Melhorando o carregamento do CSS

O CSS pode ser um fator chave na hora de "adiantar conteúdo" para que os usuários possam olhar alguma coisa enquanto o resto do site é carregado, ao invés de uma tela em branco que não diz nada. Se ainda não sabe bem como medir a velocidade do seu site, o seguinte artigo pode ajudar:

Medindo a velocidade do seu site

CSS

O CSS é o que tem menos impacto se comparado com imagens e JS, mas não deixa de ser importante.

Assim como o JS que é carregado antes do HTML e bloqueia o carregamento, o CSS faz a mesma coisa e a diferença do JS, onde você pode carregá-lo ao final do `body` (evitando que seja tão bloqueante), com o CSS não é possível fazer o mesmo porque o resultado seria um HTML sem estilos (quer dizer quebrado) até que o último CSS seja carregado.

É por isso que tem que pensar em CSS crítico e CSS assíncrono (não crítico), que pode ser carregado de forma assíncrona.

CSS Crítico

O CSS crítico é basicamente o CSS que precisa obrigatoriamente que deixe-o bloquear o carregamento do HTML para exibir o conteúdo mais relevante sem que fique quebrado, por exemplo: a navegação principal, o logotipo, um banner, a listagem de produtos, etc.

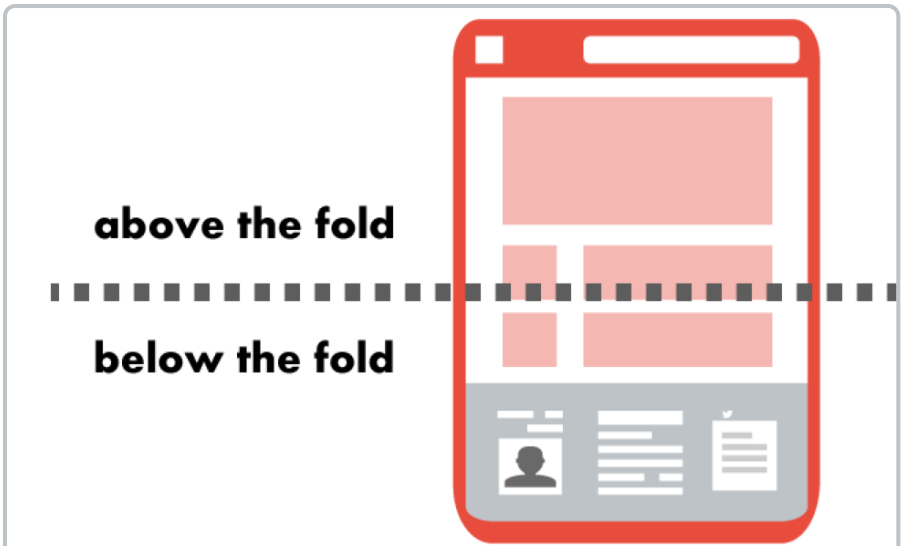


Gráfico de “above the fold”, no artigo “Below the fold: What does below the fold mean?”

O CSS que você considerar como crítico deve ser carregado *inline* no documento, pois é necessário que seja carregado o mais rápido possível sem ter que fazer um request de um arquivo de CSS ao servidor e atrasar o carregamento. Embora não seja o que te ensinaram (adicionar estilos inline) neste caso é justificado com base ao objetivo a alcançar.

Em Nuvem shop temos o CSS crítico adicionado com Twig da seguinte maneira:

```
{# Critical CSS to improve the percieved performance on first load #}
```

Fazer isso é o mesmo que inseri-lo inline mas fica num arquivo à parte e é muito mais prolixo.

