Project Part 1: Setup and Client-side Buildout
Due: March 12th, 2020
150 Points

Purpose:
We are going to start building out a front end framework that will be grown throughout the semester as we evolve our projects. You may decide to build your own game or application or you may choose to follow along with the Oregon Trail example application. If you choose to create your own game or application, the **size and scope of the effort must be equivalent to the Oregon Trail project**! You may use the detailed requirements of the Oregon Trail project as a guideline for custom project complexity.

-- Oregon Trail Project background --

You may base your project on an old game called “Oregon Trail” that was released in the 70’s. Here is a link to play the game online: https://classicreload.com/oregon-trail.html, remember you may also create a project of your own design.

This first lab will get our initial client side pieces in place so we can grow it as the semester goes on. We will create a menu, a top ten score list, some game music and stub out many of the parts that we will expand on in future labs and projects.

In addition we will create our back end server infrastructure and learn how to manage routes (the server returning the correct front end elements when requested)

Preparatory:
You will need everything working from our previous labs. In short, at a minimum, you will need to be able to do the following:

- Use git to manage your project, and be setup in the MaristGormanly organization.
- Make sure you have your local development environment setup to your liking.

Submission:
Create a **release (tag)** of your project when it is ready in github and submit the iLearn lab assignment with a link to your git repository release.

Major Pieces:

1. Create project and git repository
2. Create Node server for project with express / build routes
3. Create client side project files including:
    a. HTML views

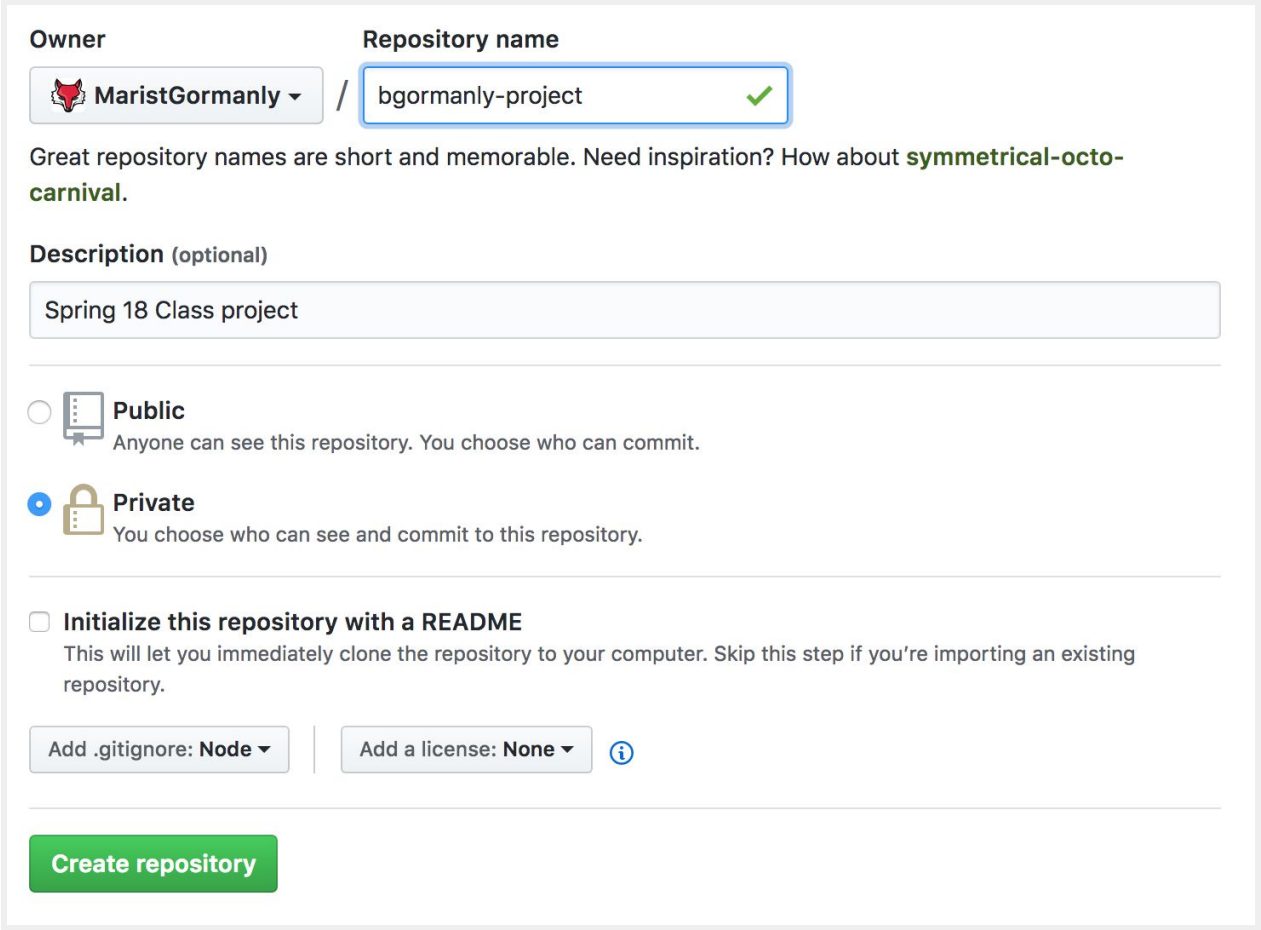
    b. CSS stylesheets
    c. JavaScript
4. Build out HTML views and CSS
5. Base client side functionality

