```html
</head>

<!-- Een responsive design betekent dat je design er goed uitziet op elk formaat scherm, bijv. telefoon en desktop. -->
<!-- Je kunt een design voor desktop apparaten maken en dit dan zo aanpassen dat de content nog goed blijft staan -->
<!-- wanneer je deze opent op je telefoon. Tegenwoordig gebruiken veel meer mensen hun telefoon om websites te bekijken -->
<!-- dus steeds meer mensen maken eerst een design voor de telefoon en passen deze daarna aan voor de desktop versie -->
<body>
    <header>
        <nav>
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="menu.html">Menu</a></li>
                <li class = "mobile"><a href="locations.html">Locations</a></li> <!-- De class mobile en desktop word aangemaakt zodat -->
                <li class = "mobile"><a href="contact.html">Contact</a></li> <!-- deze aangeroepen kunnen worden met de media query's -->
                <li class="logo"><a href="index.html">Artisan Bakery Logo</a></li>
                <li class = "desktop"><a href="locations.html">Locations</a></li>
                <li class = "desktop"><a href="contact.html">Contact</a></li>

            <!-- Door de class mobile en desktop te maken kun je ervoor zorgen dat de bovenste mobile class word weergegeven -->
            <!-- Dit doe je door middel van media query's die actief worden bij een bepaalde schermbreedte -->
            <!-- Zo kun je de mobile class laten zien als het scherm klein is en de desktop class wanneer je scherm groot is -->

            <!-- Je zou ook een hele nieuwe navigatie kunnen maken die verschijnd -->
            <!-- voor mobile of een voor desktop maar dan kopieer/plak je iets meer code -->
            </ul>
        </nav>
    </header>
    <section class="feautures">
        <figure>
            <img srcset="" src="images/breads.jpg" alt="amazing fresh baked bread">
            <figcaption>Fresh Baked Bread</figcaption>
        </figure>

        <figure>
            <img src="images/coffee.jpg" alt="fresh brewed coffee">
            <figcaption>Home Roasted Coffee</figcaption>
        </figure>

        <figure>
            <img src="images/goods.jpg" alt="home goods">
            <figcaption>Goods Market</figcaption>
        </figure>
    </section>
    <footer>
        123 Main Street Edwardsville, IL • 555-555-555 • us@me.com
    </footer>
```

Dit is de Html code van mijn Responsive Design Poc. Ik heb hiervoor de code van de Images Poc gebuikt omdat ik een website nodig had om dit op te testen. Ik heb dus van deze niet responsive website een responsive website gemaakt.

Hierboven in de code leg ik uit wat responsive inhoud. Ook heb ik twee extra <a> tags links toegevoegd in de navigatie balk. Deze twee zijn in eerste instantie ontzichtbaar (desktop/mobile). Wanneer de user switcht van desktop naar mobile worden de <a> links met de mobile class zichtbaar en die met desktop ontzichtbaar. Hierdoor zal uiteindelijk het logo onder de nav bar komen wanneer het scherm te klein is.

Hier rechts zie je de css die voor general styling van de website zorgt. Hier is niets nieuws aan.

