visible
- No obvious way to close the customisation menu
- Most tried to customise the pizza by clicking the image instead of the button
- There weren't default options

I attempted to fix the problems they faced and I added the possibility to open the customisation menu by clicking on the pizza itself.
The basket wasn't made at this point in testing. The second image is the final website once everything was implemented.

Before Testing



Final Page



Basket/Checkout

Once again I decided to keep the basket on the same page in order to make it quicker to add extra items from the menu (you don't need to change pages constantly).

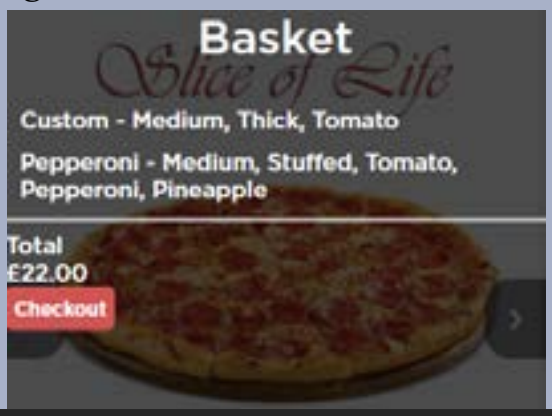
```
var pizza = [];
pizza[0] = "Medium"; //Size
pizza[1] = "Thick"; //Crust
pizza[2] = "Tomato"; //Sauce
pizza[3] = "none"; //Topping 1
pizza[4] = "none"; //Topping 2
pizza[5] = "none"; //Topping 3
pizza[6] = "none"; //Topping 4
pizza[7] = "none"; //Topping 5

function addChoice(selectedButton, option) {
    var choiceType = selectedButton.id;
    if (choiceType.includes("size") == true) {
        switch (option) {
            case 1:
                pizza[0] = "Small";
                break;
            case 2:
                pizza[0] = "Medium";
                break;
            case 3:
                pizza[0] = "Large";
                break;
            case 4:
                pizza[0] = "Extra Large";
                break;
        }
    } else if (choiceType.includes("crust") == true) {
        switch (option) {
            case 1:
                pizza[1] = "Thin";
                break;
            case 2:
                pizza[1] = "Thick";
                break;
            case 3:
                pizza[1] = "Stuffed";
                break;
        }
    } else if (choiceType.includes("sauce") == true) {
        switch (option) {
            case 1:
                pizza[2] = "Tomato";
                break;
            case 2:
                pizza[2] = "Spicy Tomato";
                break;
        }
    } else if (choiceType.includes("topping") == true) {
        switch (option) {
            case 1:
                pizza[3] = "Chillies";
                break;
            case 2:
                pizza[4] = "Pepperoni";
                break;
            case 3:
                pizza[5] = "Anchovies";
                break;
            case 4:
                pizza[6] = "Pineapple";
                break;
            case 5:
                pizza[7] = "Extra Cheese";
                break;
        }
    }
}

function removeTopping(option) {
    switch (option) {
        case 1:
            pizza[3] = "none";
            break;
        case 2:
            pizza[4] = "none";
            break;
        case 3:
            pizza[5] = "none";
            break;
        case 4:
            pizza[6] = "none";
            break;
        case 5:
            pizza[7] = "none";
            break;
    }
}
```

Each pizza is temporarily assigned the array **pizza** while the customisation menu is open. Each entry in the array corresponds to another option (top of the image left).
The array is then printed to the basket once **Add to Basket** button is clicked.

The basket is just a 50% alpha black that can be toggled using the basket button. The array is printed into the div.

This screenshot was my first test.

I fleshed out the basket page and added a checkout button.




```
<!-- BASKET -->
<div id="basket" class="basketContainer">
    <div class=titleContainer><a class="basketTitle">Basket</a></div>
    <br>
    <a id="textContainer"></a>
    <hr>
    <a>Total</a> <br>
    <a id="totalPrice">£0.00</a> <br>
    <button id="checkoutButton" class="checkout" onclick="toggleCheckout()"><strong>Checkout</strong></button>
```

