Aman Patel
Chaya Chrein
Jonathan Yulan

Project 2: Tsavo National Park

Project Links:

- Github Repository including all files
- Github EJS Files
- Wireframes
- Video Demonstration

Business Model and Website Design:

- What is your business?

  Tsavo park is one of the oldest parks in Kenya. Here you can explore much of the nature native to Africa. With our varying safari options and nature areas you can gain knowledge about the African wilderness through seeing it first hand.
- What is the business ethos?

  We are Targeted. Unbiased. Direct. We seek to give all people a pleasant and enjoyable experience while also teaching them about Africa.
- What do you produce/make/offer/sell?

  We have a variety of guided tours and safaris from which people can choose, allowing people to see animals, wildlife, and habitats that interest them.
- Who is your target audience?
    - *Those seeking to learn more about wildlife in Africa*
    - *Those seeking a safari day trip*
    - *Those seeking to enjoy a beautiful (while also educational) area with astounding scenery, nature, and animals*
- What are the key motivations of why you expect people to visit your website and how you will attract and motivate people therein?

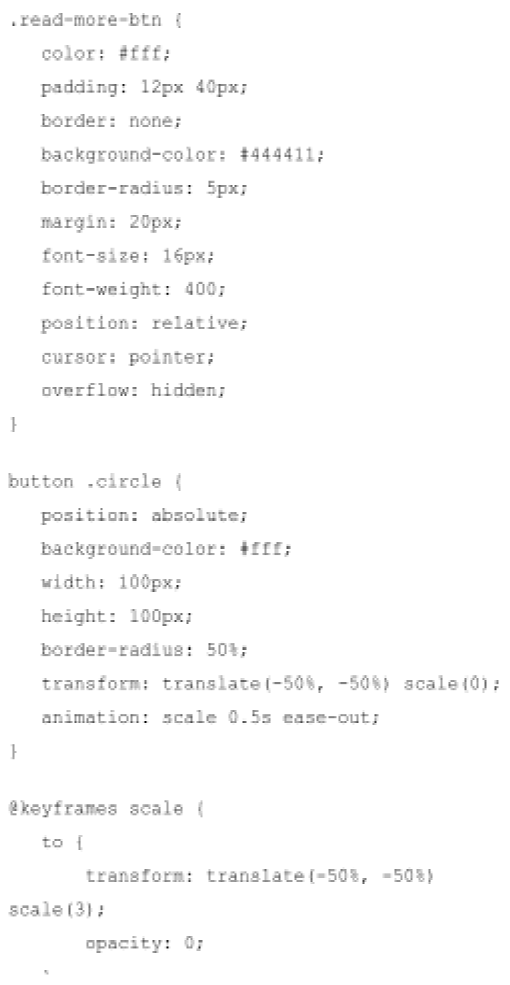
  People will visit our website to find details about our location, what we offer, and general information about what services we provide. To keep visitors engaged, we have a coherent design with natural colors (like green and brown tones) as well as astounding photos which will perhaps motivate people to come visit. We also make our website easy to navigate so people can find whatever information they are looking for and they can contact us in case they are seeking more information.
- Do they want general information / research, or are they after something specific? Are they already familiar with the service or product that you offer or do they need to be introduced to it?

  People visiting our site are seeking information about what we stand for and how they can come visit Tsavo Park. We therefore provide details about our safaris and tours, our location, and a means to contact us in case they have further questions.