Portfolio > 3. derde versie > pages > pocs > webdev > 8responsivedesign > # style.css > .feautures figure img

```css
body {
    color: white;
    font-family: Helvetica, Arial, sans-serif;
    margin: 0;
    padding: 0;
}
nav ul {
    margin: 0;
    padding: 0;
}
nav ul li {
    display: inline-block;
    padding: 60px 20px 0px 20px;
}
nav ul li a {
    color: white;
    text-decoration: none;
    text-transform: uppercase;
}
header .logo a {
    display: inline-block;
    text-indent: -9999999px;
    background-image: url("images/logo.png");
    background-size: 300px;
    width: 300px;
    height: 190px;
    background-repeat: no-repeat;
    position: relative;
    top: -55px;
}
a:hover {
    text-decoration: underline;
}
header {
    background-color: black;
    text-align: center;
    height: 400px;
    padding: 20px;
    background-image: url("images/banner.jpg");
    background-size: 100%;
    background-position: center;
}
```

```css
.feautures {
    color: black;

    display: flex;
    flex-direction: row;
    text-align: center;
    flex-wrap: wrap;  /* Dit zorgt ervoor dat je website responsive word. Als je scherm nu kleiner word zullen de afbeeldingen */
                      /* automatisch zichzelf onder elkaar zetten, maar niet netjes in het midden*/
    justify-content: center;      /* Zet alle children in het midden van de container */
                                  /* Dus alle figures in de feautures section worden nu mooi gecentered */
  /* Nu heb je dus je website al responsive gemaakt zonder media query's te gebruiken, door een slim design is dit dus niet altijd nodig! */
}
.feautures figure {

    /* width: 100%; /* Als je deze nu houd worden de figures onder elkaar gezet en krijgen ze ieder een eigen rij met daarop 100% */
                    /* Dit werkt dus niet goed met de flex-wrap en om ze nu gecentered te krijgen gebruiken we justify-content center */
    text-transform: uppercase;
}
.feautures figure img {
    width: 200px;
    border-radius: 50%;
    box-shadow: gray 00px 0px 10px;
    border: 2px solid white;
}
footer {
    background-color: black;
    color: gray;
    font-size: 15px;
    padding: 20px;
    text-align: center;
    margin-top: 25vh;
}

```

Hierboven is het tweede gedeelte van mijn Css code. Ik heb hier flexbox gebruikt. Door flex-wrap: wrap; word de content gewrapt. Dit betekent dat wanneer de website kleiner word, de content zichzelf automatisch onder elkaar zal zetten. De flex-wrap staat op de container waar alle content in zit die gewrapt moet worden.

Dit betent dat we geen widht: 100%; meer kunnen gebruiken op de afbeeldingen omdat dit dan tegen elkaar in werkt.

```css
@media screen and (max-width: 800px) { /* Dit is een media query. Als dat wat tussen de ( ) staat true is word alles wat  */
    .feautures {                       /* tussen de { } staat actief. Je kunt meerdere conditions gebruiken in één query  */
        background: pink;              /* door bijv. (min-width 400px) and (max-width 800px) tegebruiken */
                                       /* max-width 800px betekend dat dit actief is wanneer je scherm tot en met 800px breed is. */
                                       /* Dus bij een breedte van 800px of lager zal deze code actief worden */
    }
}

/* Onder de 550px worden beide query's actief. Als je deze twee om zou draaien zou de achtergrond alleen maar roze blijven */
/* omdat de laatste code de vorige overschrijft. Doordat de code nu zo staat zal onder de 800px breedte de achtergrond roze worden */
/* en onder de 550px breedte de achtergrond rood worden omdat dan onderstaande code actief word. */
/* Als de query's omgedraaid waren zou het scherm altijd roze worden omdat je altijd onder de 800px zit */
/* Om dit op te lossen kun je meerdere conditions gebruiken (min-width) and (max-widht) in dezelfde query */

/* Je begint dus met een groot scherm en als deze kleiner word word de query geactiveerd. Dit is wanneer je vanuit desktop */
/* naar mobiel toe werkt. Als je vanuit telefoon naar desktop werkt kan je min-widht gebruiken omdat je dan */
/* met een klein scherm begint en de code actief word als een minimale breedte overschreden is. Dus het scherm groter word */

@media screen and (max-width: 550px) {
    .feautures {
        background: red;
    }
}
```

