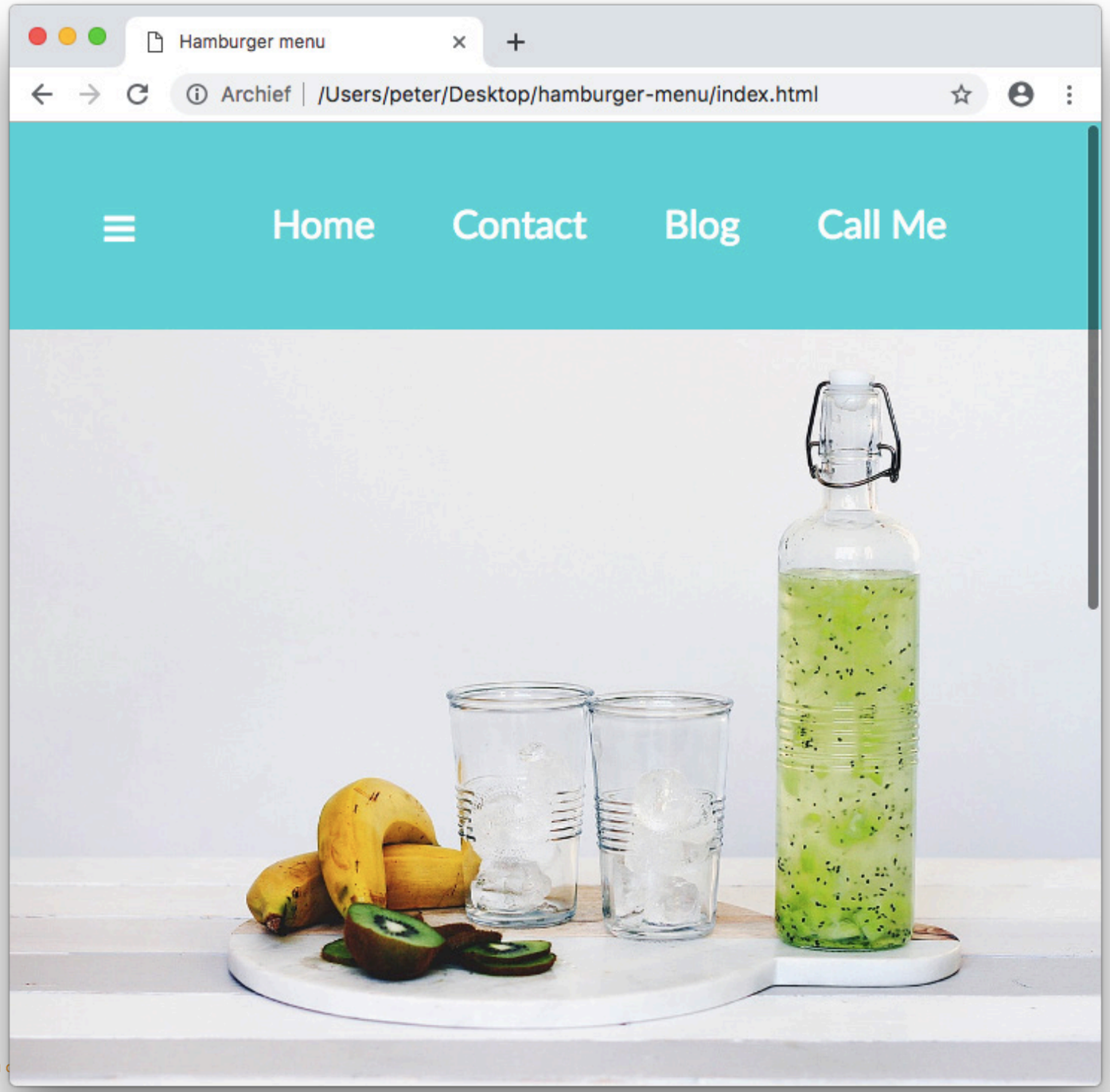
NAVIGATION

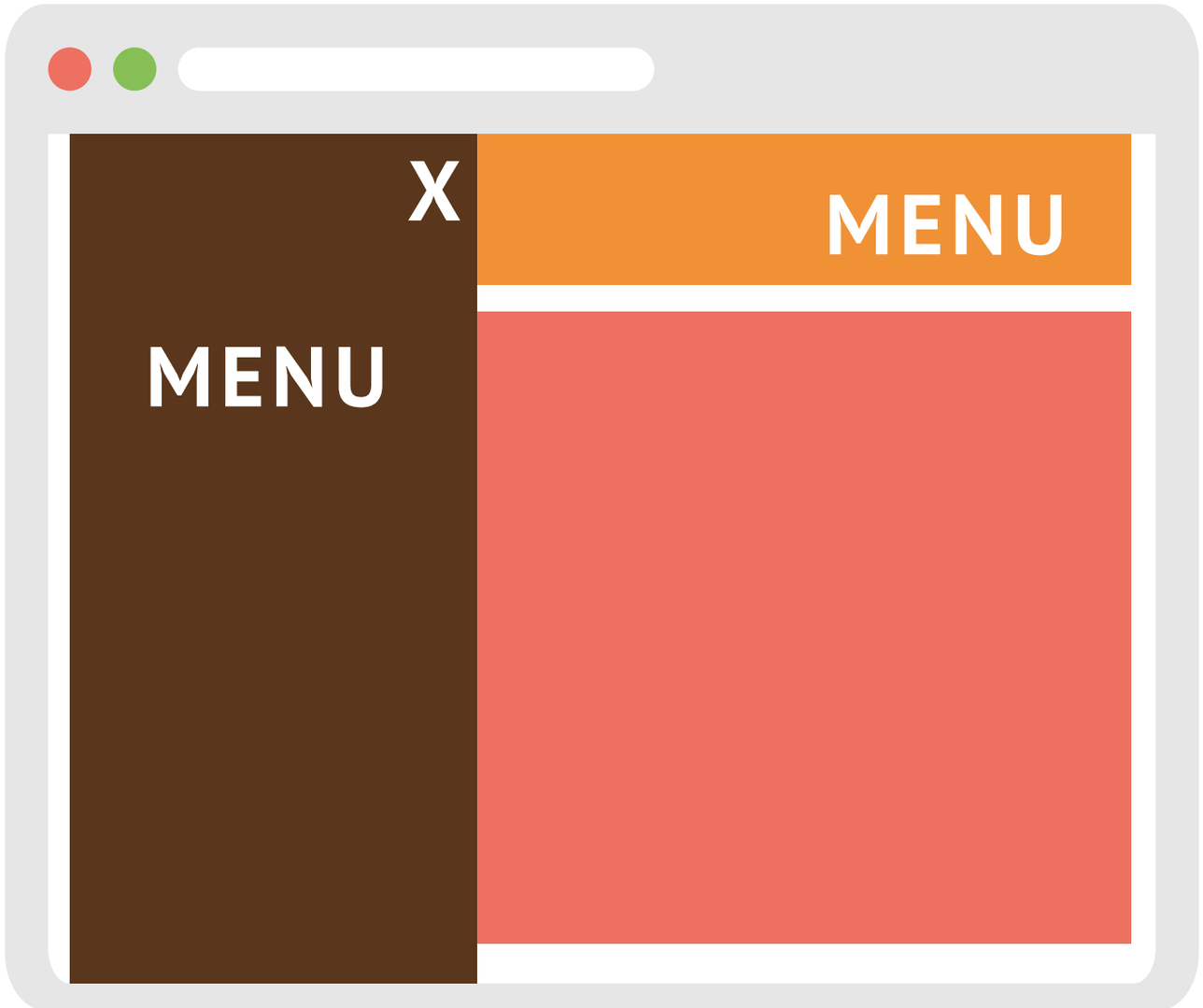
WITH HAMBURGER ICON AND SIDENAV



CARABAS HANDS-ON TRAINING
PRINT·WEB·OFFICE

MOCKUP'S

Wat gaan we maken?



