04

Critical Path

Introdução

Nesse novo curso sobre **Web Performance** vamos elaborar novas técnicas de otimização. Antes de começarmos, vamos relembrar um pouco do que aprendemos anteriormente:

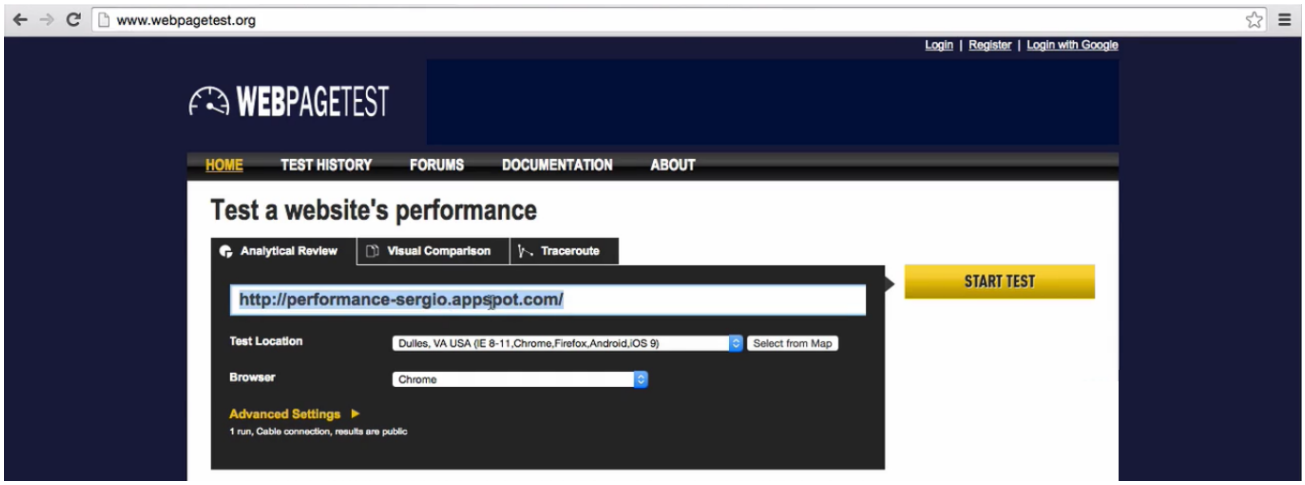
1. Minificar arquivos;
    1. Gzip;
    2. Otimização de imagens;
    3. Concatenar CSS/ JS;
    4. Usar *Sprites*;
    5. *Cache*;
    6. Inline recursos;
    7. Paralelização de *requests*;

Nesse segundo módulo discutiremos técnicas mais elaboradas e avançadas, mas antes disso vamos aprender a medir o impacto de utilizar uma determinadas técnica e também veremos o que exatamente medir. Usaremos o mesmo projeto que estávamos modificando no curso 1.

