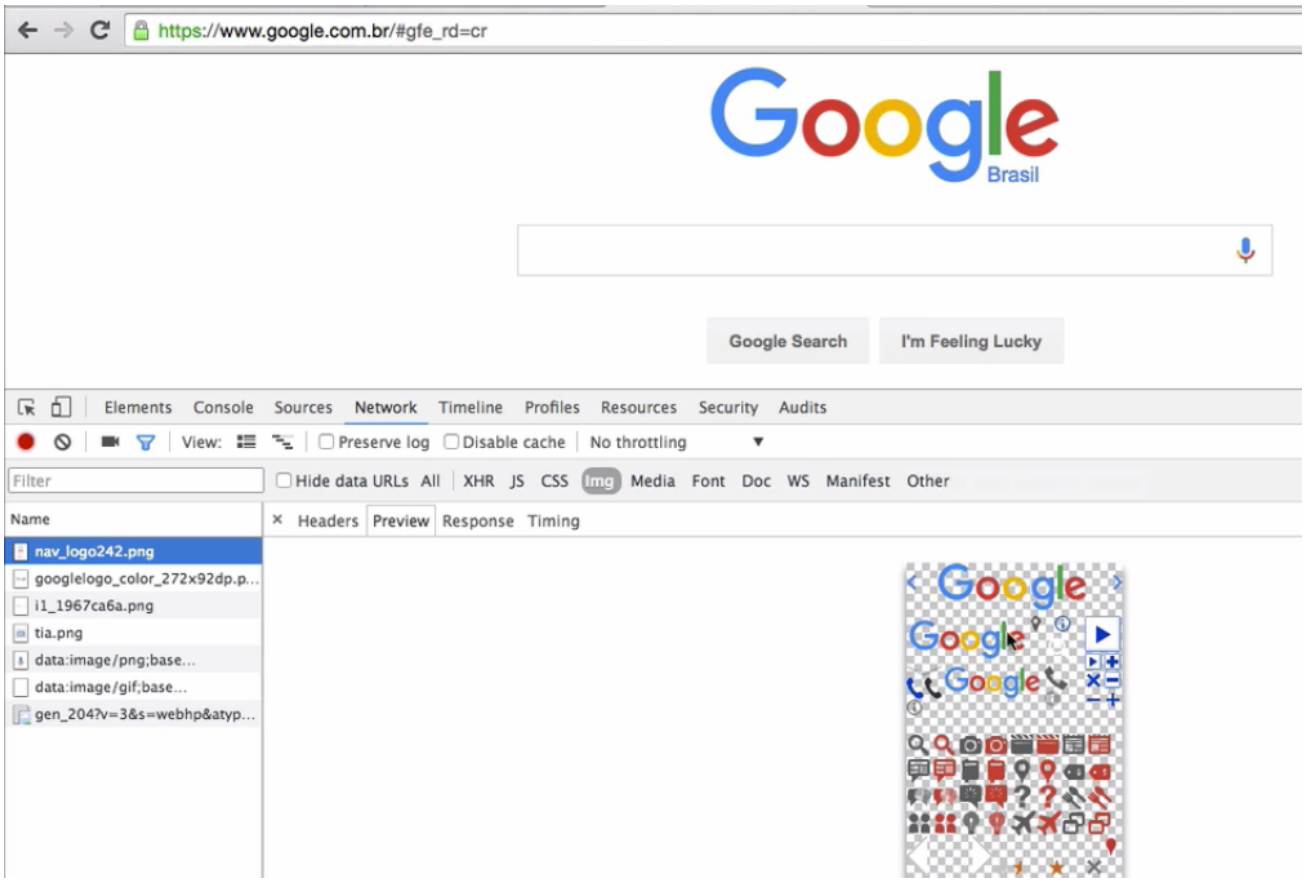
10

Transcrição das aulas

Já realizamos diversas modificações, mas ainda temos diversos aspectos com os quais nos preocuparmos. Temos diversos ícones e imagens em nosso site e todas eles estão em arquivos separados. Vamos observar isso a partir da pasta "img" que está em "Site > assets > img". Nesta pasta temos imagens de diversos tipos, nela temos 53 itens, ou seja, 53 *requests* que são realizados na *home*.

Para evitar o encavalamento, tópico que já analisamos e discutimos anteriormente, uma opção é concatenar as imagens da mesma maneira que fizemos com o *css*. Entretanto, concatenar imagens não é tão simples, mas é possível. A concatenação de imagens também é chamada de *sprite*.