The HTML component was very simple because most of the code is back-end.
The is where the items are printed. This is done in JavaScript by creating a new text element for each item in the basket.

```
function writeToBasket(subtotal, type) {
    var basketDisplay = document.getElementById("textContainer");
    var text = document.createElement("p");
    var textContent = "";
    switch(type){
        case "pizza":
            for (i = 0; i < 9; i++) {
                if (i == 0) {
                    textContent = pizza[i] + " - ";
                } else if (i == 1) {
                    textContent = textContent + pizza[i];
                } else if (pizza[i].includes("none") == false) {
                    textContent = textContent + ", " + pizza[i];
                }
            }
            break;
        case "extra":
            textContent = extra;
            break;
    }
    var textBody = document.createTextNode(textContent);
    text.appendChild(textBody);
    basketDisplay.appendChild(text);

    total = total+subtotal;
    document.getElementById("totalPrice").innerHTML = "£"+total.toFixed(2);
}
```

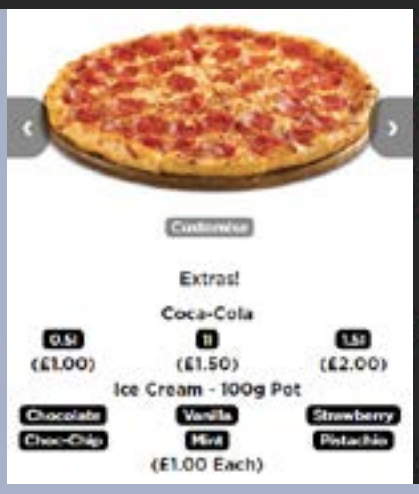
Whether they added an extra item (drink/dessert) or a pizza; the code prints differently. In order to print the pizza array in an aesthetic way I had to use a for loop to cycle through each item in the array and concatenate them in the **textContent** string accordingly. For example, if the extra topping is **none** (i.e. no topping selected) it will not print that because it is not necessary.
I create a new text element for every item in the basket using **document.createTextNode** because this was the only way I could add new items without overwriting what was already there.

The extra items on the menu (drinks/desserts) were added in a similar way to the customisation menu however there is no array because each item is an individual value and not an object that has multiple variations.

I used a table to neatly align the elements into rows and columns.

This area is also never toggled between hidden/shown because the customisation menu covers it when displayed due to it having the **position: absolute;** value.

```
<div id="extrasMenu">
  <p>Extras!</p>
  <table class="extrasTable">
    <tr>
      <th></th>
      <th><a>Coca-Cola</a></th>
      <th></th>
    </tr>
    <tr>
      <th><button id="smallCoke" class="extraButton" onclick="addExtra(this, this.classList.item(0));">0.5l</button></th>
      <th><button id="medCoke" class="extraButton" onclick="addExtra(this, this.classList.item(0));">1l </button></th>
      <th><button id="largeCoke" class="extraButton" onclick="addExtra(this, this.classList.item(0));">1.5l</button></th>
    </tr>
    <tr>
      <th>(£1.00)</th>
      <th>(£1.50)</th>
      <th>(£2.00)</th>
    </tr>
    <tr>
      <th></th>
      <th><a>Ice Cream - 100g Pot</a></th>
      <th></th>
    </tr>
    <tr>
      <th><button class="iceCream, extraButton" onclick="addExtra(this, this.classList.item(0));">Chocolate</button></th>
      <th><button class="iceCream, extraButton" onclick="addExtra(this, this.classList.item(0))">Vanilla</button></th>
      <th><button class="iceCream, extraButton" onclick="addExtra(this, this.classList.item(0))">Strawberry</button></th>
    </tr>
    <tr>
      <th><button class="iceCream, extraButton" onclick="addExtra(this, this.classList.item(0))">Choc-Chip</button></th>
      <th><button class="iceCream, extraButton" onclick="addExtra(this, this.classList.item(0))">Mint</button></th>
      <th><button class="iceCream, extraButton" onclick="addExtra(this, this.classList.item(0))">Pistachio</button></th>
    </tr>
    <tr>
      <th></th>
      <th>(£1.00 Each)</th>
      <th></th>
    </tr>
  </table>
</div>
```




```
/*EXTRAS*/
#extrasMenu {
    padding-bottom: 5vh;
}
.extrasTable {
    margin: 0 auto 0 auto;
}
.extraButton {
    background: black;
    border: white;
    border-radius: 5px;
    color: white;
}
.extraButton:active {
    background: rgba(0,0,0,0.5);
    border: white;
    border-radius: 5px;
    color: white;
}
```

I decided to keep the minimalistic style (less images) because this would allow for maximum cross-platform compatibility.