All of these represent an investment of significant time to achieve. It is **99.999%** likely that you will not finish the project on time at any decent level of quality if you wait until the final week to do it. **This project requires that you pace yourself** to be successful. (As do all the software projects that your future might have in store for you).

Part 1: Create your project and git repo

Create a git repository for project in github (you should name your project first initial-lastname-project **Example**: bgormanly-project, and it **must** be created within the MaristGormanly organization, not your own github public space) and clone it down to your local development machine in the direction of your choosing.

Here is a screen shot to help: ensure that your repository is marked private and you also should add a .gitignore file for 'Node'.



Alternatively, you may create a get repository right on your laptop and then link it to a remote on github using ‘git remote add …’ (see git slides and previous lab on using git)

Once you have the local working copy of the project on your development machine and it is connected to a remote github repository, create your node project using npm init. You should set the “entry point” to “oregonTrail.js” when it asks or by editing the package.json file. Once the package.json file is to your liking, commit and push it up to github. Verify that it is there.

Part 2: Create and test your Server (Node w/ express)

Now we are ready to code. Since this is just your project, it is perfectly okay to work directly on the master branch, however, if you decide it would be better to create a branch to work on to try something out, feel free to do so.

We will start with getting a basic Node Express server running. The express module is the one that we discussed in class that handles much of the basic web server functionality for us so we can concentrate on building our specific application.

First create a folder in your project home folder called “server” and one called “client”. In the server folder create a file called “oregonTrail.js”. This will be your main node file for the project where you will put all the code for the express part that follows:

In your oregonTrail.js file paste the following code:

```
const express = require('express')
const app = express()
const port = 1337

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

To run this code you need to have the express web server dependency installed. To do this go back to the home folder of your project (back out of the server folder) and run

```
npm install express
```

This will install the dependency in a folder it creates called node_modules/