Um ótimo exemplo sobre isso é o site do *Google*. Vamos dar um "F12" e observar a aba *network*, vamos selecionar o *img*. Se clicarmos no primeiro arquivo conseguimos observar o que é o *sprite*:



É uma imagem que contêm diversos ícones. Isso é o que chamamos de *sprite*.

Ao em vez de fazer um *requests* para cada uma dessas imagens que estão contidas nessa única imagem, tudo é juntado em um mesmo *png* e que baixa todos os ícones uma única vez . É interessante perceber que se tivermos vários ícones parecidos em um arquivo só o seu tamanho é menor do que se esses ícones estivessem separados.

E como começamos a fazer isso?

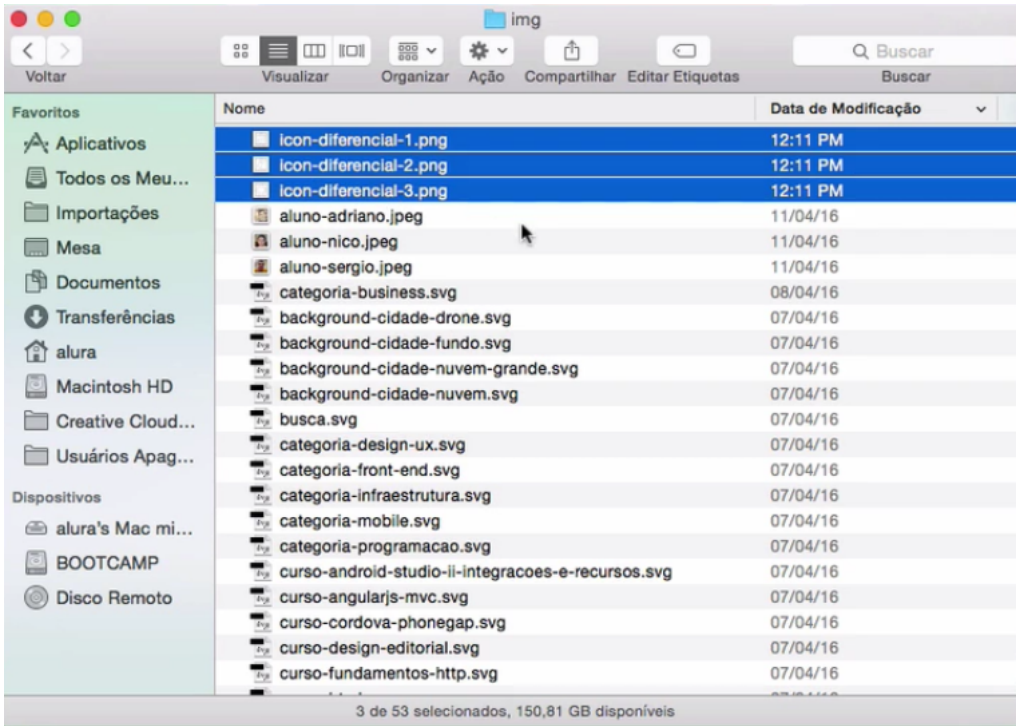
Na realidade, quando observamos um site ele possui sua distribuição de ícones organizada, eles estão em diferentes pontos da página, embora, como no caso do *google*, estejam todos em um arquivo só. Organizamos isso utilizando técnicas de *sprite* com *css*. O que fazemos é redimensionar o *div* e reposicionamos o *background* para exibir apenas um pedaço da imagem. Isto é, temos uma imagem grande de *background*, mas o *div*, como é pequeno, posicionamos ele de

maneira a mostrar apenas o pedacinho do que queremos, por exemplo, uma "setinha", dando a impressão de que temos apenas um ícone.

Vamos fazer isso na prática!

A primeira coisa que precisamos fazer é criar essa imagem e desenhar ela. Você pode fazer isso utilizando o *photoshop* ou, o ideal é automatizar essa tarefa. Uma ferramenta que aconselhamos para automatizar imagens é o *Image Magick*, que está no site www.imagemagick.org/.

