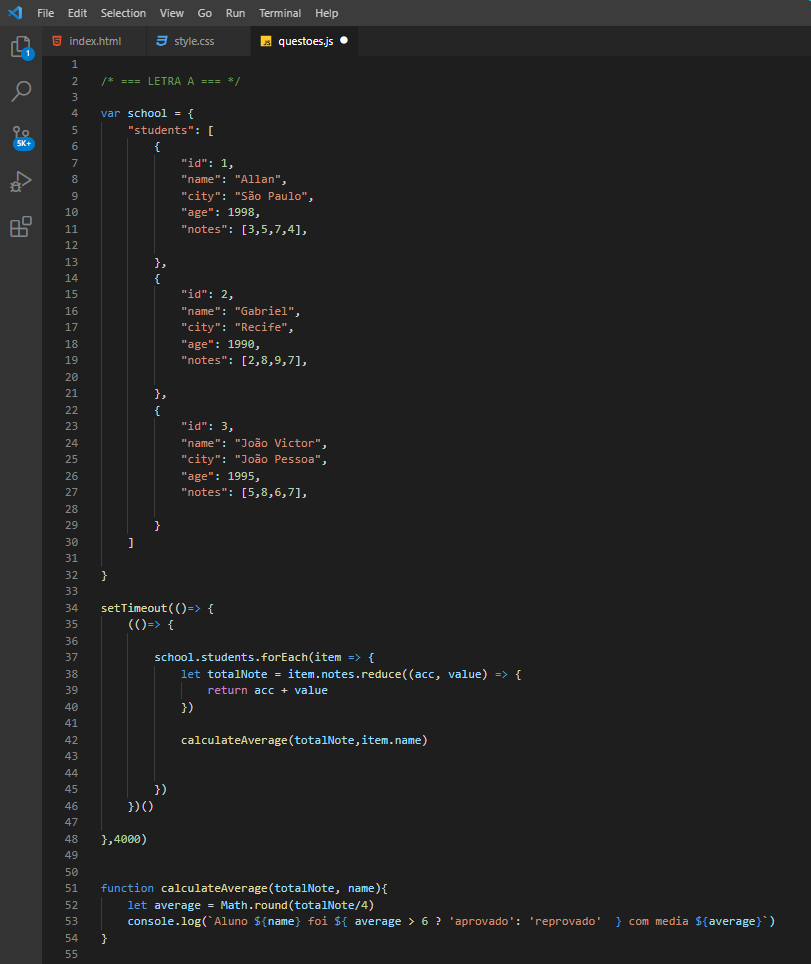
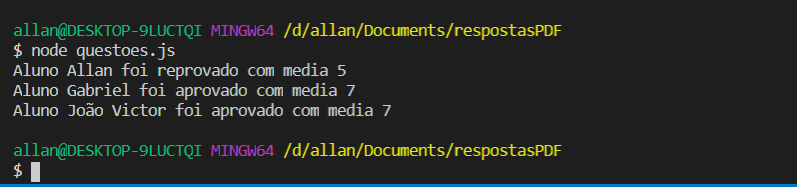