Medidas

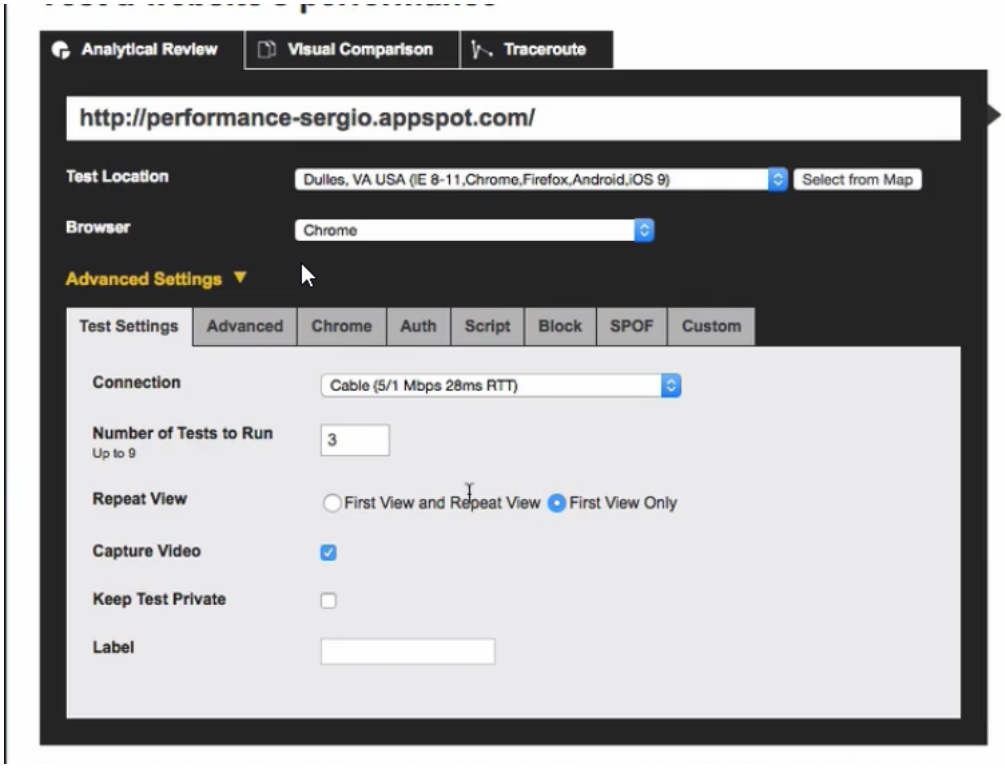
A Performance é usabilidade, ou seja, quem nos dirá se um site está rápido é o próprio usuário. É importante, entretanto, utilizarmos algum tipo de métrica para mensurarmos a performance de um site. Vamos fazer isso utilizando uma ferramenta online chamada **Web Page Test**: http://www.webpagetest.org/ (http://www.webpagetest.org/) .

Para começar o teste teremos, primeiramente, que possuir uma "url" pública. Já trabalhamos com esse site anteriormente, a ideia, é compreendermos mais profundamente as possibilidades que ele nos oferece.



O primeiro passo é inserir o endereço do site no campo em branco que aparece assim que abrimos a página. Podemos escolher qual o navegador que utilizaremos e a localização, por padrão ele traz uma localização americana. A ideia é que quanto mais perto da realidade do usuário, mais real será a simulação.

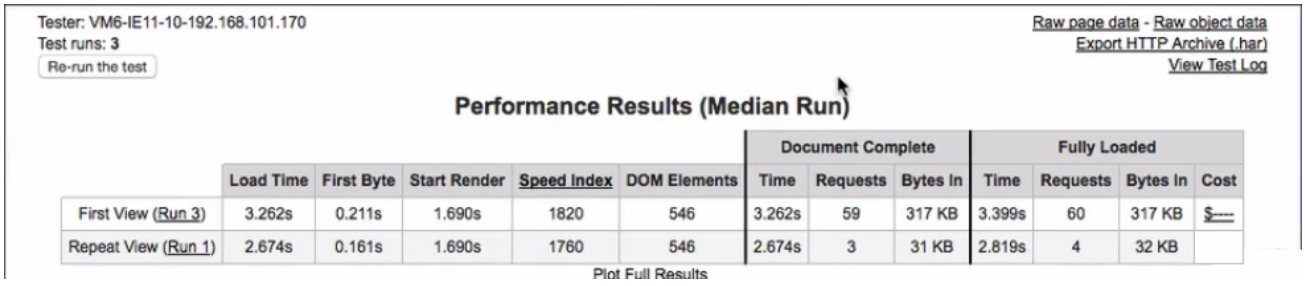
Na parte de *Advanced Settings*, na aba *Test Settings*, podemos configurar a Conexão, o número de testes que queremos que ele realize (vale a pena que esse número seja maior que 1 pois variações ocorrem, escolheremos 3 vezes). Podemos selecionar a opção de *Repeat View* que é interessante para testarmos o *cache*. Outra opção interessante e que marcaremos é a *Capture Video,* pois é gravado um vídeo do carregamento da página.



