Challenge 1 for Frontend Developer- Elife Tech Inc.
Name: Hrithik Gowda Date: 15-04-2023

About Elife Tech Inc: [https://elifelimo.com/]

Elife covers personal, business and corporate transportation needs, with a complete range of transportation options: from sedans to buses and luxury vehicles, in airport transportation services, local and long-distance transfers, hourly services, private tours and a wide etcetera.

Source for evaluation:

Website, chrome dev tools, page source code.

Challenge-1: technical evaluation of the website:

As per challenge I've been tasked with evaluating the website from a JavaScript perspective. These are my findings:

- The website seems to have a lot of JavaScript code that is responsible for various functionalities such as image sliders, contact forms, pop-ups, and menu toggles.
- Some of the JavaScript files are minified and compressed, which is a good practice to reduce the size of the files and improve the website's performance.
- The website uses a lot of third-party JavaScript libraries such as jQuery, Bootstrap, and Font Awesome. While these libraries can help developers build websites more efficiently, they can also slow down the website's loading speed if they are not optimized properly.
- The website seems to be using Google Analytics for tracking visitor behaviour and Google Maps for displaying the location of the limousine service.
- The website does not seem to have any major JavaScript errors or issues that could affect its functionality or performance.
- The website uses inline JavaScript code in several places, including event handlers and form submission functions. This can make the code difficult to manage and update and can also be a security risk if not properly sanitized.
- The website appears to use an outdated version of jQuery (v 1.11.3). It's generally recommended to use the latest version of jQuery or a more modern JavaScript library/framework to ensure better performance and security.
- The website uses several HTTP requests to load JavaScript and CSS resources, which can affect the website's performance and load times. It may be beneficial to combine and minify these resources to reduce the number of HTTP requests and improve performance.
- The website uses Google Maps API to display a map on the contact page. However, the API key is included in the JavaScript code, which can be a security risk. It's recommended to store the API key securely and access it using server-side code instead.
- The website uses several plugins and widgets, including a chat widget and a testimonial carousel. These plugins can provide useful functionality but can also add unnecessary bloat and affect performance if not used correctly.
- The website appears to be missing some accessibility features, such as alt tags for images and proper labelling for form fields. This can make the website difficult to use for users with disabilities.
- In the main.js file, you are using the "document. write method" to display the tracking results on the web page. This method is not recommended as it can overwrite the entire document if called after the page has finished loading. Instead, you should consider using DOM manipulation techniques to update the content of specific elements on the page.

Challenge 1 for Frontend Developer- Elife Tech Inc.

Name: Hrithik Gowda Date: 15-04-2023

- You are using the var keyword to declare variables. It is recommended to use let and const instead of var as they have block scope and can help prevent unintended variable hoisting.
- The for loop used to iterate through the “cookieEnabled array” can be simplified using a forEach loop or a for...of loop.
- In the “trackMouseMovement function”, you are using the “mousemove event” to track the user's mouse movement. This can be resource-intensive as it triggers the event handler every time the user moves the mouse. Instead, you should consider using the “mouseover and mouseout” events to track when the user enters and leaves the web page.
- The “trackScrollMovement function” can be improved by debouncing the scroll event. This will prevent the event handler from being called too frequently, which can reduce the performance of the web page.
- The” trackWindowSize function” can be improved by using the resize event to track changes to the window size. Currently, the function only tracks the initial window size.
- The code can benefit from better organization and modularization. Consider breaking the code into smaller functions and modules to improve readability and maintainability.
- Here's an example of how you could simplify the “cookieEnabled” loop using a forEach loop:

```
const cookieEnabled = ["Enabled", "Disabled", "Unknown"];


cookieEnabled.forEach((status) => {
const listItem = document.createElement("li");
const textNode = document.createTextNode(status);
ListItem.appendChild(textNode);
cookieList.appendChild(listItem);
});
```

- You are using the eval() function in the “trackLocalStorage function” to evaluate the value of the “localStorage object”. The eval() function can be dangerous as it can execute any arbitrary code passed to it, which can lead to security vulnerabilities. Instead, you should consider using a safer alternative such as JSON.parse() to parse the JSON string.
- You can optimize the “trackLocalStorage function” by using the storage event to track changes to the “localStorage object”. This will allow you to update the tracking results in real-time without having to periodically check the localStorage object.
- You can improve the code by adding error handling and validation to prevent unexpected errors and user input. For example, you can check that the necessary APIs and objects are available before using them and validate user input before using it.
- You can modularize the code further by separating the tracking logic into different modules and files. This can help make the code more organized, maintainable, and reusable.
- The code can be further optimized for performance by reducing the number of event listeners and minimizing the amount of DOM manipulation required. You can also consider using asynchronous code and caching to improve performance.