A ideia é que podemos passar um comando, usaremos o comando *convert* que é cheio de opções. Ele faz vários tipos de edição de imagem só que na linha de comando. Portanto, ele é muito útil quando se trata de automatização. Vamos instalar o *Image Magick* e como temos vários ícones distintos, vamos tentar criar uma *sprite* para diminuir os *requests*. Vamos pegar três ícones que diferenciam as funcionalidades do Alura:



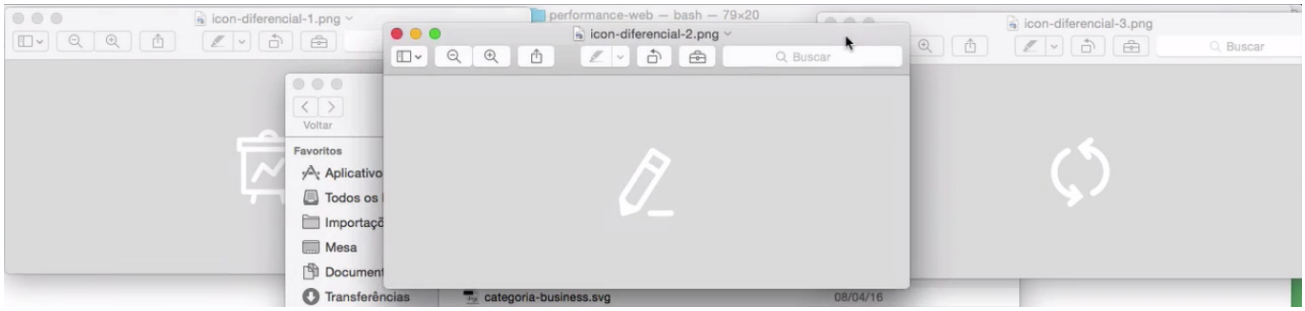
Esses três icones são *png* e possuem os respectivos nomes: icon-diferencial-1, icon-diferencial-2 e icon-diferencial-3.

A ideia é que juntemos os três arquivos em um único arquivo. Vamos para a linha de comando, após instalarmos a ferramenta, e usaremos o comando *convert* e passaremos as imagens que queremos transformar, no caso, todos os *png* da seguinte pasta: `site/assets/img/*.png` e como queremos transformar isso em um único arquivo. Para fazer isso, utilizamos o comando *append*, que pega as imagens e coloca umas em baixo das outras. Por fim, jogamos a imagem final na pasta `site/assets/img/diferenciais.png` . Teremos na linha de comando o seguinte:

```
convert site/assets/img/*.png -append site/assets/img/diferenciais.png
```

Damos um "Enter" nisso e veremos se funcionou.

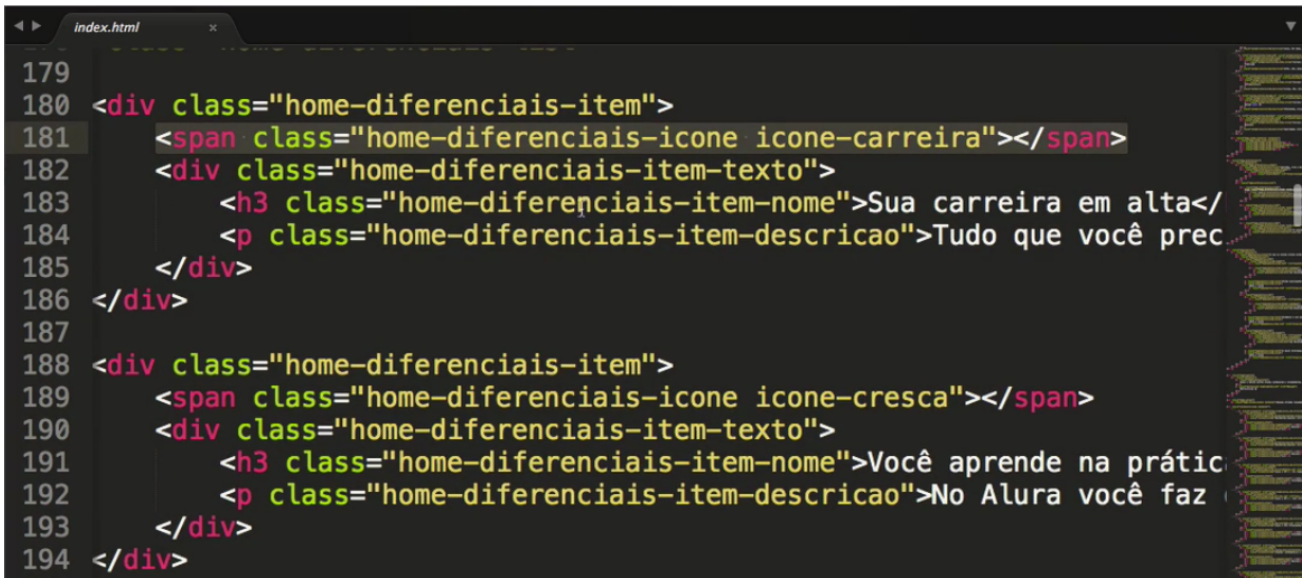
Bom, os ícones que queremos juntar são os seguintes:



Geramos um diferencial *png* que possui os três ícones em uma mesma imagem e um em baixo do outro, essa distribuição horizontal é o padrão, mas a orientação pode ser modificada. Repare que cada um dos *png* separados possuíam 3 KB cada enquanto nossa imagem com os três possui 4 KB. Ainda, otimizaremos essa imagem tornando ela ainda menor.