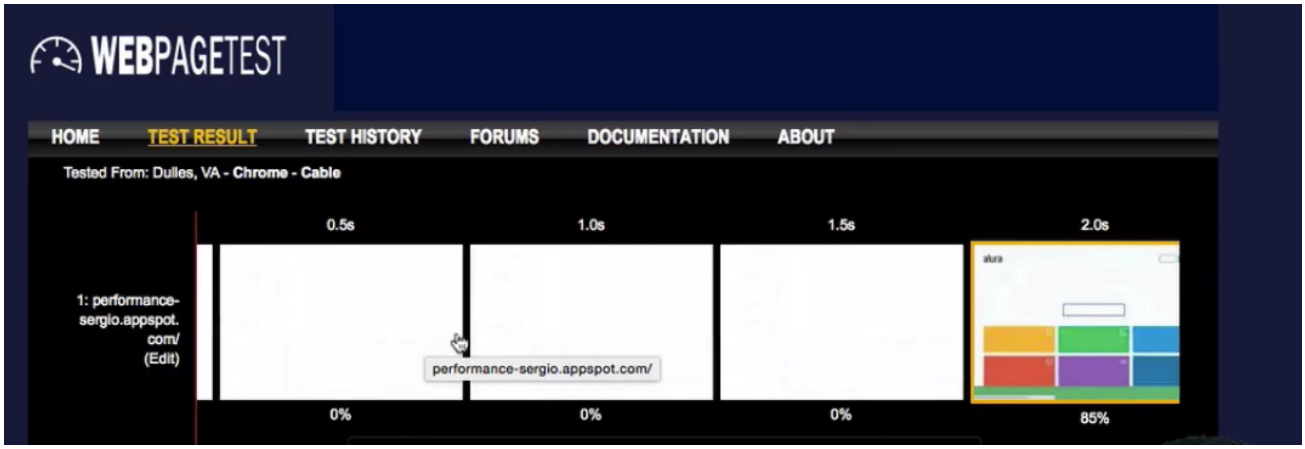
Existem diversas opções, mas nem todas elas foram comentadas. Se você tiver interesse em aprender sobre as demais possibilidades basta explorar as opções avançadas. Bom, agora que fizemos as configurações vamos rodar o teste. Tenha paciência, ele pode demorar um pouco.

Como pedimos para que fossem realizadas três testes ele traz, por padrão, uma mediana e nos mostra, gráficos de cada uma das execuções. Se clicarmos no "*Run 3*" será mostrada a mediana e poderemos visualizar o gráfico em forma de cascata, o *Waterfall*. Na sequência podemos visualizar o gráfico de *Connection View*. Mais abaixo temos uma tabela detalhada de cada um dos sessenta *requests* que fizemos.

Mas, a tabela mais interessante é a seguinte:

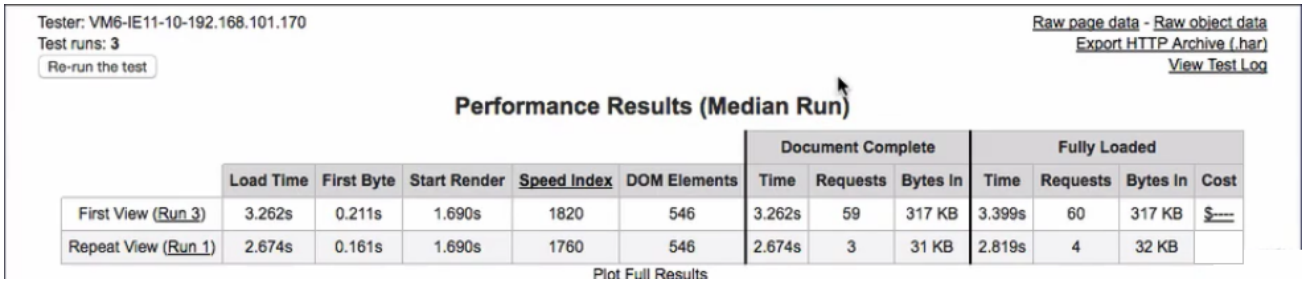


Esses números são os que fornecem a métrica para sabermos se nosso site está agil ou não. Temos diversos dados nessa páginas, alguns deles são mais importantes para que compreendamos a experiência do usuário. Vamos retornar a página que nos fala sobre o *Filmstrip View* e vamos abrir o vídeo.