Challenge 1 for Frontend Developer- Elife Tech Inc.

Name: Hrithik Gowda Date: 15-04-2023

- The onload event in the window object can be used to track when the page has finished loading instead of relying on the “setTimeout function”.
- Here's an example of how you could use the storage event to track changes to the “localStorage object”:

```
window.addEventListener('storage', function(event) {
  if (event.storageArea === localStorage) {
    const listItem = document.createElement("li");
    const textNode = document.createTextNode(`${event.key} - ${event.newValue}`);
    listItem.appendChild(textNode);
    localStorageList.appendChild(listItem);
  }
});
```

- You can use modern JavaScript features and syntax to improve the readability and maintainability of your code. For example, you can use arrow functions, template literals, destructuring, and spread syntax where appropriate.
- The “trackWindowSize function” can be improved to track changes to the window size continuously instead of just once when the page loads. You can achieve this by adding an event listener to the resize event and updating the tracking results accordingly.
- You can improve the user experience by adding more informative and user-friendly messages and error handling. For example, you can show a message when the user's browser does not support a particular API or feature, or when there is an error in the code.
- Here's an example of how you could use the “Object.entries() method” to simplify the “trackLocalStorage function”:

```
Object.entries(localStorage).forEach(([key, value]) => {
  const listItem = document.createElement("li");
  const textNode = document.createTextNode(`${key} - ${value}`);
  listItem.appendChild(textNode);
  localStorageList.appendChild(listItem);
});
```

Challenge 1 for Frontend Developer- Elife Tech Inc.

Name: Hrithik Gowda Date: 15-04-2023

My points for improvement:

- JavaScript scripts block parallel downloads; that is, when a script is downloading, the browser will not start any other downloads. To help the page load faster, move scripts to the bottom of the page if they are deferrable.
- Minify and combine JavaScript and CSS files to reduce the number of HTTP requests and improve performance. There are many tools available to accomplish this task, such as Gulp, Grunt, or Webpack.
- Use an updated version of jQuery or a more modern JavaScript library/framework to ensure better performance and security.
- Remove inline JavaScript code and instead move it to external JavaScript files to improve code organization and maintainability.
- Securely store the Google Maps API key and access it using server-side code instead of including it in the JavaScript code.
- Use lazy loading for plugins and widgets to improve page load times and reduce unnecessary bloat.
- Add accessibility features, such as alt tags for images and proper labeling for form fields, to improve the user experience for all users.
- Avoid using document.write() to dynamically insert HTML into the page, as it can cause performance issues and unexpected behavior. Instead, use DOM manipulation methods like appendChild() or innerHTML.
- Use the defer or async attributes when loading external JavaScript files to avoid blocking the page load.
- Use JavaScript modules to organize your code into logical units and avoid global scope pollution. Modules help to make the code more modular, easier to maintain, and less prone to naming conflicts.
- Optimize images by compressing them and reducing their size. Large images can cause slow page load times and negatively impact performance. You can use image optimization tools
- Use the async or defer attribute when loading external scripts to improve page load times. The async attribute loads the script asynchronously, while the defer attribute loads the script after the document has finished parsing.
- Reduce the number of HTTP requests: The website currently makes a number of HTTP requests to load various JavaScript files and other resources. One way to reduce the number of requests is to combine multiple JavaScript files into a single file
- Use lazy loading: The website has a number of images that are loaded on page load. Implementing lazy loading for these images can reduce the initial page load time, as images are only loaded as they become visible in the viewport.
- Implement caching: Caching can significantly improve website performance by storing frequently accessed data in the browser's cache or on the server. Implementing caching for JavaScript files, images, and other resources can reduce the number of HTTP requests and speed up page load times.

****