At first I wanted to show the checkout form at the same time as the basket. This proved to use too much space on the screen and because the basket has **position: absolute;** there was no way to scroll up and down.

```
<form id="checkoutForm" style="display: none;" method="post">
    <!-- User Details -->
    <div>
        <a>First Name</a>
        <br>
        <input class="checkoutForm" type="text" name="postcode">
    </div>
    <div>
        <a>Last Name</a>
        <br>
        <input class="checkoutForm" type="text" name="postcode">
    </div>
    <div>
        <a>Email</a>
        <br>
        <input class="checkoutForm" type="text" name="postcode">
    </div>
    <div>
        <a>Phone</a>
        <br>
        <input class="checkoutForm" type="text" name="postcode">
    </div>
    <div>
        <a>Delivery Instructions</a>
        <br>
        <textarea class="checkoutForm" type="text" name="postcode"></textarea>
    </div>
    <div>
        <input id="cardCheck" type="checkbox" name="card" onclick="payByCard()"><a>Pay by Card</a><br>
    </div>
    <hr>
    <!-- Card Payment Details -->
    <div id="cardPaymentContainer" style="display: none;">
        <div>
            <a>Card Number</a>
            <br>
            <input class="checkoutForm" type="text" name="postcode">
        </div>
        <div>
            <a>Expiry</a>
            <br>
            <select name="month">
                <option value="01">01</option>
                <option value="02">02</option>
                <option value="03">03</option>
                <option value="04">04</option>
                <option value="05">05</option>
                <option value="06">06</option>
                <option value="07">07</option>
                <option value="08">08</option>
                <option value="09">09</option>
                <option value="10">10</option>
                <option value="11">11</option>
                <option value="12">12</option>
            </select>
            <select name="year">
                <option value="2018">2018</option>
                <option value="2019">2019</option>
                <option value="2020">2020</option>
                <option value="2021">2021</option>
                <option value="2022">2022</option>
                <option value="2023">2023</option>
            </select>
        </div>
        <div>
            <a>CVV</a>
            <br>
            <input class="checkoutForm" type="text" name="postcode">
        </div>
        <hr>
    </div>
    <button class="checkout"><strong>Pay</strong></button>
</form>
```

The entire checkout "page" is within one form. I added a check-box where the user can pay with card (if it is not checked they can pay by cash). The div **cardPaymentDetails** is toggled between **display: hidden** and **block** depending on whether the check-box is checked or not.

The checkout form is initially hidden but when the button is clicked it toggles the form to **display: block;** and the basket to **display: hidden;** this ensures that the checkout form displays on screen for all devices and that the basket and the checkout are not visible at the same time.

```
//FRONT END CODE
var basketHidden = true;
function basketToggler() {
    if (basketHidden == true) {
        showBasket();
        basketHidden = false;
    } else {
        hideBasket();
        basketHidden = true;
    }
}

function showBasket() {
    document.getElementById("checkoutForm").style.display = "none";
    document.getElementById("textContainer").style.display = "block";
    var basket = document.getElementById("basket");
    basket.style.display = "block";
}

function hideBasket() {
    var basket = document.getElementById("basket");
    basket.style.display = "none";
}

function payByCard() {
    var cardPayment = document.getElementById("cardCheck").checked;
    if(cardPayment == true){
        document.getElementById("cardPaymentContainer").style.display = "block";
    }else{
        document.getElementById("cardPaymentContainer").style.display = "none";
    }
}
function toggleCheckout() {
    document.getElementById("textContainer").style.display = "none";
    document.getElementById("checkoutForm").style.display = "block";
}
```

The function names are self-explanatory:
basketToggler is called by the basket button and makes the basket visible or hidden depending if **basketHidden** is true or false respectively.
showBasket is the function that makes the checkout form hidden and the basket visible, this is called from **basketToggler**.
hideBasket does the opposite and is called from **basketToggler**.
(in retrospect I could have merged the first three functions together)
payByCard hides/shows the card detail forms depending on the state of the check-box.
toggleCheckout is called by the checkout button; hides the basket and shows the checkout form.