Veremos quadro a quadro a renderização do site, o intervalo é de meio segundo, mas podemos diminuir esse tempo. Podemos visualizar também os diferentes filmes dos distintos gráficos. Olhando os gráficos podemos nos perguntar em que ponto o usuário percebe que o site está carregado e é isso que importa, o ponto de vista de quem está utilizando o site.

Voltando a tabela com os números:

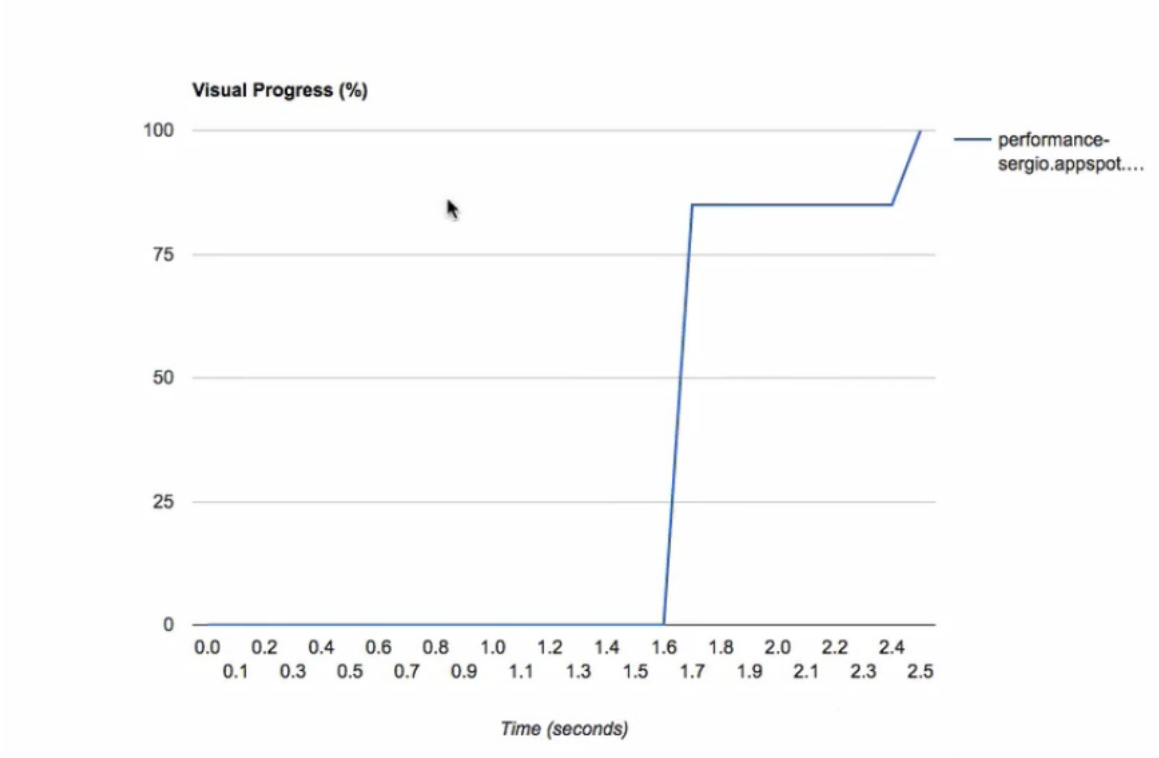


Alguns deles traduzem melhor a experiência dos usuários. O *load time* antes era mais interessante, entretanto, atualmente, ele já não é tão bom para a medição de performance. Em particular, estamos interessados no *start render*, ele indica uma sensação de início para o usuário, para quem está utilizando o site a pior parte é quando a tela está toda em branco, isso significa, para o usuário, que nada está sendo carregado.