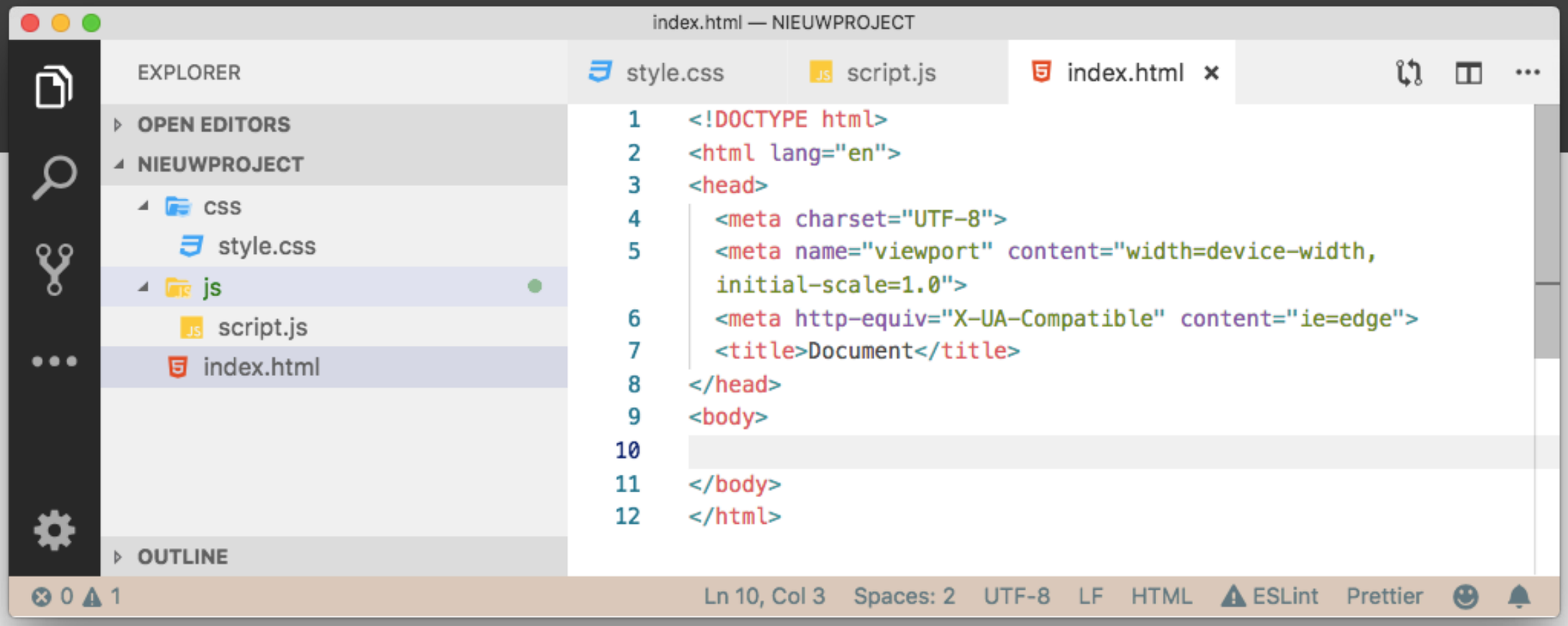
Een navigatie aan de bovenkant van de pagina.

Als het scherm te klein wordt, tonen we een hamburger icoon.

Als er op het icoontje geklikt wordt, verschijnt er een overlay met navigatie.

PROJECT SETUP

basic folder structure



We maken een simpel basis structuur; index.html
en een map 'css' met daarin 'style.css'
en een map 'js' en daarin 'script.js'.

HTML

koppelen van css & javascript

```
    <title>Document</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>

    <script src="js/script.js"></script>
</body>
</html>
```

Let op dat je ‘link’ gebruikt om de stylesheet te koppelen.
Onderaan voor de closing tag van de </body> schrijf je een script-tag.

HTML

de <header>

index.html — NIEUWPROJECT

EXPLORER

OPEN EDITORS

NIEUWPROJECT

- css
  - style.css
- js
  - script.js
- index.html

OUTLINE

style.css | script.js | index.html ×

```
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <header>
    <div class="hamburger">
      <a href="#" class="hamburger-btn">&#9776;</a>
    </div><!-- end of hamburger -->
    <nav>
      <ul>
        <li>Home</li>
        <li>Contact</li>
        <li>Blog</li>
        <li>Call Me</li>
      </ul>
    </nav>
  </header><!-- end of header -->

<script src="js/script.js"></script>
```

Niets speciaals: <header> met daarin een en een <nav>

HTML

het <main> content & de .hero

```
<!-- end of header -->

<div class="hero"></div>

<main>
  <h1>Yummy</h1>
  <p>Idea from: <a href="https://www.youtube.com/watch?v=o_WI2-wSudc">this movie</a></p>
</main>
```

Achter de closing-tag van de </header> voegen we de daadwerkelijke inhoud van de pagina toe: een lege , dat wordt de grote afbeelding op de pagina.

Vervolgens de <main> met daarin gewoon wat dummy inhoud: een H1 en een <p>, maar dit mag uiteraard vanalles zijn.

HTML

Het mobile menu, zichtbaar na click op icon

```
39    <!-- mobile-menu -->
40    <div class="mobile-menu">
41      <ul>
42        <li>Home</li>
43        <li>Contact</li>
44        <li>Blog</li>
45        <li>Call Me</li>
46      </ul>
47    </div>
48
49    <script  src="js/index.js"></script>
50    </body>
51    </html>
```

Onder de <main> schrijven we het menu dat nu uiteraard zichtbaar is, maar wat we later verbergen met CSS.

Een div met een classe, en een een <ul>.
Dat is alle HTML die we nodig hebben.