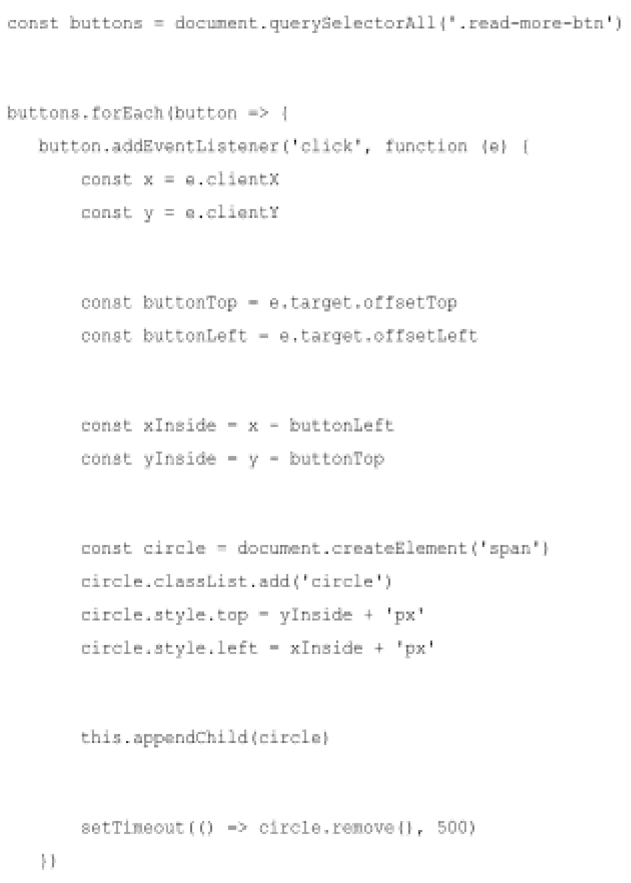
- Do they need to contact you? If so, can they visit in person (which might require opening hours and a map)? Or might they need email or telephone contact details.
  Should someone need to contact us, we have contact details in the footer of every page and we also have a contact us page where they can message us directly.
- Will visitors be familiar with your subject area / brand or do you need to introduce yourself? Will they be familiar with the product / service / information you are covering or do they need background information on it?
  People visiting our site will know general information regarding the services we offer, such as safaris and tours, but will get more information about our specific safaris from the website.
- What are the most important features of what you are offering? What is special about what you offer that differentiates you from other sites that offer something similar?
  We are proud to provide a safari that educates people about the beauty of Africa and we try to give that over to our visitors. In addition to the breathtaking views of our area and the wildlife, we hope that people have a better standing about Africa and understand its beauty as well.

Salient Points of Code:

1. Ripple Button Effect:



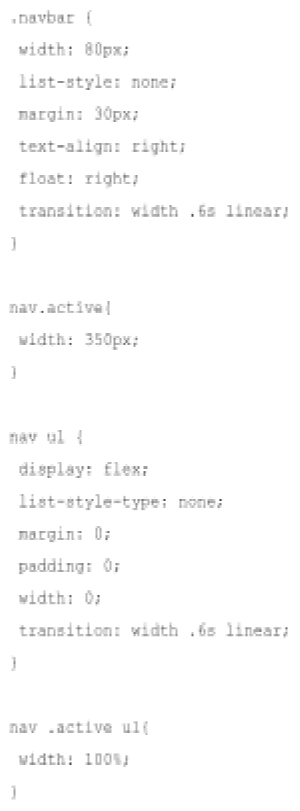
```
.read-more-btn {
   color: #fff;
   padding: 12px 40px;
   border: none;
   background-color: #444411;
   border-radius: 5px;
   margin: 20px;
   font-size: 16px;
   font-weight: 400;
   position: relative;
   cursor: pointer;
   overflow: hidden;
}

button .circle {
   position: absolute;
   background-color: #fff;
   width: 100px;
   height: 100px;
   border-radius: 50%;
   transform: translate(-50%, -50%) scale(0);
   animation: scale 0.5s ease-out;
}

@keyframes scale {
   to {
       transform: translate(-50%, -50%)
scale(3);
       opacity: 0;
```



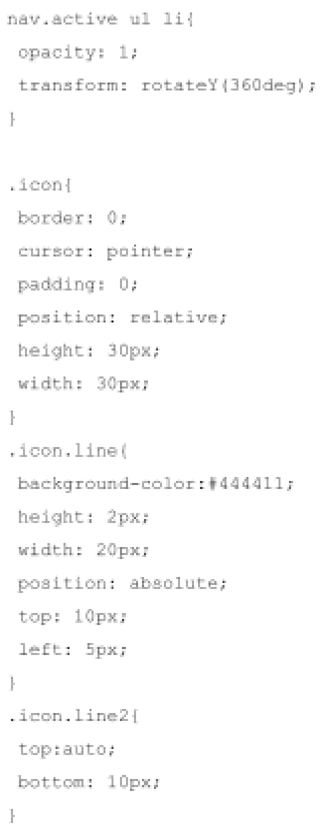
```
const buttons = document.querySelectorAll('.read-more-btn')


buttons.forEach(button => {
   button.addEventListener('click', function (e) {
       const x = e.clientX
       const y = e.clientY


       const buttonTop = e.target.offsetTop
       const buttonLeft = e.target.offsetLeft


       const xInside = x - buttonLeft
       const yInside = y - buttonTop


       const circle = document.createElement('span')
       circle.classList.add('circle')
       circle.style.top = yInside + 'px'
       circle.style.left = xInside + 'px'


       this.appendChild(circle)


       setTimeout(() => circle.remove(), 500)
   })
```