A coluna *Speed index* indica o tempo que tarda o topo, o *above the flood*, para ser desenhado/carregado. Isso é importante pois o topo é o que passa a estabilidade ao usuário. Na tabela ele indica que no segundo 1.8 já temos um cabeçalho estável.

O *Visually Complete* indica que o a visualização do site estão completas, ou seja, a página de cima a baixo está completa. O que não nos é tão interessante, o que mais conta é a percepção de estar completo, por isso, os outros dois dados são mais interessantes.

Um detalhe: O **Web Page Test** não consegue entender direito as fontes customizadas, como é o caso do site do **Alura**, por esse motivo, ele interpreta de maneira equivocada o carregamento da página. Mas, podemos visualizar um outro gráfico que nos auxilia a compreender, é o *Visual Progress*, também:



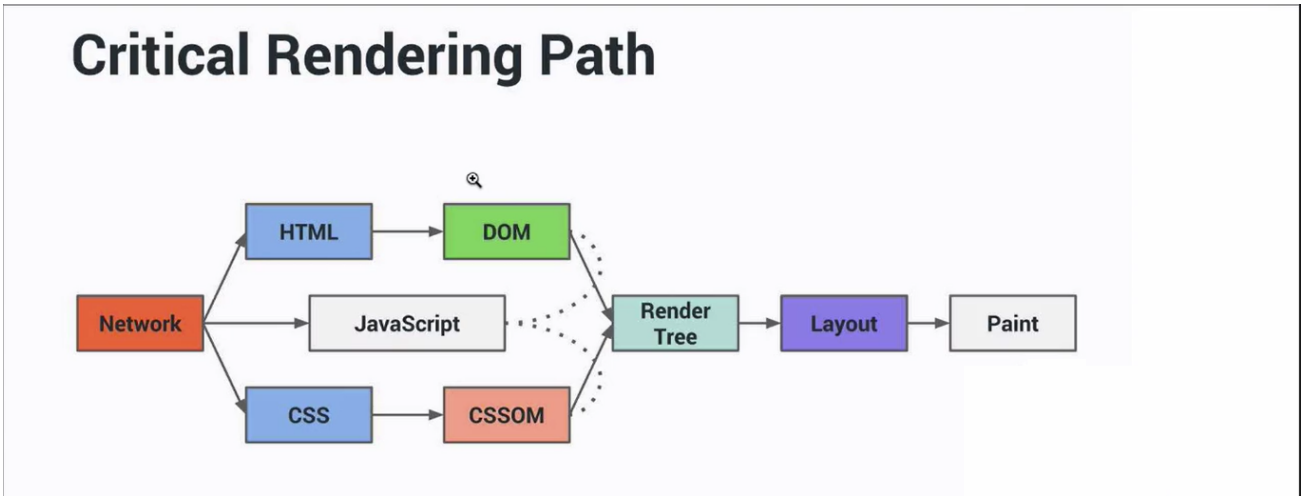
Ele é interessante para comparação. Você pode otimizar o site e fazer uma comparação desses gráficos. É interessante nos atentarmos para a renderização perceptiva, aquela que o usuário tem a sensação de que o site carregou efetivamente. Se quiser visualizar o site carregando em formato de vídeo você também pode e pode alterar também a velocidade do filme:

O que queremos deixar aqui é que analisar o *Filmsrtrip* é muito importante, principalmente, se atentar ao início da renderização e também - e talvez mais importante ainda - quando ela começa a ser significativa e, por fim, o visualmente completo (o topo).

É importante compreender: se a entrega da página for boa o resto já não é "tão" importante!