CSS

We openen ‘style.css’

We laden een lettertype van Goolge Fonts,

Een *-selector selecteert alle elementen op de pagina en zet de margi-
in en padding op 0.

Verder passen we op alle elementen het lettertype ‘Lato’ toe.

```
@import url('https://fonts.googleapis.com/css?family=Charmonman|Lato');

* { margin: 0; padding: 0; font-family: 'Lato';}
```

CSS

Het stylen van de header

De <header> krijgt een blauw/groene kleur, wat padding, een grotere letter,

en ‘display: flow-root’, die ervoor zorgt dat als we kind-elementen floaten, de hoogte van de <header> intact blijft.

We hebben dan geen ‘clearfix’-klasse mee nodig.

Het <a> element met de klasse hamburger-btn, floaten we naar links, en geven we wat basis stijling.

De <ul> met de navigatie, floaten we naar rechts. Van de <ul> maken we een flex-container, waardoor de <li> items automatisch flex-kinderen worden en dus naast elkaar komen te staan.

```css
header {
  background: #61D2D6;
  padding: 2em;
  font-size: 26px;
  color: white;
  display: flow-root;
}

a.hamburger-btn {
  float: left;
  color: white;
  text-decoration: none;
  margin-left: 10px;
}

header ul {
  float: right;
  list-style: none;
  display: flex;
}
header ul li {
  margin-right: 2em;
}

header li a{
  color: white;
  text-decoration: none;
}
```

CSS

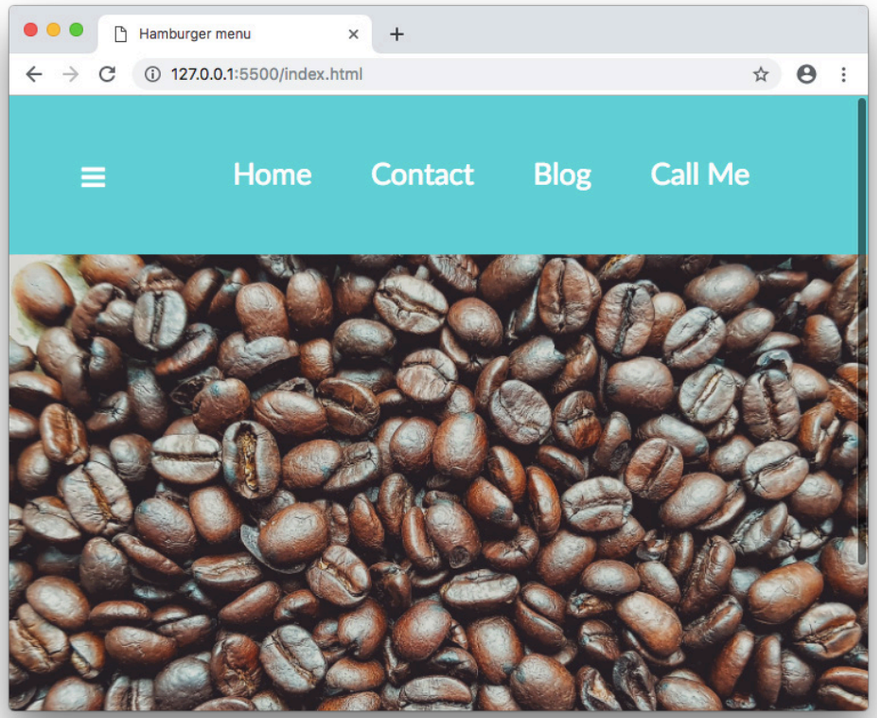
Het stylen van de hero

```css
/* Hero section */
.hero {
  background-image: url("https://source.unsplash.com/1600x900/?food");
  background-position: center;
  background-size: cover;
  height: 80vh;
}
```

Een simpele afbeelding met als thema ‘food’.

Een minimale hoogte van 80vh.

Uitlijnenen van de foto.