E agora, como usamos esse arquivo com as três imagens no nosso site?

Vamos observar o código fonte para perceber como ele é exibido. Vamos olhar o *html* e na parte dos diferenciais temos basicamente que cada um dos ícones é um *span* e cada um deles está nomeado de uma maneira distinta:



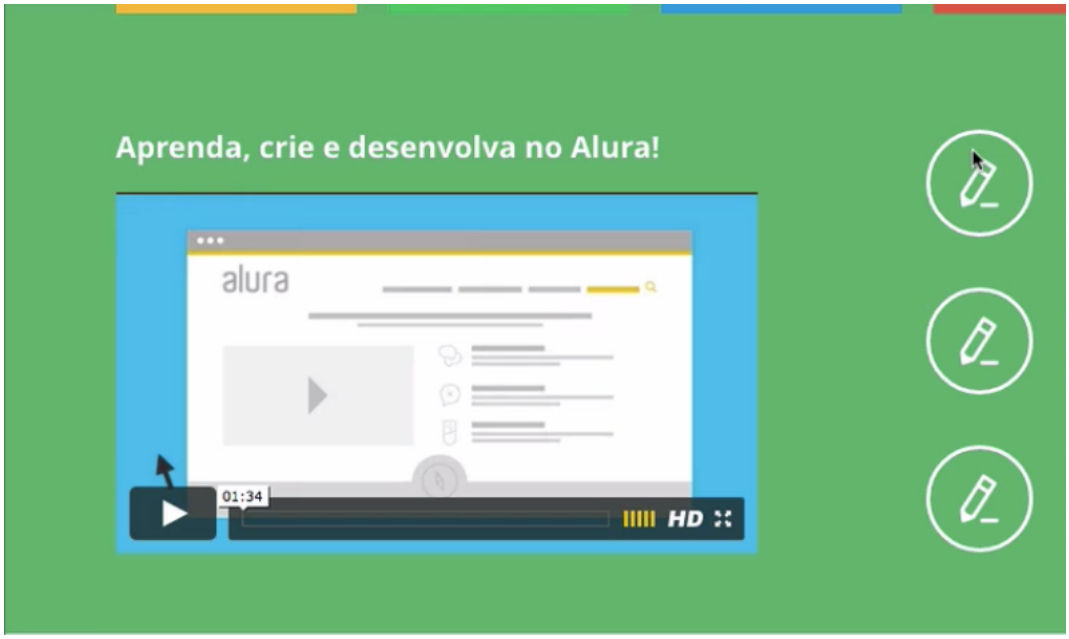
Se é um *span* isso significa que não tem nada dentro dele e estamos usando isso dentro do *css*. Abrimos o arquivo "home-diferenciais.css" e localizamos a parte que fala dos ícones:

Nessa parte estão descritas as regras das imagens, as bordas arredondadas e outros detalhes... O que queremos mostrar é que cada um daqueles ícones estão sendo diferenciados com um *background image* diferente.