Se você trabalha com Bootstrap, vai ter levar em conta dividi-lo em 2 partes, uma crítica e uma não crítica. Tem que pensar quais são os elementos que você esta fazendo uso above the fold (nav, botões, utilities classes como `pull-left`, `pull-right`, etc) e coisas que podem ficar em segundo plano (modals, tables, algúns forms, etc)

Uma ajuda que você poder usar para evitar exibir conteúdo "quebrado"

até que tudo o CSS fique carregado, é esconder os elementos não críticos com uma class para mantê-los invisíveis até ter todo o CSS pronto.

Para isso você pode pensar duas classes de CSS:

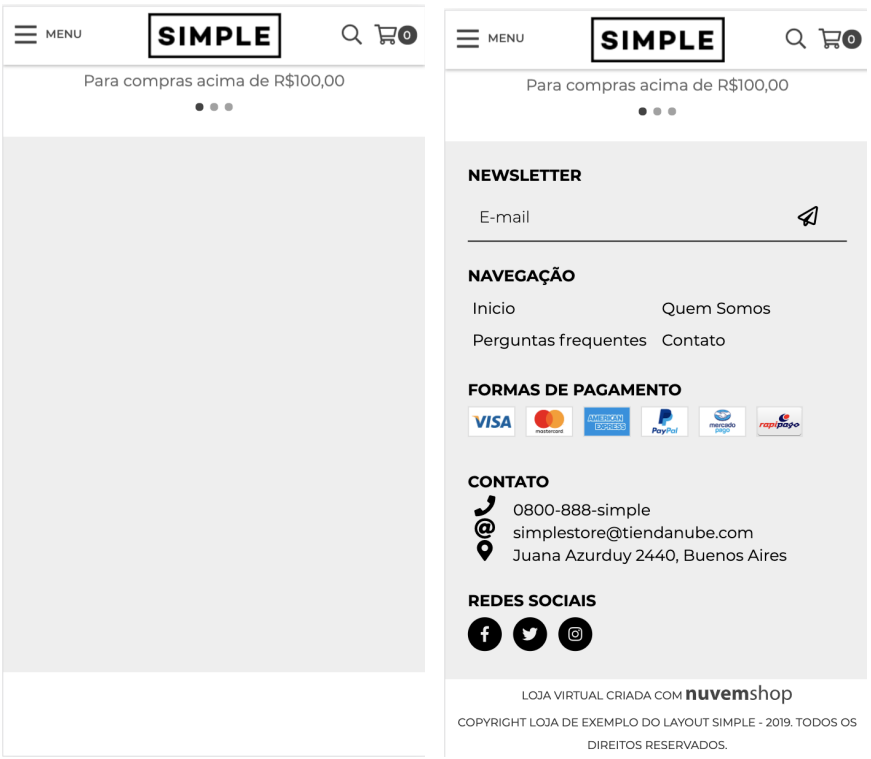
Primeiro tem que adicionar no seu CSS crítico as seguintes classes no final (uma para esconder os elementos mas que ainda ocupem um espaço e outra para que fiquem ocultos sem ocupar espaço):

```
.visible-when-content-ready{
  visibility: hidden!important;
}
.display-when-content-ready{
  display: none!important;
}
```

Depois no CSS assíncrono (vou falar mais adiante sobre isso) tem que adicionar o seguinte, também no final:

```
.visible-when-content-ready{
  visibility: visible!important;
}
.display-when-content-ready{
  display: block!important;
}
```

Agora você pode adicionar as classes aos elementos que não são críticos mas tem um risco de ser exibidos como "quebrados" até ter tudo o CSS carregado, por exemplo pode ser no rodapé fazendo que seja visualizado como o seguinte exemplo:



Na esquerda tem o rodapé oculto mas ainda ocupando um espaço e na direita o mesmo rodapé com seu CSS pronto para ser exibido.

E pode evitar mostrar um rodapé quebrado até que seu CSS esteja pronto, como no exemplo embaixo

O rodapé ficaria assim se as clases visible-when-content-ready ou display-when-content-ready não fossem usadas

CSS assíncrono

Basicamente é tudo o que você não considere como crítico, deveria ser muito já que é todo o CSS que está sendo removido para não bloquear o carregamento.