a. Circle (css):
    i. The circle is made to be white with a set width and height
    ii. It's made to translate by 50% on the x and y axis so it's centered
    iii. We create a keyframes animation to scale the size up and fade out
b. In the JS, we create an event listener for each button when it's clicked:
    i. Collect the x and y coordinates of the button and then use offset to get where we clicked inside the button
    ii. Create a span element and add the circle to it with a top and left location so that our circle created in the css appears
    iii. And we use setTimeout so the circle not only fades out, but also is removed from the dom.
c. Struggles encountered:
    i. When implementing this udemy project, I had a hard time getting it to work. When I tried implementing it on a button individually it worked, however when I added elements above it, it stopped working. I tried fixing it by changing code to create test cases and see if the circle just appeared without the ripple effect anywhere on the page but that didn't help. I ended up just rearranging my layout so that the elements with the buttons were above the other elements.

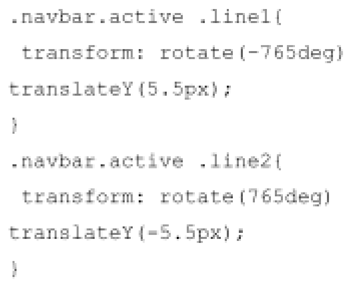
2. Animated Navigation Bar:



```
.navbar {
 width: 80px;
 list-style: none;
 margin: 30px;
 text-align: right;
 float: right;
 transition: width .6s linear;
}

nav.active{
 width: 350px;
}

nav ul {
 display: flex;
 list-style-type: none;
 margin: 0;
 padding: 0;
 width: 0;
 transition: width .6s linear;
}

nav .active ul{
 width: 100%;
}
```



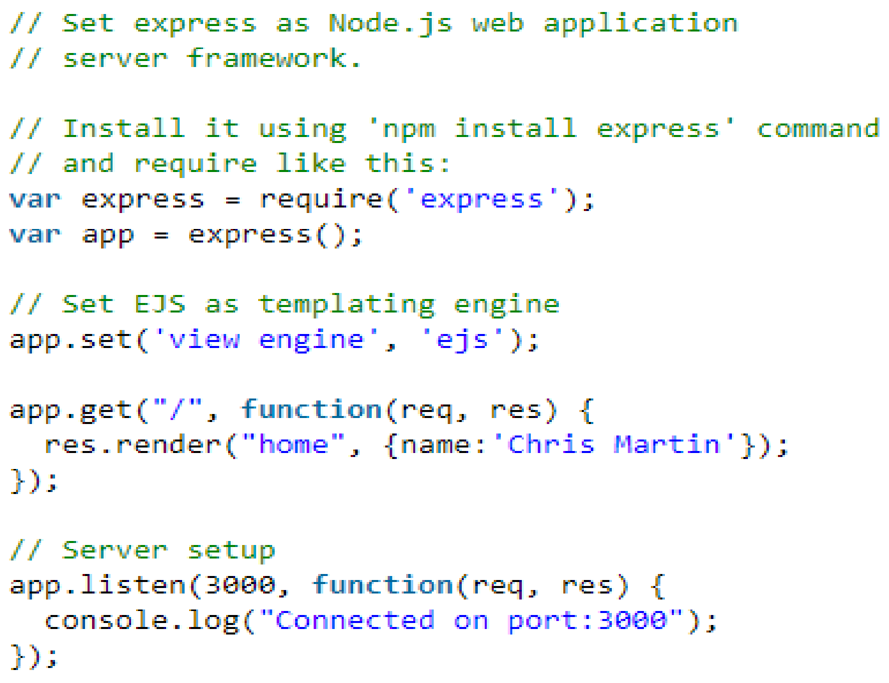
```
nav.active ul li{
 opacity: 1;
 transform: rotateY(360deg);
}

.icon{
 border: 0;
 cursor: pointer;
 padding: 0;
 position: relative;
 height: 30px;
 width: 30px;
}
.icon.line{
 background-color:#444411;
 height: 2px;
 width: 20px;
 position: absolute;
 top: 10px;
 left: 5px;
}
.icon.line2{
 top:auto;
 bottom: 10px;
}
```



```
.navbar.active .line1{
 transform: rotate(-765deg)
translateY(5.5px);
}
.navbar.active .line2{
 transform: rotate(765deg)
translateY(-5.5px);
}
```