Destravando o *Rendering Path*

Vamos refletir um pouco a cerca das tarefas que o navegador realiza para carregar um site. Estamos discutindo isso pois existem algumas tarefas que o "navegador precisa fazer de maneira a executar os recursos na ordem correta e também da maneira mais otimizada possível. Isso que estivemos falando é o que é denominado *Critical Rendering Path*, ou seja, o caminho crítico de renderização de uma página. Isso é o que está entre o que o usuário consegue visualizar e a página totalmente em branco. Observe a seguinte ilustração:

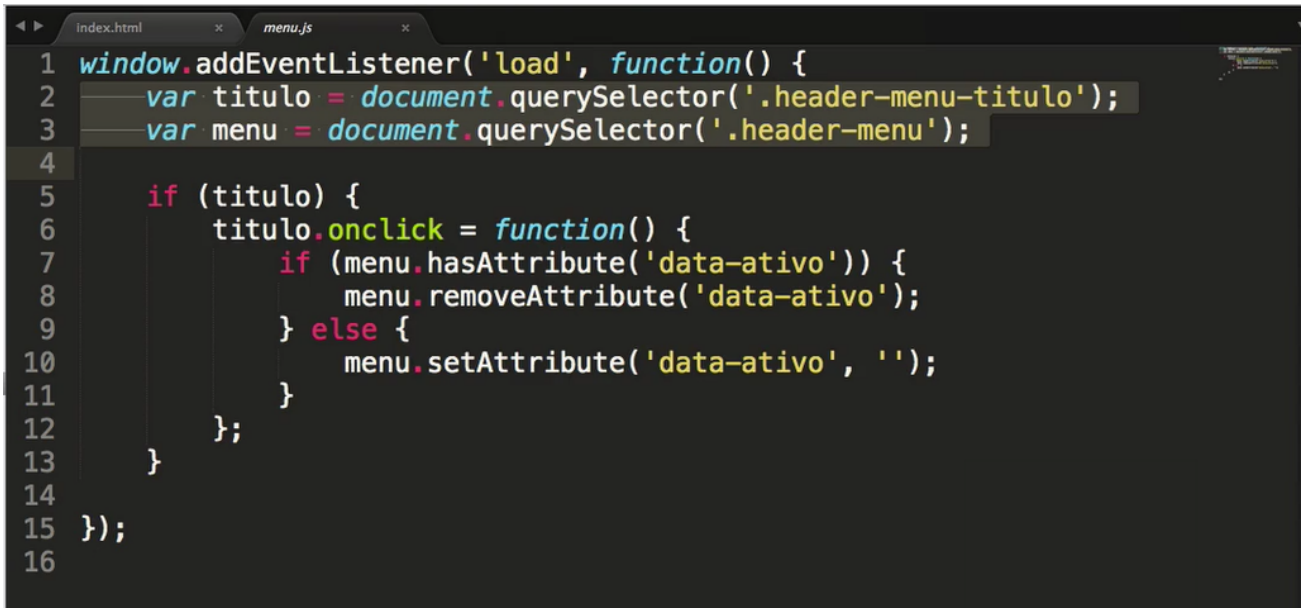


Para resolver esse problema é necessário que destravemos nosso *Critical Rendering Path. Se o "CSS" é blocante e essencial para a renderização final o que desejamos fazer é priorizar o* css*. Observando nosso código conseguimos verificar que ele não é prioridade. O que faremos para resolver esse problema é mover esse "CSS" para o topo, para o* head*, para que ele passe a ser executado o mais rápido possível, uma vez que ele afeta o* ***Paint*** *final e também para que seja o menos possível afetado pelo* Javascript*.

Atenção! É importante não misturar o "CSS" com o *javascript*.