Testing

I periodically tested my website on mobile, tablet and desktop resolutions in landscape and portrait mode. In retrospect I should have also tested my website in portrait mode for desktop resolutions because there is a small minority of people who use their desktop screens in portrait mode. However, it is extremely unlikely that the target market (students) would be doing this.

The page that was tested the most was the menu page because it took the longest to develop. The homepage was used very little by peer testers because its only function is to accept the postcode of the user.

Peer testing was extremely useful when developing the menu because there were some features that I did not think of implementing and some that I thought were obvious but turned out to be hard to use. A more in depth explanation of the difficulties faced by testers and my changes is available on page 12.

Homepage:

The homepage responded well on all devices/resolutions I tested on, however, the postcode form was difficult to code because I was not sure on how to check the postcode to see if it is real.
I satisficed by checking the length of the postcode, and if it is more than 8 characters an error is displayed to the user. However, it is possible to enter anything into the form as long as it is less than 8 characters. It is also possible to enter a blank string.

Menu:

The menu page also responded well on all devices. Some changes had to be made such as disabling the logo on some devices/orientations as well as the size of certain elements. On larger screens (laptops and above) the left/right buttons stopped hugging the sides of the browser. This left them looking like they were floating in the middle of the page. I fixed this by rounding the corners slightly on tablet resolutions and above.

```
function mobileSite(){
    if(landscape == true){
        document.getElementById("headerLogo").style.display = "none";
    }
}
function tabletSite(){
    if(landscape == true){
        document.getElementById("headerLogo").className = "smallHeaderLogo";
        document.getElementById("headerLogo").style.display = "none";
        document.getElementById("prevButton").style.borderRadius = "10px 20px 20px 10px";
        document.getElementById("nextButton").style.borderRadius = "20px 10px 10px 20px";
    }
}
function desktopSite(){
    document.documentElement.style.overflow = 'hidden';  // Chrome, Firefox
    document.body.scroll = "no"; // IE
    document.getElementById("prevButton").style.borderRadius = "10px 20px 20px 10px";
    document.getElementById("nextButton").style.borderRadius = "20px 10px 10px 20px";
    if(landscape == true){
        if(height < 1200){
            document.getElementById("headerLogo").style.display = "none";
        }else if(h > 2500 && w > 1400){
            document.getElementById("headerLogo").style.width = "auto";
        }else{
            document.getElementById("headerLogo").style.width = "300px";
        }
    }
}
```

Evaluation

Overall, the website turnout much better than I had anticipated. There were some features that I planned in deliverable 1 that I either didn't have time to implement or didn't have the technical skill to code without heavily relying on other people's code or on templates (something I didn't want to do).

In deliverable 1, I planned to create separate pages for the checkout and basket. I decided to merge the menu, basket and checkout pages into one for two main reasons: I wasn't comfortable saving information on the client computer through cookies or temp files in order for the website to "remember" the basket/price across pages. I wasn't sure if the uni computer system would block this. Another reason was because I felt that it would be quicker to navigate the website if everything was on one page. This refers back to the persona outlined in deliverable 1 and on **page 4**; a student who wants to order food quickly.

A feature I really wanted to implement was a google maps feature that allowed the user to select their postcode from a map, however, I did not have the time to implement this. Had I started earlier, it is possible that I could have implemented this through some form of open source JavaScript plugin or Google plugin.
I also failed to implement the deals section where the user can select a deal for a discount, this was mainly due to skill level. E.g. I didn't know to detect that the user had selected two pizzas and give a notification that they can have a drink for free (and also provide the drink for free). Had I had more time, I would definitely have implemented this because I would have been able to become more proficient in JavaScript.

In terms of time, I gave myself enough time to make the website without stressing about the deadline. However, in retrospect I should have given myself more time because I miscalculated how long it would take to make everything outlined in deliverable 1.

In conclusion, I am happy with the way the website turned out and I believe that it follows the brief. My main disappointment was that I didn't give myself enough time to implement every feature that I wanted and that I didn't stick closer to my initial plans.

References

Page 10:
https://www.w3schools.com/howto/howto_js_slideshow.asp

Graphics

https://www.turner.com/pressroom/pizza-steve-pose-1