```
 ~/projects/courses/cmpt221/s20-project  >   npm install express
```

```
npm WARN saveError ENOENT: no such file or directory, open
'/Users/brian/projects/courses/cmpt221/s20-project/package.json'
npm notice created a lockfile as package-lock.json. You should commit this
file.
npm WARN enoent ENOENT: no such file or directory, open
'/Users/brian/projects/courses/cmpt221/s20-project/package.json'
npm WARN s20-project No description
npm WARN s20-project No repository field.
npm WARN s20-project No README data
npm WARN s20-project No license field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 126 packages in 1.607s
found 0 vulnerabilities
```

From your project home directory you should be able to start your server with:

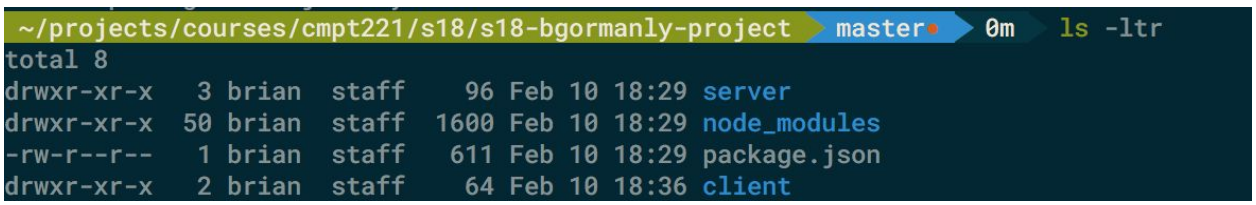
```
 ~/projects/courses/cmpt221/s20-project > node server/oregonTrail.js
Example app listening on port 1337!
```

Once you get your Server working, you should be able to go to http://localhost:1337 in your browser and see “Hello World!”

Commit and push your code up to github! (as you should at a minimum everytime you complete a part, but feel free to commit / push more as you see fit).

Part 3: Create client side structure and files

In the project home directory, create another folder called “client”. You should now have something that looks similar to this when you display the contents of your project home folder
Note: my project folder name changed from the part 2 examples from s20-project to s18/s18-bgormanly-project. This is just because I created the examples using 2 different folders. yours will have not changed!:

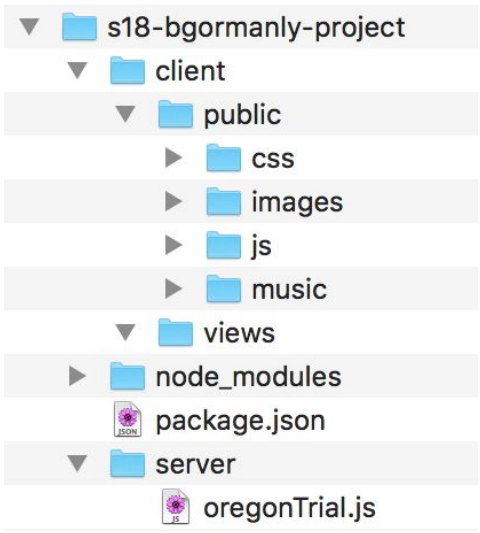


In your client folder create the following:
A folder called “views” for your html views

A folder called “public” that will contain all public resources (more on this soon)
Now, Inside the public folder, create the following folders:

- A folder called “css” for your stylesheets
- A folder called “js” that will have your client side javascript
- A folder called “images” for your images
- A folder called “music” for your sound

You should have a directory structure similar to the following when done:



Once you have created all the directories needed we are going to setup your node server so that it can find and respond with the correct files when requested.

The first step is to create an html file so we can test, create one called index.html in the client/views folder. Put some basic html in for now so you have something to look at when it is displayed, a simple Hello world! will be enough, along with the mandatory <html> and <body> tags of course. Additionally put 1 image in the client/public/images folder. Add an <img> tag to your index.html file to test loading it. Your tag should link to the image this way: .

Now we are going to setup node for 2 things:

1. We are going to tell node that anything in the client/public folder is automatically accessible from the sites root path url. For example, if you have a image in the file system called image.png and the path from your project root folder is client/public/images/image.png you will be able to see the image by going to http://localhost:1337/images/image.png. To do this add the following to your oregonTrail.js node file after the const app = express(); line:

```
app.use(express.static('client/public'));
```

Once you have added the line, startup your node server and try going to your image url like the example above changing the name of the image for your own (http://localhost:1337/images/image.png) you should see your image!

2. We are now going to set up a route to tell the Node server what file to return when specific requests come in for pages. We are going to start with our index.html example. In your oregonTrail.js file replace your

```
app.get('/', (req, res) => res.send('Hello World!'));
```

With

```
app.get('/', function (req, res) {
  res.sendFile('index.html', {root: './client/views' })
})
```

Let's look at what this code does. The '/' which is the first parameter passed into the get function (remember app is your instantiated express application server) is the url that the server will be 'listening' for if you go to http://localhost:1337 it is the same as if you had gone to http://localhost:1337/ with a trailing slash. The slash indicates the 'root' of the website, essentially indicating you did not ask for a specific page. Now we are saying if you ask for this url, use the callback function that we pass in as the second parameter. Notice that the function we pass as the callback is getting a file, in this case our index.html file (note that the directory you find it in is in the {root: … }). So what happens is that everytime you ask for the / url (no page, root url of the site) it returns the index.html file. If you wanted to see the index.html file only when the url explicitly said http://localhost:1337/index.html, you could do this instead (don't keep this one, it is just an example to help explain the point):

```
app.get('/index.html', function (req, res) {
  res.sendFile('index.html', {root: './client/views' })
})
```