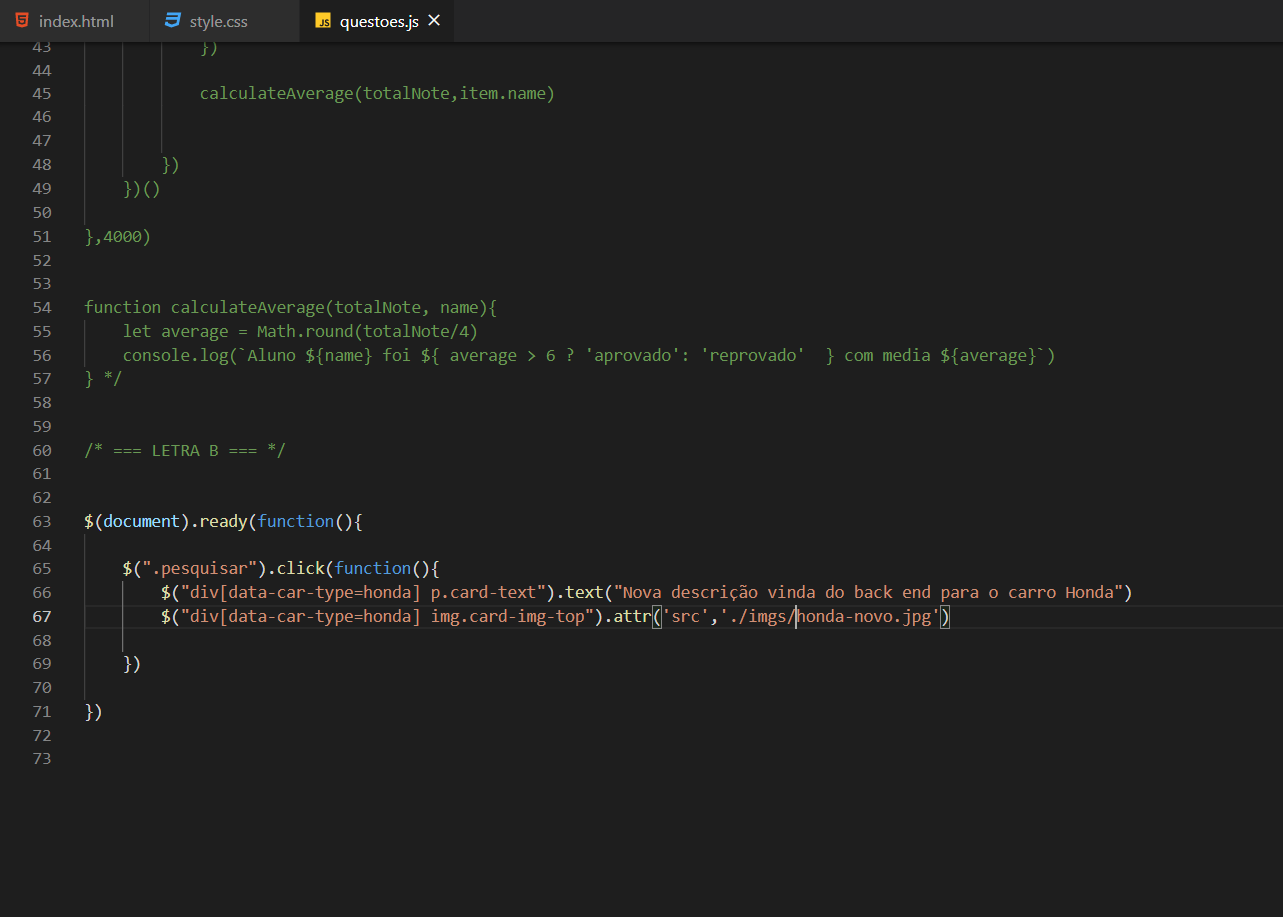
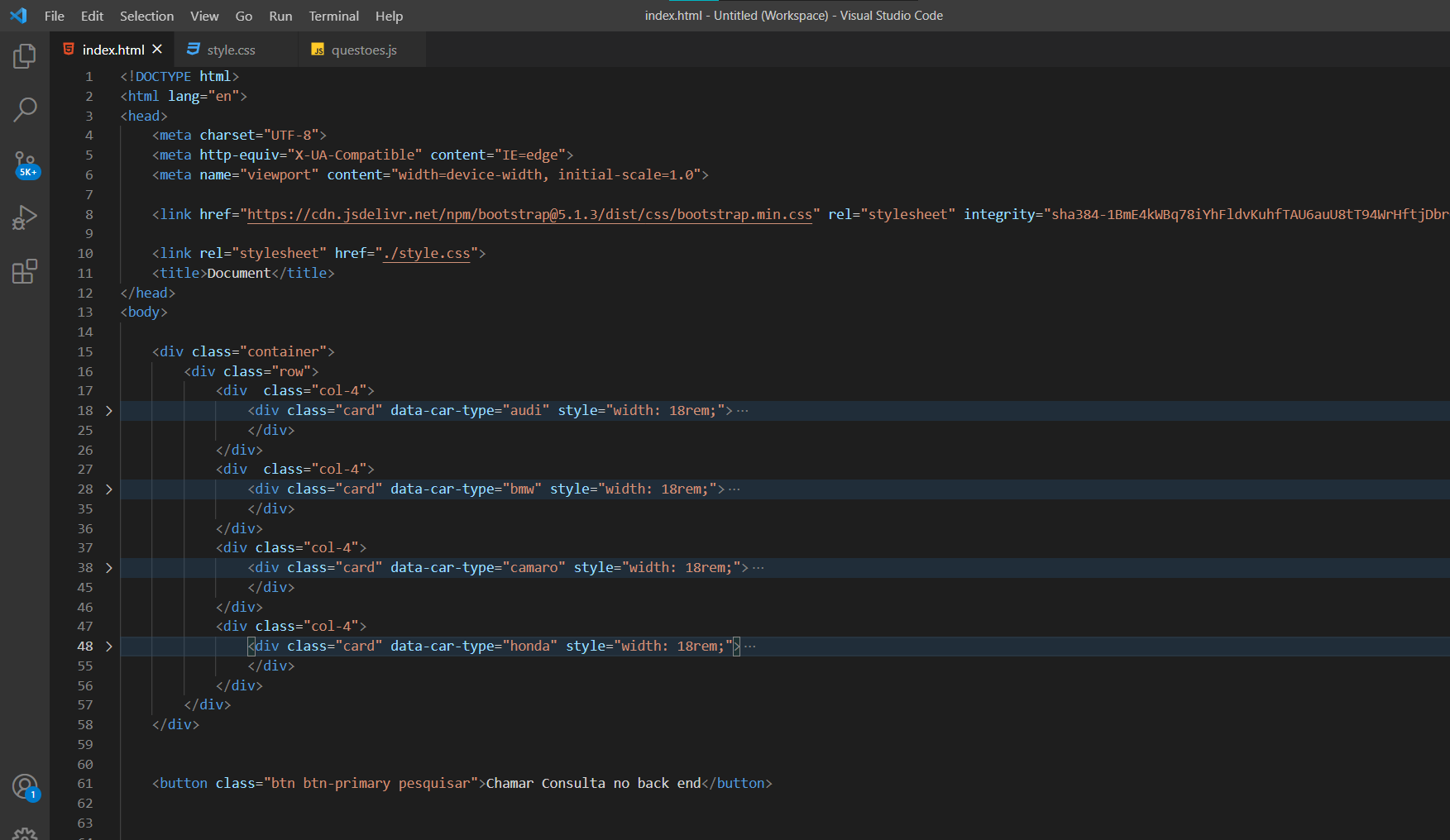
**Desafio Front end - iCarros**