É importante, ainda, refletirmos sobre a posição dos *scripts*. Movendo o "CSS" para o *head* isso significa que o *body* não irá carregar enquanto o *head* não estiver pronto. O que temos que fazer para destravar a renderização do "DOM"? Vamos mover o *javascript* para depois do "html". Isso faz com que o *javascript* seja o último ponto na priorização do *Critical rendering Path*. Ou seja, destravamos o nosso caminho. O *Script* pode estar em qualquer lugar da página, não existe nenhuma regra que o coloque no início. Lembrando que toda vez que fazemos uma alteração é necessário que *rebuildemos* o site.

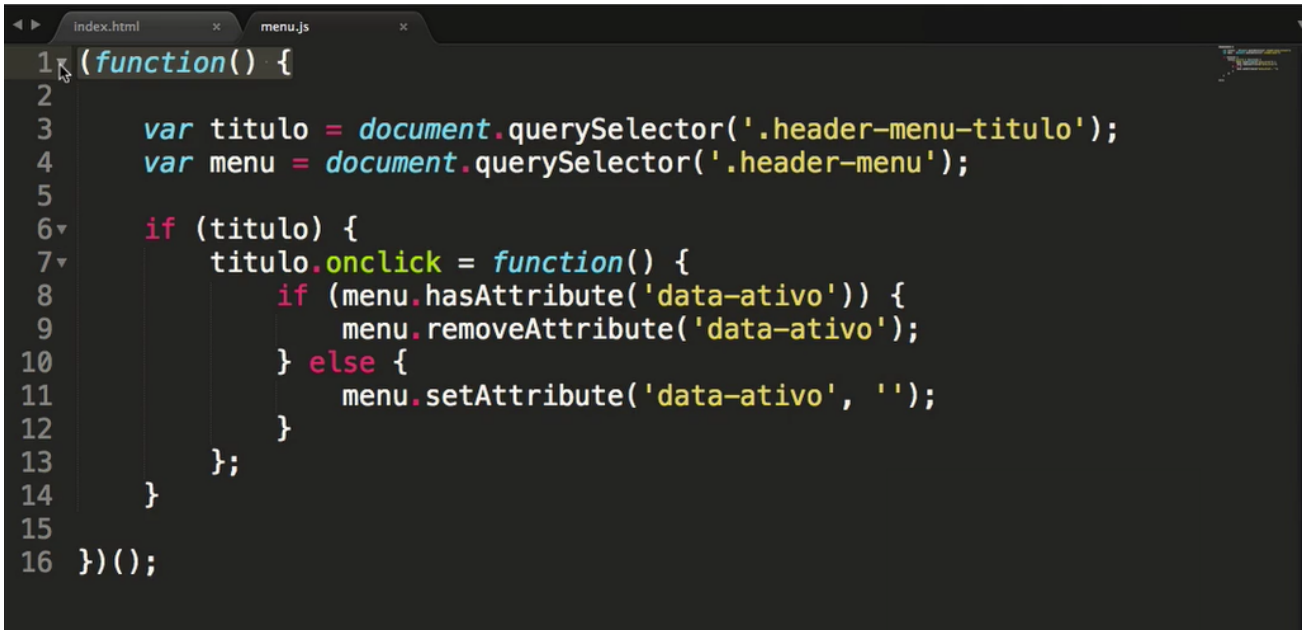
Acessando o *localhost* podemos verificar que nossa página está igual. Um bônus para essa estratégia de jogar o *javascript* para baixo é que podemos revisar os *scripts* e observar que estávamos usando a prática comum de colocar o *Script* em um evento de *onload*. A ideia é que se colocarmos o script embaixo não precisamos mais do *onload*, pois temos a certeza que os elementos estarão carregados, assim, podemos retirar o *onload*:



Vamos apagar o seguinte:

```
window.addEventListener
```

Essa é uma otimização pequena, mas ainda assim, importante. Se você achar necessário pode embalar o código em uma função para não vazamento das variáveis.



Fazendo um *build* novamente veremos que não temos nenhum erro no *Console*.

O que aprendemos aqui é que para destravar o *Rendering Path* precisamos otimizar o "CSS" e tirar o *javascript* para o início da renderização.