Once you have setup the route, stop your node server if it is running and restart it. You should be able to get your index.html file (using the route in the first parameter of the get method) and also see the image on the page since we have all resources in the public folder available without explicitly creating routes for them. If you can see your index.html content and the image on it when you got to the http://localhost:1337 url then you are ready to move on to the next part.

Part 4:

Create the following files in the appropriate folders:

- You will need a total of 5 html views: index.html, mainmenu.html, setup.html, trail.html and topten.html all in the views folder. You should already have index.html there.
- You will need a total of 6 client side JavaScript files, 1 global.js and 1 for each of your views (index.js, mainmenu.js, topten.js, setup.js, trail.js)
    - Each of your views should have a link to your global.js and the corresponding JS file for that html view (game.html will have links to global.js and game.js).
- You will need a total of 6 css files, 1 global.css and 1 for each of your views (index.css, mainmenu.css, topten.css, setup.css, trail.css)
    - Each of your views should have a link to your global.css and the corresponding CSS file for that html view (setup.html will have links to global.js and setup.js).
- You will need some images and an mp3 file for game music
    - Make sure to search for royalty free images and music!

Note:::: You will have to create routes in oregonTrail.js for each of the .html files you created to use them! These will be just like the one we created earlier for / going to index.html.
Create routes for the following urls:

- /mainmenu -> mainmenu.html
- /topten -> topten.html
- /setup -> setup.html
- /trail -> trail.html

Now that the files and routes exist, build out your html files using the following requirements (you can add css styling at your own discretion) There are screen shots at the end of this document to help guide you.

1. index.html: This screen is a splash screen that you created earlier to test with. You can now change out the contents, it should have the following:
    a. Basic components
        i. A large image
        ii. A title