    a. Most of the code for this is done in css. The JS is simply adding the active class when the icon is clicked.
    b. List items in the navbar:
        i. Have rotational transitions
        ii. Opacity of 0 when not active
        iii. Opacity of 1 when active
    c. Lines that from the x or the stacked hamburger:
        i. Have rotational transitions to switch from x to = based on the active class of the navbar
    d. Struggles encountered:
        i. I found this udemy project to be a more difficult one to implement as although I didn't alter much css and really only added to it, it reformatted some elements. I also found the rotation from the x to = a bit complex. Overall, I implemented it as best as I could while touching up the design.

3. Embedded JavaScript Templates(EJS)
    a. Embedded Javascript is part of the MVC model, and its primary function is the view component of the MCV model. As a new technology that our team had to learn, there were some nuances required in order to ensure that the rendering of our webpages was done properly. An issue that we ran into was understanding how important our website's file structure was used in conjunction with our static HTML and CSS files. We had to ensure our file structure was created in such a way that the EJS handler was able to render the static parts of our webpages, as well as the dynamic parts properly. The EJS view handler allowed us to easily implement the dynamic aspects of our website, utilizing the Udemy lessons specifically when implementing javascript in the .ejs files of our website. The snipe below showcases a basic setup in order to use the EJS template viewer, which was implemented in our design.



```
// Set express as Node.js web application
// server framework.

// Install it using 'npm install express' command
// and require like this:
var express = require('express');
var app = express();

// Set EJS as templating engine
app.set('view engine', 'ejs');

app.get("/", function(req, res) {
  res.render("home", {name:'Chris Martin'});
});

// Server setup
app.listen(3000, function(req, res) {
  console.log("Connected on port:3000");
});
```

4. Node | Express
    a. Continuing off the third salient point of our website; we utilized both Express and Node in order to handle the server-side rendering of our website. Our website was rendered and hosted locally via Express and Node. Node and Express in conjunction with our node-modules that we downloaded in order to ensure that our website worked as intended. Node is an open-source, easily implemented, cross-platform runtime environment that allows us to render and host our website locally on our localhost port of 5000. Node's powerful runtime engine allowed us to not only server our static HTML and CSS files, but also allowed us to use browser-specific Javascript in order to allow for a more dynamic and aesthetic website. Being that this was the first time we had utilized Node and Express for web development, there was a learning curve as our engineers spent time researching and reading through documentation and implementing our design into our website. Node and express was the technology we used to build our localhost server, and we used it to route and handle all of our websites pages. This was implemented in our overall website design but also allowed us to handle mis-routing of web pages as well. Weirdly enough, one of the more difficult aspects of learning how to use node and express was starting and debugging the server when we ran into issues. Once we learned the proper nuances pertaining to starting the server, and how to utilize the CLI to start, stop (node server.js , npm install -- save, nodemon server.js etc...) and download the Node appropriate nodules and libraries it became intuitive and became quite easy to run.

5. DB | Heroku
    a. This aspect of our website was the most difficult to implement for our engineers and required multiple different sources of technology to ensure it worked as intended. We used MYSQL as the database language, MYSQL workbench in order to view, and query the database, Heroku for hosting the database and implemented this with express, and node was daunting. Our team of engineers relied on a multitude of sources in order to implement the design as it was intended. The easiest aspect of this portion of this position of the website was setting up the form to have the express server post to the backend. The most difficult part was actually creating the database and utilizing software to create the backend database. We implemented this into our website by utilizing our HTML form to collect the inquiries of our potential customers, in order to obtain feedback and to build the website. We were able to connect the form aspect of our website with our backend quite easily once we understood express, mySQL, post routing and node. This aspect of our website is fully functional, and can be expanded and configured for future more advanced implementations.