CSS

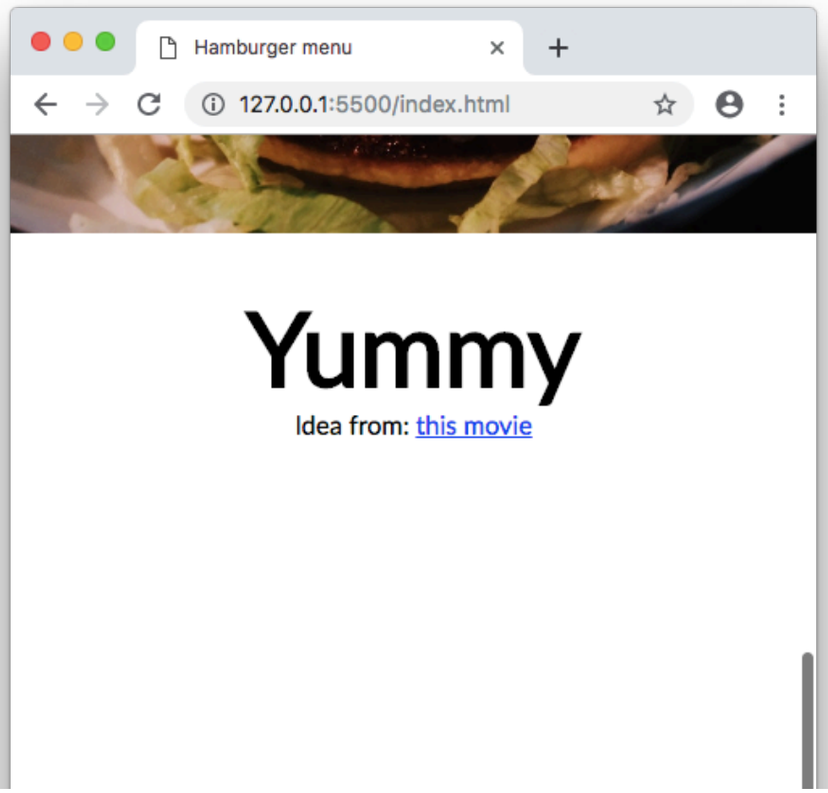
Het stylen van de main area.

In de main creeren we aan de bovenkant een beetje witruimte.

De hoogte wordt ingesteld op 90% van de hoogte van het scherm.

Display: flex & align-items: center zet alles in het midden van de pagina. Door Flex-direction: column staat alles op elkaar.

Een beetje styling op de kleur en tekstgrootte maakt het af.




```
main {
  padding-top: 2em;
  height: 90vh;
  background: white;
  display: flex;
  align-items: center;
  flex-direction: column;
}
main h1 {
  color: black;
  font-size: 4em;
}
```

CSS

Het stylen van het off-screen menu

Mobile Menu:
Position: Fixed om hem uit de flow te nemen, en vast te plakken aan het scherm.

De width: 80vw, zet de breedte altijd op 80% van het scherm, hoe groot of hoe klein het ook is. De hoogte is altijd 100% van het scherm.

Van het hele menu zelf, maken we een flex-container, en met justify-content, en align-items wordt alles naar het midden van de container gezet.

De transition-property zorgt ervoor dat als we de waarde 'tranform' wijzigen, dat deze wijzigingen met een animatie gebeuren.

Het menu wordt met 'translateX' volledig buiten beeld gezet. We willen het menu immers pas tonen als er op de hamburger-knop wordt geduwd.

```css
.mobile-menu {
  position: fixed;
  background: rgba(0, 0, 0, 0.8);
  width: 80vw;
  top: 0;
  height: 100vh;
  color: white;
  display: flex;
  justify-content: center;
  align-items: center;
  transition: transform 500ms ease;
  transform: translateX(-100%);
}
.mobile-menu ul {
  list-style: none;
}
.mobile-menu ul li {
  font-size: 2em;
  line-height: 2em;
}

.mobile-menu a{
  color: white;
}
```

CSS

Het stylen van het off-screen menu

Mobile Menu:
Position: Fixed om hem uit de flow te nemen, en vast te plakken aan het scherm.

De width: 80vw, zet de breedte altijd op 80% van het scherm, hoe groot of hoe klein het ook is. De hoogte is altijd 100% van het scherm.

Van het hele menu zelf, maken we een flex-container, en met justify-content, en align-items wordt alles naar het midden van de container gezet.

De transition-property zorgt ervoor dat als we de waarde 'tranform' wijzigen, dat deze wijzigingen met een animatie gebeuren.

Het menu wordt met 'translateX' volledig buiten beeld gezet. We willen het menu immers pas tonen als er op de hamburger-knop wordt geduwd.

```css
.mobile-menu {
  position: fixed;
  background: rgba(0, 0, 0, 0.8);
  width: 80vw;
  top: 0;
  height: 100vh;
  color: white;
  display: flex;
  justify-content: center;
  align-items: center;
  transition: transform 500ms ease;
  transform: translateX(-100%);
}
.mobile-menu ul {
  list-style: none;
}
.mobile-menu ul li {
  font-size: 2em;
  line-height: 2em;
}
```