        iii. A message fading in and out that says “Press the spacebar to continue”
2. mainmenu.html
    a. Basic Components
        i. A title image
        ii. A menu consisting of the following options:
            1. Travel the trail
            2. Learn about the trail
            3. See the Oregon top 10
            4. Turn Sound (Off / On)

3. topten.html
    a. Basic Components
        i. Title image (can be the same as the one on the mainmenu.html page).
        ii. A top ten score list with fields including: Name, Score and Date
        iii. The ability to return to the main menu by pressing spacebar (or clicking the message)
4. setup.html
    a. Basic Components
        i. Title image (can be the same as the one on the mainmenu.html page).
        ii. A menu of items from the first game question:
            1. Be a banker from Boston
            2. Be a carpenter from Ohio
            3. Be a farmer from Illinois
            4. Find out the differences between the choices
        iii. Text to allow the user to press space bar to go back to the main menu.
5. trail.html
    a. This will be where we play the game, for now we are just going to create the major areas including:
        i. Title image (can be the same as the one on the mainmenu.html page).
        ii. An area for a background image showing terrain types and wagon progress. For now it should show 1 image as a placeholder
        iii. An area for game information to be displayed. While we do not have the information to populate yet, you can create the titles and areas for data once we have it using s and s. Titles should include:
            1. Days on trail
            2. Miles traveled
            3. Party health status
            4. Current weather
            5. Current pace
            6. Current Terrain
            7. # of party members alive

Again, make sure to commit and push to github!

Part 5: Implement required functionality

6. index.html / js:
    a. When the user presses the spacebar they should be brought to the main menu. They should also be able to click the “Press the spacebar to continue” message to get to the main menu
        i. The fade on the message should be smooth and use Javascript.
        ii. Any JavaScript for this page should be in index.js unless it is global.

7. mainmenu.html / js
    a. Use should be able to press the corresponding number to the menu choice for 1. Travel the trail, 3. See the Oregon top 10 and go to that page. This should be done with JavaScript! Not through html links!
    b. Choosing 4. Turn sound (On / Off) should start or stop the sound and the menu item should update to show the user what the current state of the music playback is.
8. topten.html / js
    a. The top ten list should be stored and displayed from a JavaScript Array that contains TopScore objects. A TopScore object should be able to store the full name, score and date of a top score.
    b. The top ten list should be displayed on the page and SORTED! The sorting should be so that the highest score is shown at the top and the lowest at the bottom. The scores must be added to the array initially out of order!
9. setup.html / js
    a. Just returning to the main menu via pressing the spacebar or clicking the “Press spacebar for the main menu” text.
    b. All other functionality will be built out at a later date for this screen.
10. trail.html / js
    a. No functional requirements yet, just html / css buildout.

General Requirements:

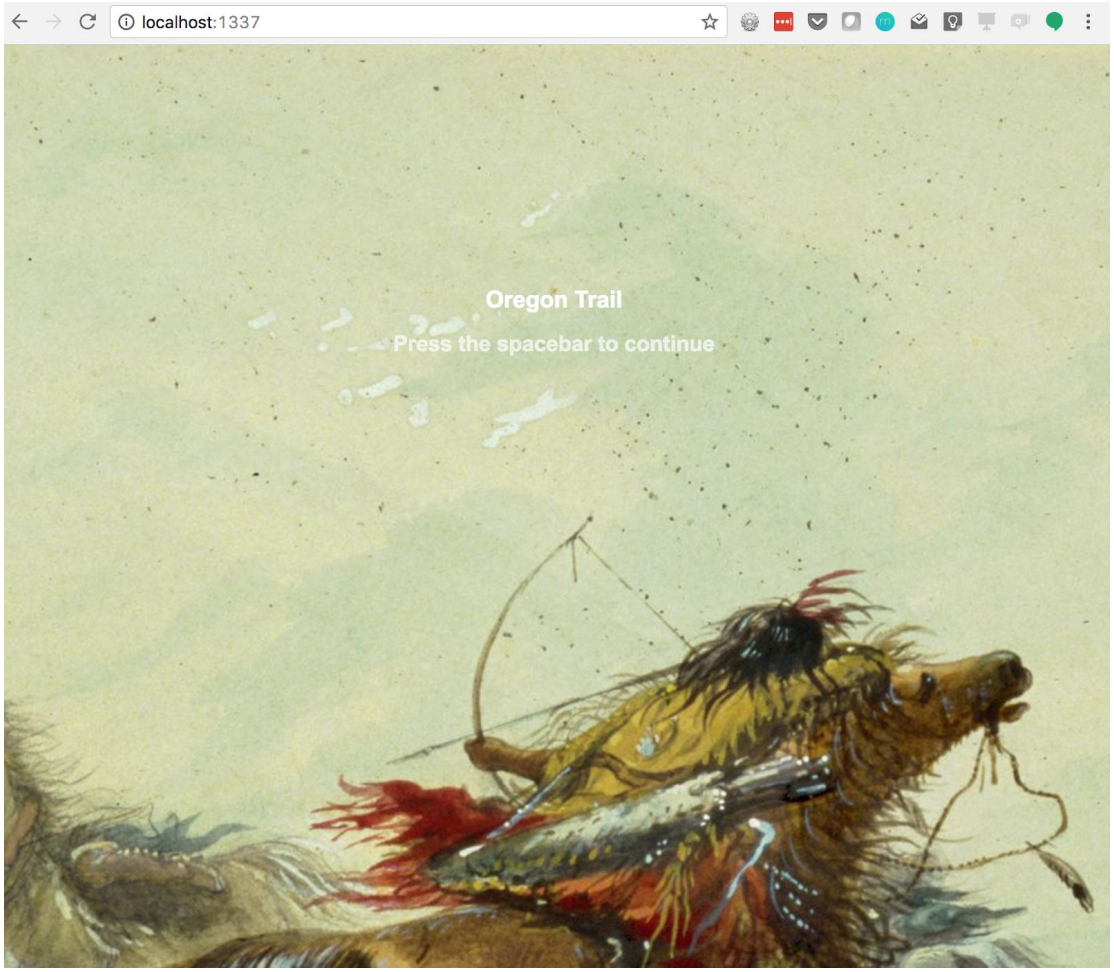
- All Content should have set resolution: (1160x720px or another of your choice, not smaller than 800x600). It should be centered horizontally in the browser on all screens
- All HTML should be completely semantic!!!
- CSS and JS that is unique to a page should be in that page's corresponding .css and .js files!
- You should find non-copyright protected images to use for your splash page, main menu and anywhere else you want. Hint: to help find non-copyrighted images use Google image search, go to search tools -> Usage Rights -> Labeled for Reuse.