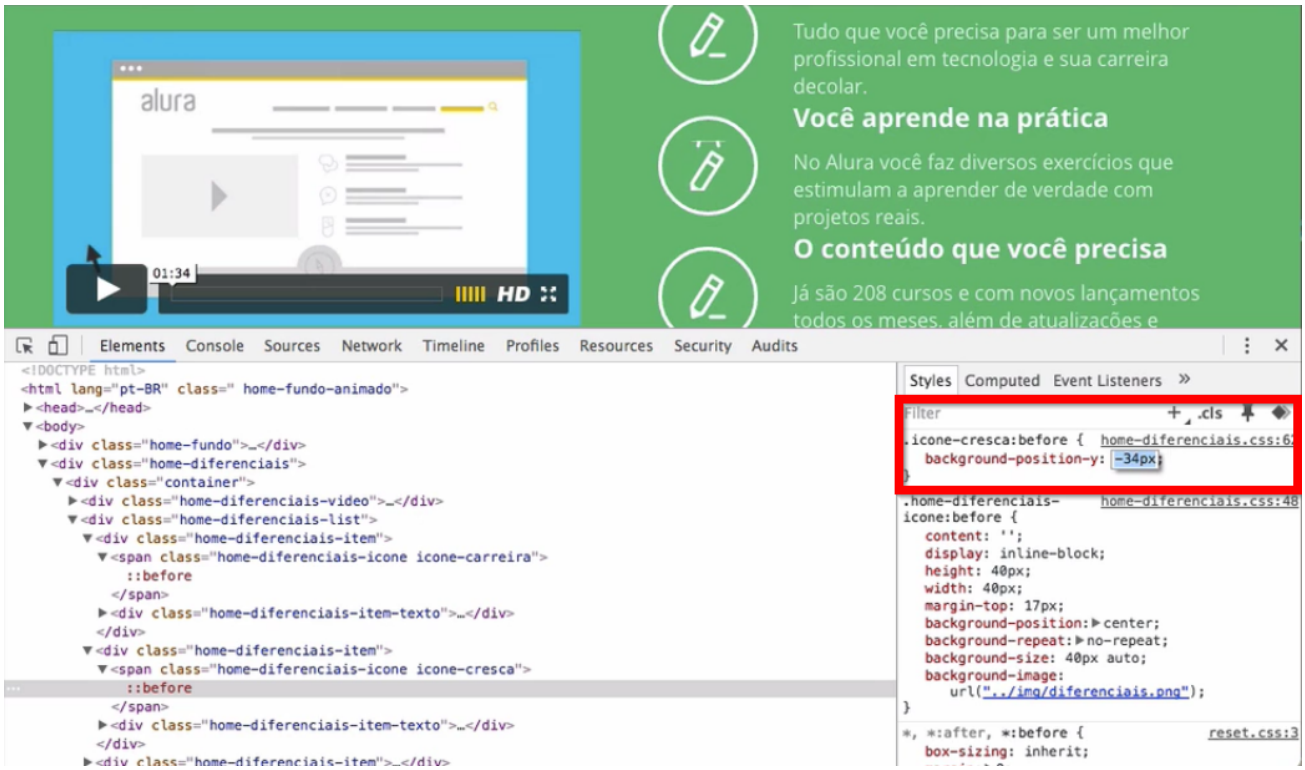
O que vamos fazer?

Vamos parar de usar os arquivos diferentes e vamos utilizar o arquivo novo e único, vamos alterar, então, o nome do arquivo para `../img/diferenciais.png` e colocar isso para cima e apagar os nomes que diferenciavam as imagens.

A ideia é que vamos usar o mesmo *background*, um único arquivo, nos três ícones. Vamos dar um *refrash* para ver o que acontece:



Vemos que temos o mesmo item nos três espaços que eram reservados para ícones diferentes. Vamos dar um "F12" e ficar na aba *Elements*. O que acontece é que o arquivo "diferenciais.png" é muito longo e como ele tem três ícones o que ocorre é que vemos um ícone apenas de cada vez. O que queremos, na verdade, é que em cada uma das opções seja mostrado um ícone diferente. Fazemos isso deslocando a imagem do *background*, da *sprite*, isto é, deslocamos alguns pixels. Vamos observar o terminal, por exemplo, podemos pegar `home-diferenciais.css` e colocar um `background-position` e através de um `-y` podemos controlar quantos pixels queremos mostrar. Perceba que conforme vamos alterando os `pixels`, estamos alterando qual o item que ocupará esse espaço:

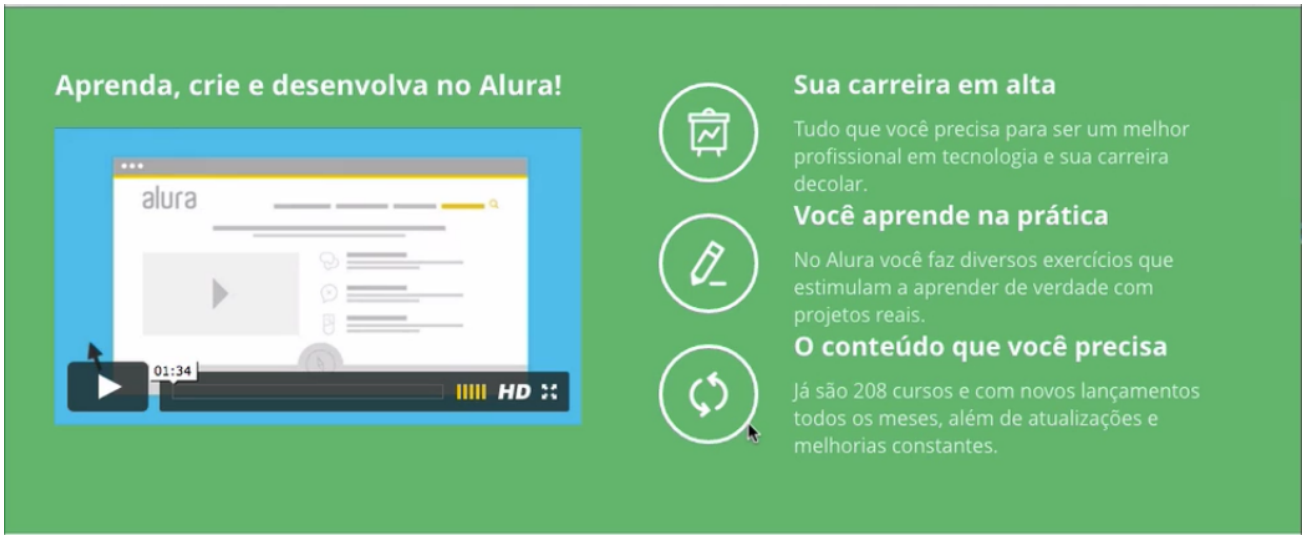


Se soubermos o tamanho do ícone basta irmos mexendo que conseguimos exibir um ícone ou outro. Se queremos exibir o ícone que está mais na ponta temos que acrescentar `-80 px` porque é menos os outros dois - cada um tem "40 pixels" - e assim por diante. Podemos utilizar, ainda, alguns atalhos: `top` para fazer referência ao primeiro ícone, `button` para mostrar o último e `center` para mostrar o do meio. Ficaremos, por exemplo, com `background-positions-y: top`. Se tivermos muitos ícones, entretanto, esse atalho não funciona, teremos que usar as coordenadas.

O que vamos fazer agora? Voltamos ao terminal e no arquivo `home-diferencias.css` vamos diferenciar os nossos ícones:

```
.icone-creca:before {
    background-position: center;
}
.icone-carreira:before {
    background-position: top;
}
.icone-impacto:before {
    background-position: bottom;
}
```

Então, diferenciamos as posições de cada ícone. Vamos atualizar e reparar como está nosso site:



Temos, finalmente, os três ícones que queríamos. Fizemos esse truque de juntar as imagens em um único arquivo e via *css* posicionamos para exibir apenas o ícones da vez, dando a impressão de que temos três arquivos separados, quando na verdade temos apenas um. Essa técnica que estamos utilizando é via *css* e se chama *sprite*. É muito interessante para economizarmos *requests*.

Bom, antes de encerrar vamos fazer uma discussão rápida. Viemos na pasta "site" original, editamos o *css* e geramos a imagem na pasta "site", ou seja, estamos manipulando tudo direto na pasta "site", a original, ao em vez de fazermos isso na pasta "dist", de produção.

Aqui, existe muita controvérsia, há pessoas que acreditam que deveríamos fazer o site original com os arquivos separados e no site final é que geraríamos a *sprite*. Ou seja, deveríamos ter algum tipo de automação para gerar a *sprite*.