CSS

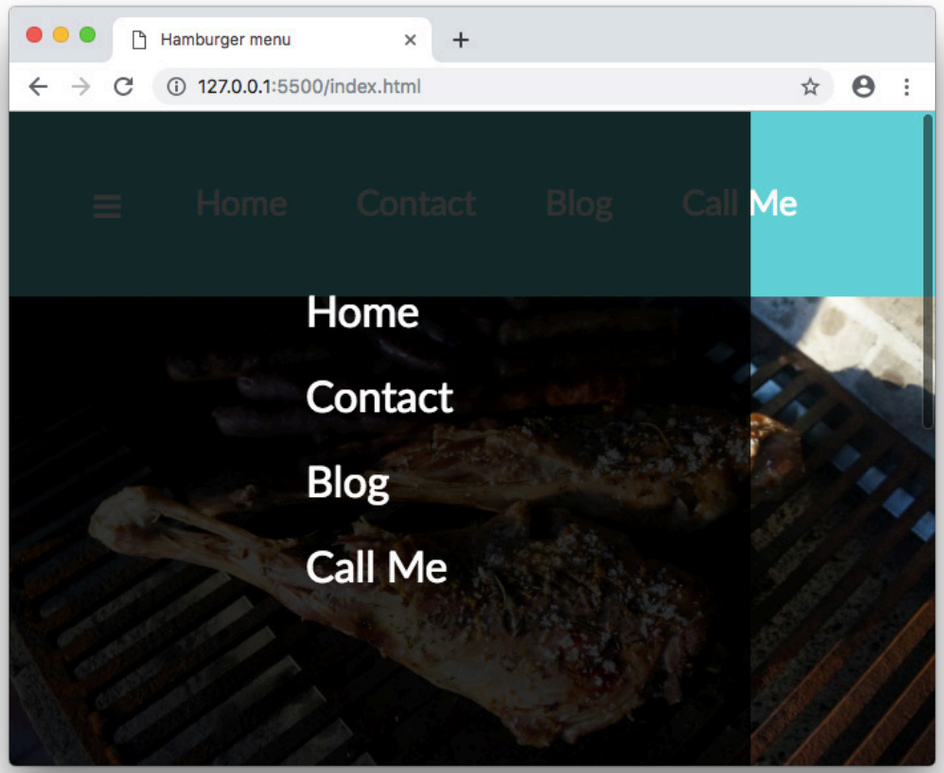
De klasse ‘open’

Als we nu handmatig een klasse ‘open’ toevoegen op het mobile-menu, springt het open.

Bij het weghalen van de klasse open, springt het menu dicht.

Exact de functionaliteit die we nodig hebben.

Let op, uiteraard schijf je de ‘open’ niet de html, maar in je style.css.




```
<!-- mobile-menu -->
<div class="mobile-menu open">
  <ul>
    <li>Home</li>
    <li>Contact</li>
    <li>Blog</li>
    <li>Call Me</li>
  </ul>
</div>

<script  src="js/index.js"></script>
</body>
</html>



<style>
.open {
    transform: translateX(0);
}
</style>
```