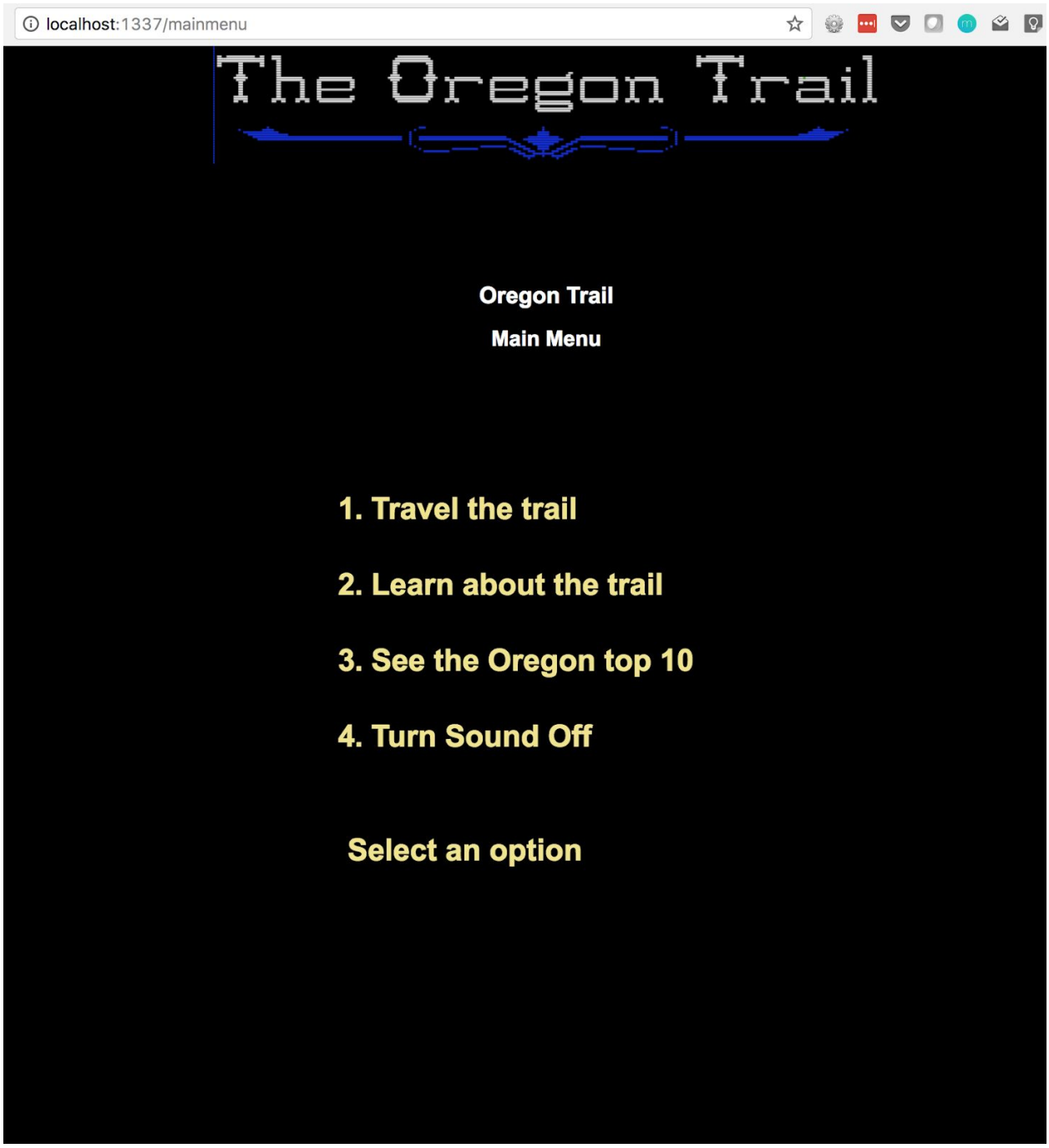
- Your designs should look and act the same across all major desktop browsers (chrome, ff, safari, opera, and yes, i.e.)
- The splash page will be the landing page (index.html) and should have graphics as you see fit and a “Press Spacebar to Continue” message on it that fades in and out. When the user presses the spacebar they should proceed to the main menu. Bonus: Add a clever way for tablet users to continue without the need for the virtual keyboard spacebar.

Grading breakdown:

- Creating the project with good folder structure, file names, etc
- Semantic HTML, nice clean code and proper use of .js and .css files
- Splash screen functionality and look
- Menu functionality and look
- All js works!
- Music works
- Overall look (fit and finish, graphics, fonts, etc)

Example Screenshots (these are meant as guidance / idea starter only! You should make your project look like your own!!!




localhost:1337/mainmenu
The Oregon Trail
Oregon Trail
Main Menu
1. Travel the trail
2. Learn about the trail
3. See the Oregon top 10
4. Turn Sound Off
Select an option

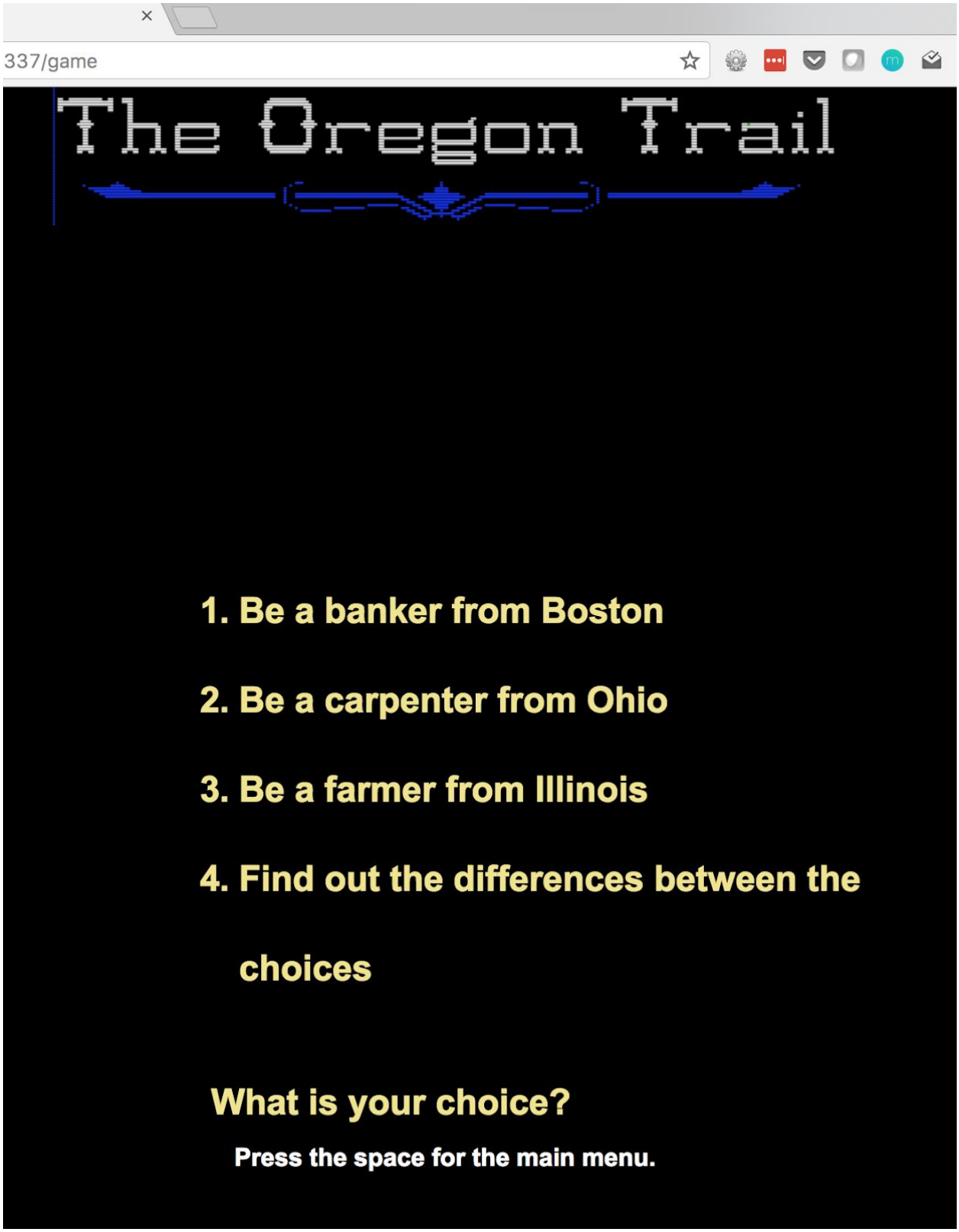
/topten

Oregon Trail

Top Ten

	Name	Score	Date
1)	Barney Rubble	2050	2/21/2014
2)	Pebbles Flintstone	1334	3/1/2010
3)	Daffy Duck	1100	3/1/2014
4)	Fred Flintstone	1000	1/11/2012
5)	Pokey Jones	950	9/10/1977
6)	Tazmanian Devil	725	3/1/2014
7)	Foghorn Leghorn	500	4/22/2014
8)	Bugs Bunny	300	11/2/2012
9)	Gumby Smith	200	6/14/2011
10)	BamBam Rubble	150	7/4/2013

Press the space for the main menu.


337/game
The Oregon Trail
1. Be a banker from Boston
2. Be a carpenter from Ohio
3. Be a farmer from Illinois
4. Find out the differences between the choices
What is your choice?
Press the space for the main menu.

The Oregon Trail

Days on the Trail:0
Miles Traveled:0
Current Weather:Warm
Current Health:100
Current Pace:Steady
Current Terrain:Grassland
# of Party Members:5

Press ENTER to travel a day or SPACEBAR to change pace!