**QUESTÃO A** - IIFE a gente também conhece como função anônima. É uma função que é executada logo após a sua criação. Uma de suas vantagens é o encapsulamento de escopo, ou seja qualquer variável que for declarada dentro da IIFE nao vai poder ser acessada fora dessa função, isso por si só já resolve diversos problemas que podem acontecer relacionado a conflito de nomes de variáveis no escopo do objeto global do javascript. No exemplo abaixo eu simulo o retorno do back end com um Timeout e logo em seguida chamo uma função auto executável para verificar se o aluno foi aprovado ou não.





**QUESTÃO B** -



no exemplo acima eu criei uma estrutura de cards com informações sobre carros e simulei um retorno de uma consulta no back end clicando no botão, onde o retorno da consulta deve alterar apenas os cards que são da marca honda, representados pelo atributo **data-car-type,** trocando a descrição e a imagem via jquery.

**QUESTÃO C** - São tag importantes que contêm informações relevantes do seu site que vão ser úteis para os algoritmos dos mecanismo de busca, por exemplo o google. Exemplos

**description**

<meta name=”description” content=”Alguma descrição sobre o que é seu site.”>

Essa descrição aparece no momento da busca pelo site. Representada abaixo



**keywords**

<meta name=”keywords” content=”programação, javascript ,js, frontend, react, angular ”>

é uma lista de palavras que você considera importantes para o conteúdo do seu site e influencia no rankeamento de sua página nos mecanismos de busca.

**QUESTÃO D** - O **atributo async** determina que aquele script vai ser executado assincronamente, ou seja, em paralelo ao resto dos carregamentos seguintes. Exemplo, se um script async estiver no meio da página html, o script será lido e executado em paralelo a análise do conteúdo html que não será pausada para dar prioridade aquele script.

**script sem async ou defer** - quando a tag script é declarada desse jeito, o seu conteúdo vai ser lido de forma síncrona, ou seja, vai parar todo o resto do carregamento html até aquele script ser totalmente lido e executado.

O **atributo defer** também tem seu carregamento assíncrono, porém o script só será executado quando o conteúdo html for totalmente lido e finalizado.