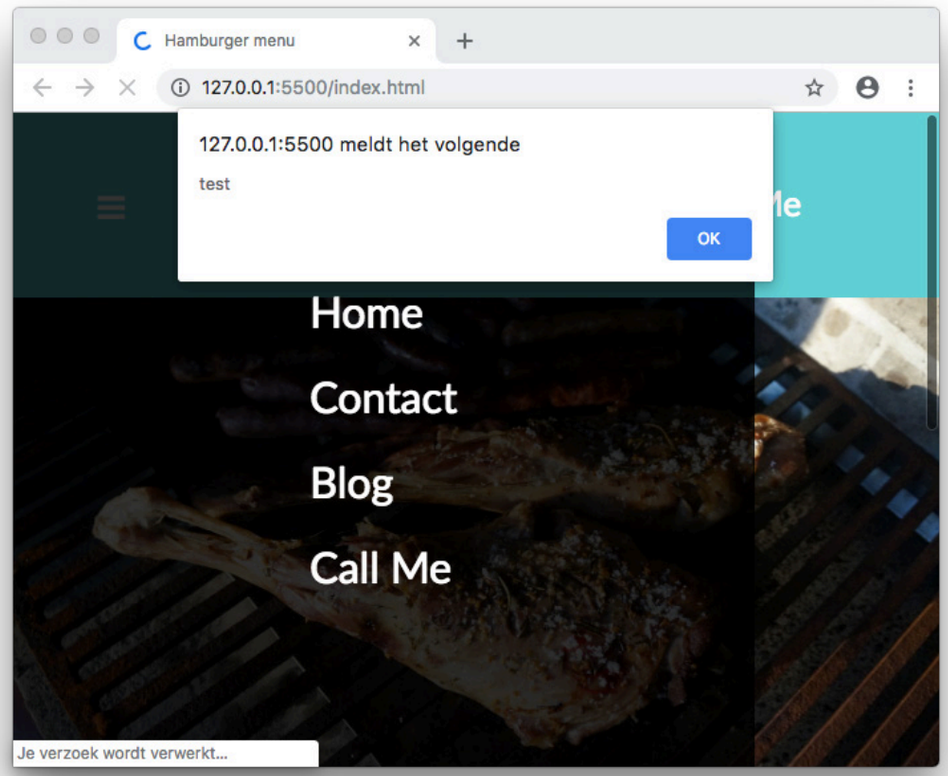
SCRIPT.JS

interactiviteit: javascript

Open nu de script.js en om te testen of alles werkt schrijf ik al eens dit in de code: ‘alert(‘test’);

Na een refresh geeft dit een popup in de browser.

```
alert('test');
```



SCRIPT.JS

2 variabelen.

```
// alert('test');

var hamburgerButton = document.querySelector('.hamburger-btn');
var mobileNav = document.querySelector('.mobile-menu');
```

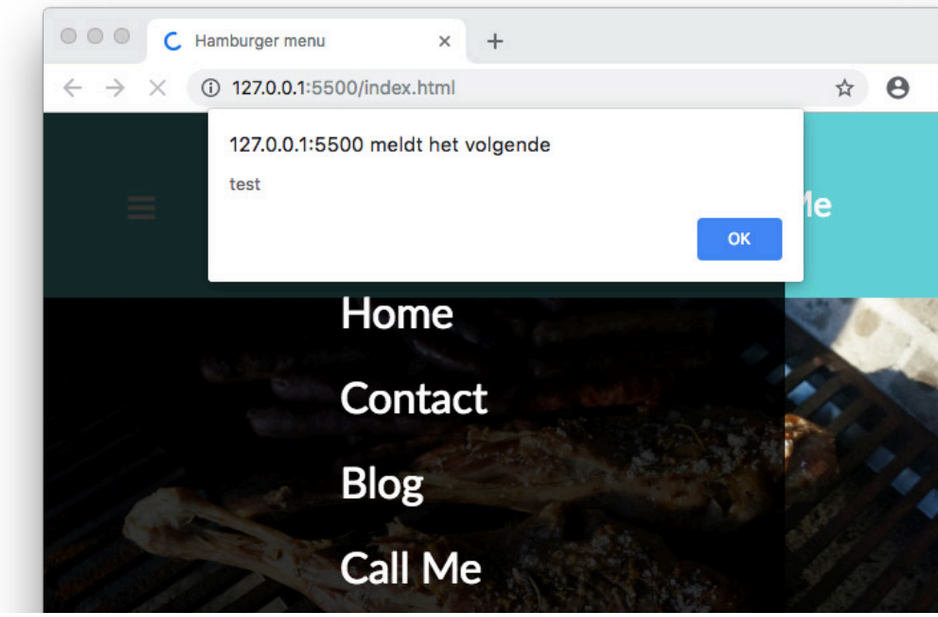
Met het keyword ‘var’ kunnen we zelf variabelen maken om teksten in op te slaan. Beschouw het als een multifunctioneel tupperware doosje.

Met de method document.querySelector(“.example”); kan je je eigen HTML doorlopen, en het eerste element met de opgegeven class opslaan.

In dit geval ideaal, omdat we een aantal keren het hamburger-menu icoontje nodig hebben.

Hetzelfde geldt natuurlijk voor de zijbalk, we slaan deze dus zo op.
(Zie voorbeeld hierboven.)

Meer info over QuerySelector:
https://www.w3schools.com/jsref/met_document_queryselector.asp



SCRIPT.JS

Event-Listener

```
var hamburgerButton = document.querySelector('.hamburger-btn');
var mobileNav = document.querySelector('.mobile-menu');

hamburgerButton.addEventListener('click', openMobile);
mobileNav.addEventListener('click', closeMobile);
```

Met de ingebouwde ‘addEventListener’-method kan je ‘events’ registreren in de browser. Een Event is een actie die plaatsvind in de browser, waarop je reacties kan koppelen.