Hierboven staan de eerste twee media queries die ik heb gemaakt. Om te testen of ze werkte heb ik alleen de achtergrond kleur verandert. Ik heb ook uitgelegd in de comments hoe deze precies werken en waar je rekening mee moet houden. Zo kan ik deze queries niet omdraaien omdat ze elkaar dan overschrijven. Hieronder staat de media query die ik heb gebruikt om de navigatie balk responsive te maken.

```css
/* Je kijkt op je website naar de breedte waarop je design niet meer mooi is. Als je op pagina inspecteren klikt kun je zien hoeveel */
/* pixels hoog en breed je scherm is rechts bovenin. Je ziet dat de navigatie verschuifd zodra de website 820px breedte bereikt */

/* Je kunt grote stukken code in een media query onderaan je normale code zetten of je zet voor iedere tag die code in kleinere */
/* query's onder de bijbehorende tag in de normale code. */

header .mobile {
    display: none; /* Hierdoor worden de linkjes met de class mobile ontzichtbaar, deze kun je vervolgens oproepen door media query's. */
}

@media screen and (max-width: 820px) { /* Deze code word actief bij een breedte onder de 820px */
    header .mobile {
        display: inline-block;  /* Dit zorgt ervoor dat de mobile class content weer zichtbaar word en in inline-block word gezet */
    }

    header .desktop {    /* Dit zorgt ervoor dat de desktop class nu niet meer zichtbaar is */
        display: none;
    }

    header {
        height: 280px; /* Dit zorgt ervoor dat de background hoogte verandert waardoor je geen herhalingen ziet als het scherm klein is */

        background-position: 0 0; /* Je zou de background position nog kunnen aanpassen als deze er niet goed uit zou zien op mobile */
        /* In de normale code staat background position centered. Als je hier nu 20px 20px aangeeft */
        /* zal de eerste waarde voor de horizontale as zijn en de tweede waarde voor de verticale as*/
    }

    /*
    .feautures {  /* Omdat we slim gebruik maken van de flex-wrap zijn de afbeeldingen in figures al responsive dus is dit niet nodig
        display: block;  Hierdoor worden de afbeeldingen onder elkaar gezet wanneer het scherm kleiner word dan 820px
    } */


                /* Omdat telefoons ook 4g gebruiken en je je website zo geoptimaliseerd mogelijk wilt maken kan je er soms voor kiezen om kleinere */
                /* afbeeldingen te laten zien bij een kleiner scherm. Dit doe je anders voor design afbeeldingen en content afbeeldingen (met img tag) */
    header {    /* Zo optimaliseer je foto's die gebruikt worden als design */

        background-position: 0 0; /* Je zou de background position nog kunnen aanpassen als deze er niet goed uit zou zien op mobile */
        /* In de normale code staat background position centered. Als je hier nu 20px 20px aangeeft */
        /* zal de eerste waarde voor de horizontale as zijn en de tweede waarde voor de verticale as*/

        background-image: url("images/bannersmall.jpg"); /* Hiermee word een andere afbeelding weergegeven */
        /* Omdat het origineel een erg grote foto was zou dit niet optimaal zijn voor telefoon gebruikers op een slechtere verbinding */
        /* Daarom kan je een kleinere versie van je foto inladen wanneer het scherm een klein formaat heeft */
    }

    /* Voor content images (die met een img tag) moet je in je html wat tags gaan toevoegen */
    /* Wanneer je dezelfde afbeelding wilt laten zien maar dan een kleinere versie ervan gebruik je de srcset attribute */
    /* Deze gebruik je door            <img src="images/breads.jpg" alt=amazing fresh baked bread">                    */
    /* te veranderen naar              <img srcset="" src="images/breads.jpg" alt="amazing fresh baked bread">          */


    /* Wanneer je een afbeelding wilt veranderen qua grootte of vorm. Je wilt er bijv een vierkant van maken ipv een cirkel */
    /* zetten we deze in de <picure> </picture> tag */
```

Dit is het laatste gedeelte van mijn Css code. Deze code is voor het responsive maken van mijn navigatie balk.

In het begin leg ik uit wat een goede manier is om een website responsive te maken. Eerst zorg ik ervoor dat alle <a> tages met de .mobile classnaam ontzichtbaar worden. Vervolgens maak in een media query.

Hierin zet ik de screensize op max-width 820px omdat dit het punt is waarop de navigatie balk niet meer mooi is. Wanneer dit actief is worden de <a>tags met .mobile class zichtbaar door inline-block en die met .desktop ontizchtbaar.

Het is niet nodig om de afbeeldingen onder elkaar te zetten omdat dit al gefixt is door flexbox.

Ook heb ik ervoor gezorgd dat een kleinere versie van de afbeelding word geladen waardoor je minder internet gebruikt. Vooral voor mobile users is dit handig omdat hun vaak gebruik maken van een databundel die op kan gaan.