**QUESTÃO E** - A cor vai ser vermelha, por causa do tipo de seletor que está bastante específico (conceito de especificidade css) para determinar a cor das li. A linha com a class=”destaque” só ficaria azul se for adicionado o **!important** que ignoraria a cor que foi setada no seletor acima

.destaque {

color: blue !important

}

**QUESTÃO F** - (**Sobre CSS)** Porque o conteúdo css vai carregar primeiro junto com o html mostrando a página já estilizada. Em alguns modelos de projetos o css carregado inicialmente nas tag <head> é apenas o essencial para as primeiras impressões da página.

**(Sobre Js)**  carregar scripts no final da tag body permite que todo o resto do html seja carregado para o usuário antes, e só depois os scripts vão começar a ser processados, dando uma impressão de uma página mais rápida.

**Exceções**

**(CSS)** muitas bibliotecas js manipulam o css e os insere inline na tag, por exemplo

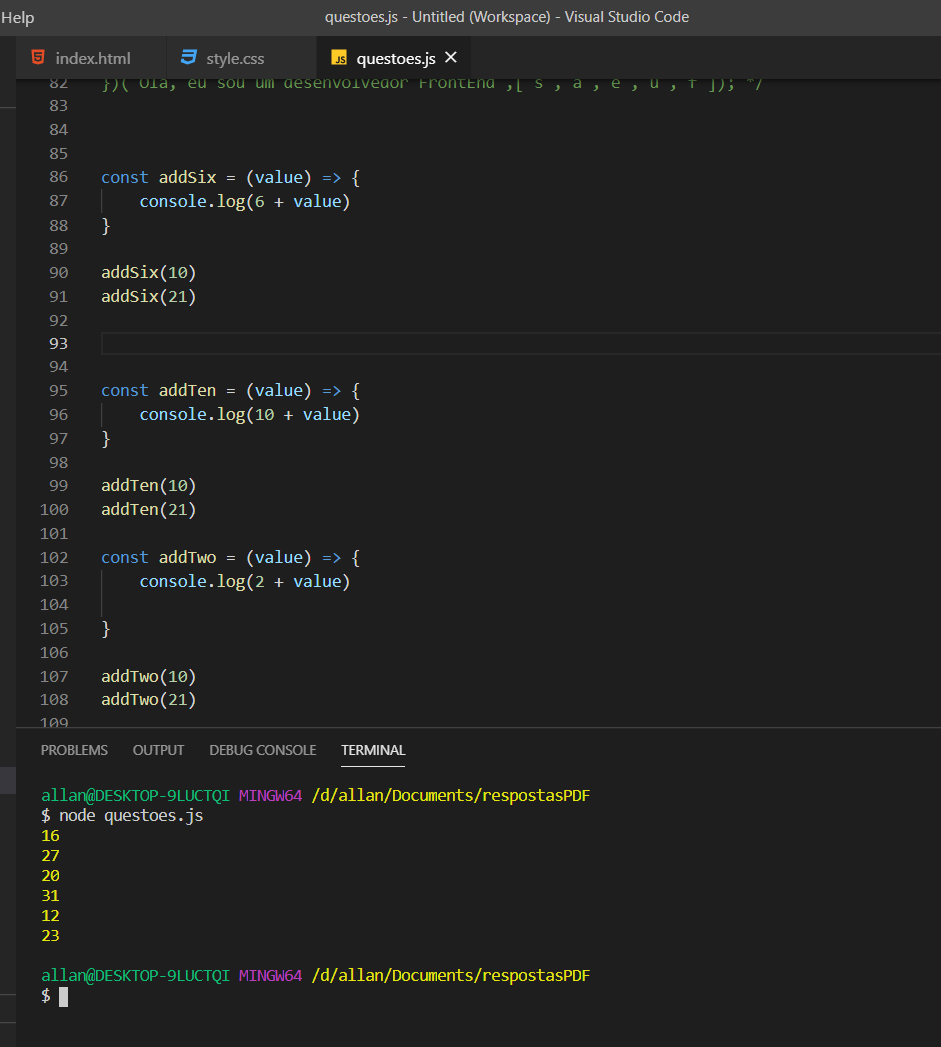
<div style=”transform:translate(50%); padding: 10px 30px;”>....</div>

O css também pode ser carregado de forma assíncrona conforme as interações do usuário com a página, evitando carregar todo css do projeto de uma vez na tag <head>