Voorbeelden van Events; document.load, click, dubbelklik, focus, etc.
Meer info: https://www.w3schools.com/js/js_events.asp

In ons geval koppelen we een event aan het hamburger-icon in de header. En we willen daaraan een functie koppelen ‘menuOpen’

SCRIPT.JS

De open- en close function

```
var hamburgerButton = document.querySelector('.hamburger-btn');
var mobileNav = document.querySelector('.mobile-menu');

hamburgerButton.addEventListener('click', openMobile);
mobileNav.addEventListener('click', closeMobile);

function openMobile(){
  mobileNav.classList.add('open');
}

function closeMobile(){
  mobileNav.classList.remove('open');
}
```

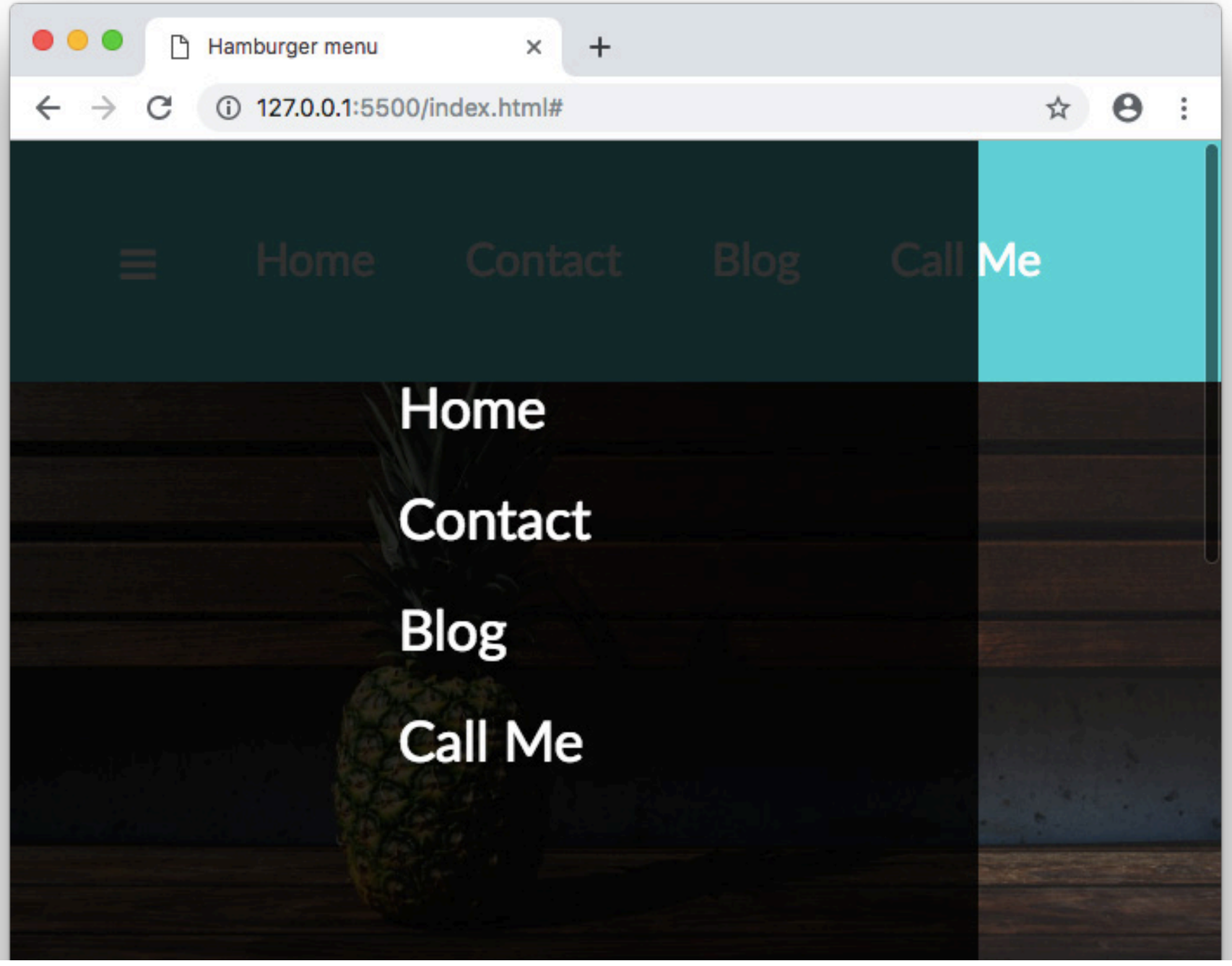
Met functions kan je meerdere handelingen die vaak nodig hebt, bundelen in een simpel aan te roepen functie.

In ons geval: een open-functie en een close functie.

Wat moet deze doen? Simpel weg een classe toevoegen en weghalen op de het menu.

De .classList method kan klassen toevoegen en verwijdernen van elementen.

DONE


Hamburger menu
127.0.0.1:5500/index.html#
Home
Contact
Blog
Call Me
Home
Contact
Blog
Call Me