Mas, como pode carregar o CSS assíncrono? Tem que usar o plugin loadCss feito com Javascript e permite "demorar" o carregamento do CSS que não precisa de imediato. O carregamento vai ser definido do mesmo jeito que o browser trabalha no tag de JS `async`.

Para usar o plugin, simplesmente tem que (no tag head do HTML) adicioná-lo da seguinte maneira:

```
```

E embaixo fazer os *requests* aos arquivos CSS:

```
```

Tem que lembrar que a url do exemplo anterior fica feito com Twig mas pode adicionar a url que quiser. É importante que combinem o `id` do tag `script` com o id usado no `getElementByID`, porque o que faz o script é inserir antes do tag `script` com o `id` "X" (dentro no DOM), o mesmo CSS com o `id` "X". Neste caso o `id` é "style-css".

O mesmo plugin pode ser usado para o carregamento do CSS de Font Awesome e evitando o carregamento duma fonte que pode chegar a ser pesada (ainda em sua versão 5 carregando-a com JS `defer`, é melhor carregá-la com CSS assíncrono). No exemplo anterior, antes do "style-css", fica o chamado do Font Awesome (caso que precise sobrepor classes de Font Awesome com seu CSS)

Em nossas lojas temos o CSS dividido, como falei anteriormente, (assíncrono e crítico) mas também temos um SASS com as cores e fontes salvas na personalização de cada loja. Até agora o arquivo é carregado da maneira convencional bloqueando o carregamento do documento a propósito já que precisamos ter todas as cores e fontes aplicados antes de exibir algo, evitando um salto visual nestos estilos específicos (se fossem aplicados como algo assíncrono).

Fontes

Das fontes vou falar pouco mas tem muito espaço para melhorar neste aspecto. As mudanças aplicadas em nossas lojas foram em relação a fazer o *request* só das fontes que forma salvas na configuração em lugar de chamar a todas no documento. Chamar mais de 3 ou 4 fontes diferentes (ainda sendo as Google Fonts) pode começar a afetar a velocidade de carregamento, por isso sempre tentem manter a quantidade de fontes ao mínimo no seu site.

```
{% if params.preview %}

  {{'//fonts.googleapis.com/css?family=PT+Sans+Narrow:400,700|Open+Sans:400,3
00,700|Slabo+27px|Oswald:400,300,700|Lora:400,700|Montserrat:400,500,700|Robo
to+Condensed:400italic,700italic,300,400,700|Droid+Sans:400,700|Playfair+Disp
lay:400,700' | css_tag }}

{% else %}

    {{ [settings.font_headings,
    settings.font_navigation,
    settings.font_product_title,
    settings.font_buttons,
    settings.font_rest] | google_fonts_url | css_tag }}

{% endif %}
```

No exemplo anterior todas as fontes são carregados só se o usuário fica

na tela de configuração da sua loja, já que nesta tela as mudanças de fontes são visualizadas em tempo real e precisamos ter tudo carregado no HTML. Por outro lado (dentro do *else*) somente carregamos as fontes de cada configuração salvada em cada "setting".

Também, assim como usando o loadCSS o CSS pode ser carregado de maneira assincrônica, também podem carregar fontes do mesmo jeito mas o resultado vai ter algo chamado (Flash of Unstyled Text), que dizer que num momento o site vai ter fontes default (Arial, Helvética, etc) definidas no CSS, sans serif ou con serif; e logo o CSS com as fontes finais sera carregado. Isso pode melhorar a velocidade de carregamento mas também gera um salto visual nos textos.

Para mais informação no FOUT (e outros casos como FOIT e FOFT) deixo um artigo um artigo que fala sobre isso.

Que segue?

Se você já otimizou tudo o que tem a ver com CSS, recomendo ler as mudanças feitas em outras áreas:

Melhorando o carregamento das imagens e SVGs
Melhorando o carregamento do Javascript