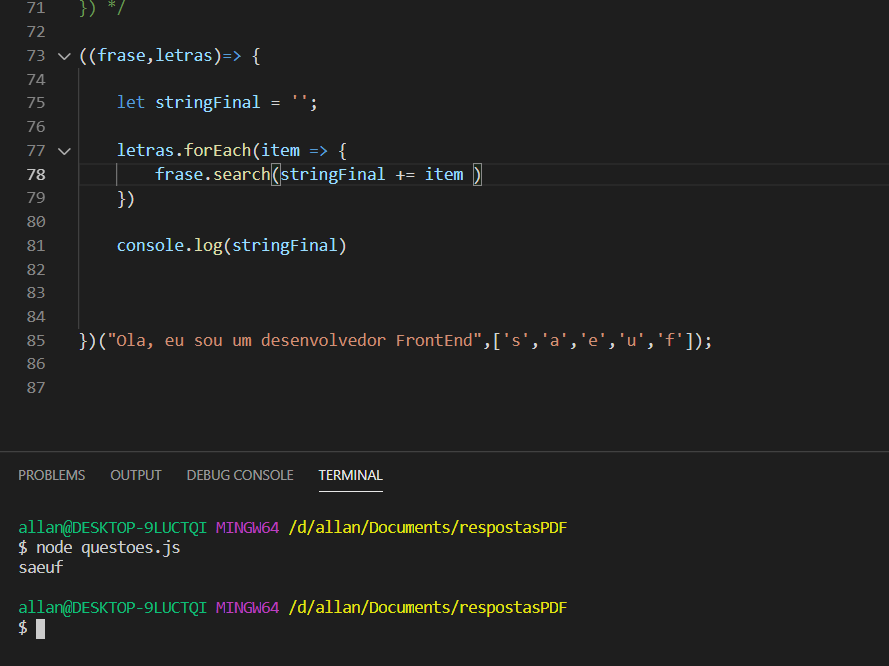
**(JS)** Colocar script no head do site garante que aquele script vai ser executado antes de qualquer elemento, garantindo que quando a página for montada, aquele script vai estar presente e pronto para funcionar. Por exemplo uma tag do google analytics e pixel do facebook.

**QUESTÃO G** - Na letra **A** vai printar **hello**, porque existe o operador lógico || (or) logo ele verifica se a primeira string (**hello**) é true, como a string não retorna false porque ela não é vazia, o print no console vai ser **hello** e a segunda string (**world**) é ignorada.

Na letra **B** vai ser printado **bar,** porque o operador && (and) vai verificar que as duas string são true, porém só vai ser printado **bar** porque ela está por último. Se algumas das duas fossem vazias não seria printado nada no console.

**QUESTÃO H** - 

**QUESTÃO I** -



**QUESTÃO J -** Independente do tipo de aplicação, tamanho e escalabilidade, 10 anos é um bom tempo para os padrões web se modernizarem, eu faria uma releitura do layout do sistema, com conceitos de UI/UX modernos e com interfaces mais bem trabalhadas esteticamente. Em questões técnicas, implementaria o conceito de grids do bootstrap, para montar uma estrutura (html/css) com padrão moderno que já vem sendo adotado pela maioria dos sites, e também mantendo a responsividade, se adequando aos diversos tipos de dispositivos que temos no mercado. Fazer chamadas de css e js mais inteligentes, diminuindo a quantidade de códigos, de bibliotecas de terceiro, que não usamos, fazer minificação de arquivos css/js e compressão de imagens.

**QUESTÃO K -** Usaria React junto com Node.js, react porque é um framework extremamente performático, pelo fato de não vir carregado com um bundle de funções e bibliotecas, que muitas vezes acabamos nem usando, assim podemos ganhar desempenho instalando recursos que realmente nossa aplicação vai precisar, usando a sua lógica do Virtual DOM o react consegue ser bastante eficaz relacionado a complexidade de diversos processos que estão acontecendo na tela. O React Native é possível programar em js e fazer builds para dispositivos mobile. Node pela vantagem do back e front serem feitos utilizando a mesma linguagem (js) resolvendo diversos problemas de comunicação entre esses dois lados, diminuição do tempo de execução pelo fato do node trabalhar de forma assíncrona. Javascript é umas das linguagens mais populares do mundo, explorar fóruns para tirar dúvidas, bibliotecas e códigos prontos não será problema.