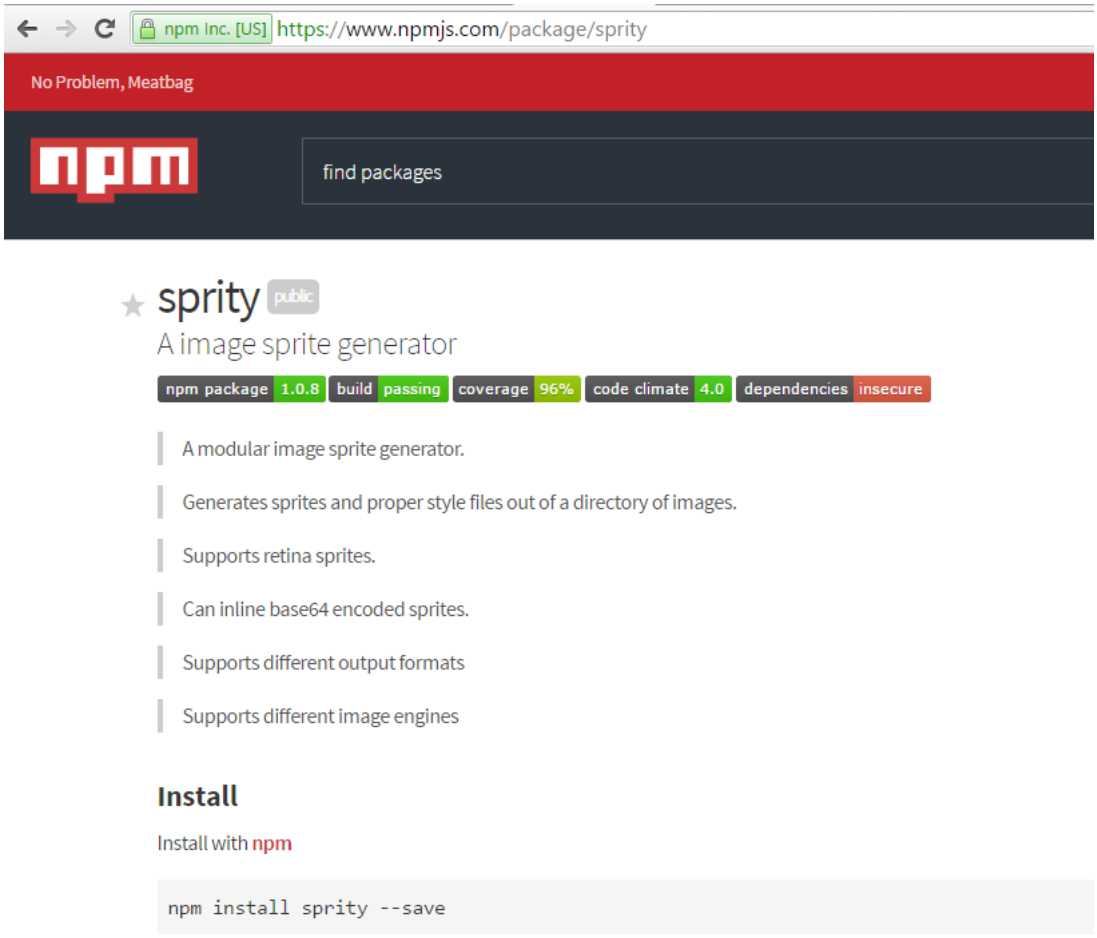
Tem pessoas que preferem manipular as *sprites* no arquivo original e fazem isso porque envolve uma série de *css* que podem ser um pouco complicados quando são gerados. É uma preferência escrever o *css* e saber que ele estará funcionando. Não quer dizer que esta imagem será desenhada no *photoshop* a mão, pois, foi apresentado, anteriormente, como gerar essa image. Esse método é um gosto pessoal, tem pessoas que preferem gerar o *css* final e para isso será necessário utilizar algumas ferramentas, como *pluggins* do brunch, para o *gulp*.

Por exemplo, o *Spritesmith* que está no link (https://github.com/Ensighten/spritesmith)https://github.com/Ensighten/spritesmith (https://github.com/Ensighten/spritesmith) e ele gera uma série de opções. Vejamos a página:

Tem que cuidar para que o *css* gerado não entre em conflito com os demais e que ele seja inserido no local correto. As vezes, não é lá tão prático assim.

Outro *pluggin* é o *Sprity* que está no link (http://www.npmjs.com/package/sprity)https://www.npmjs.com/package/sprity (https://www.npmjs.com/package/sprity) que é um módulo do *npm* que usamos com *gulp* e podemos utilizar também na linha de comando e etc. Vamos observar como funciona isso. Na linha de comando basta utilizar o comando `create` e informamos onde queremos criar as imagens e quais as imagens que queremos usar.

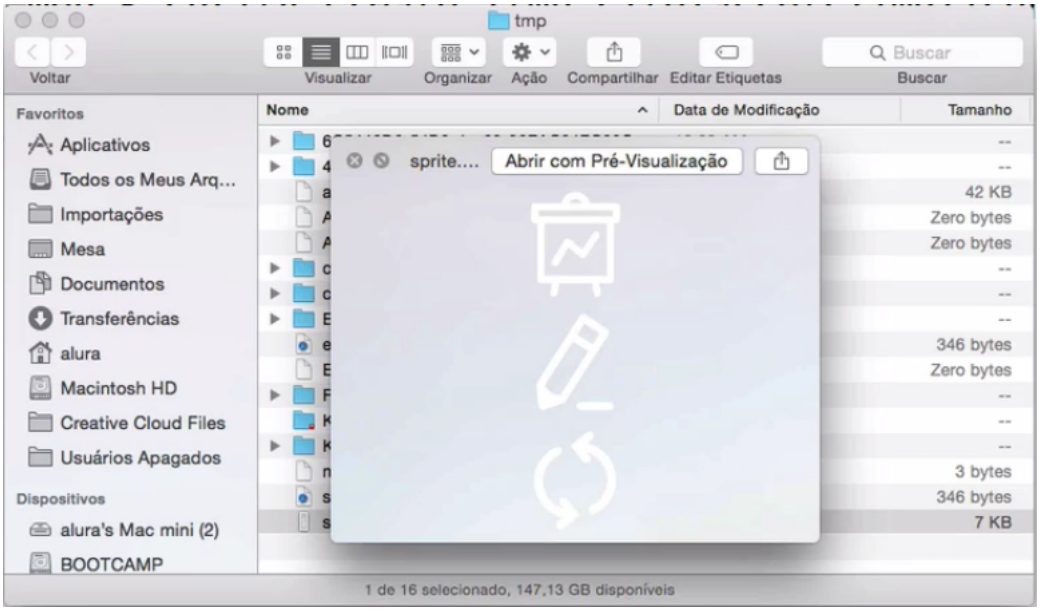
Essa é a página dessa ferramenta:



E para usar esse comando na linha de comando basta digitar:

```
sprity create /tmp site/assets/img/icon-*.png .
```

E ele gera praticamente a mesma imagem que tínhamos antes:



Podemos passar também um `-s` que ele é capaz de gerar todos os *css* que queremos com as posições certas, para fazermos o *position*. Mas, se repararmos esse *css* parece, ainda, um pouco estranho, cheios de *icons*:

performance-web — bash — 79×20

bash bash

```
.icon {
  background-image: url('../images/sprite.png');
}

.icon-icon-diferencial-1 {
  background-position: -0px -0px;
  width: 88px;
  height: 88px;
}
.icon-icon-diferencial-2 {
  background-position: -0px -88px;
  width: 88px;
  height: 88px;
}
.icon-icon-diferencial-3 {
  background-position: -0px -176px;
  width: 88px;
  height: 88px;
}
performance-web $
```

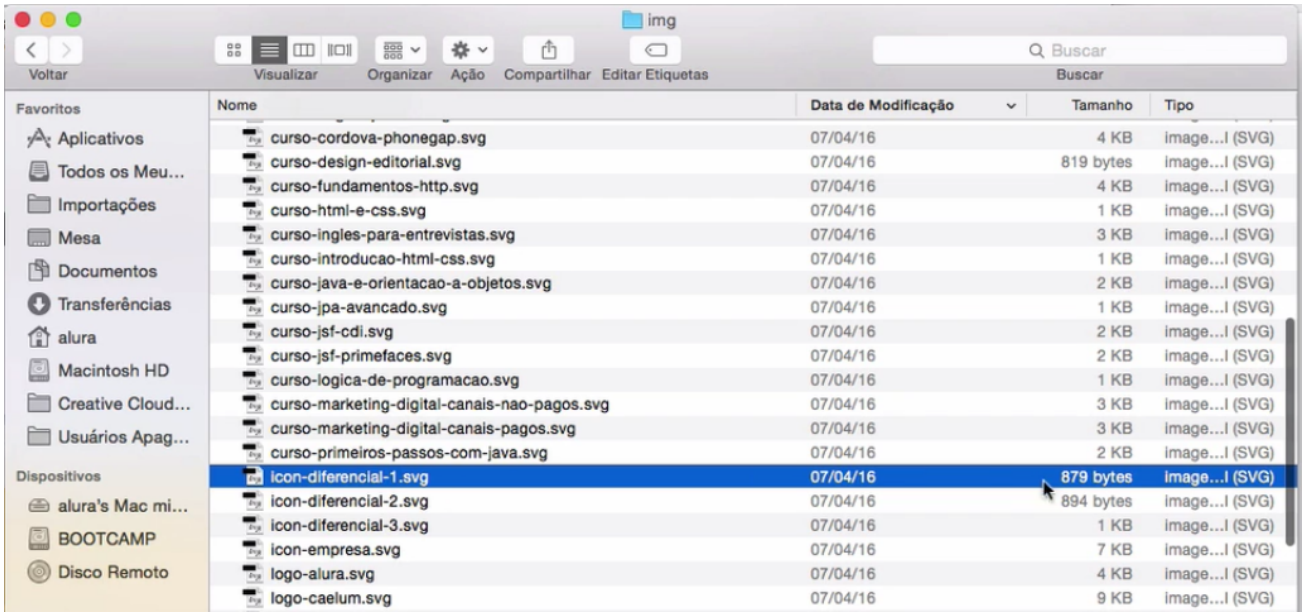
Bom, este é um código gerado que pode facilitar o trabalho.

Bom, existem esses dois meios de gerar as imagens. Uma mais manual, onde primeiro a imagem é gerada e depois o *css* é configurado corretamente. E a segunda que foi apresentada utilizando essas ferramentas.

Nesse nosso caso é preferível a maneira manual de fazer essa tarefa.

É claro que em casos de imagens muito grandes como foi o caso da imagem do *google*, essas ferramentas podem, efetivamente, ajudar.

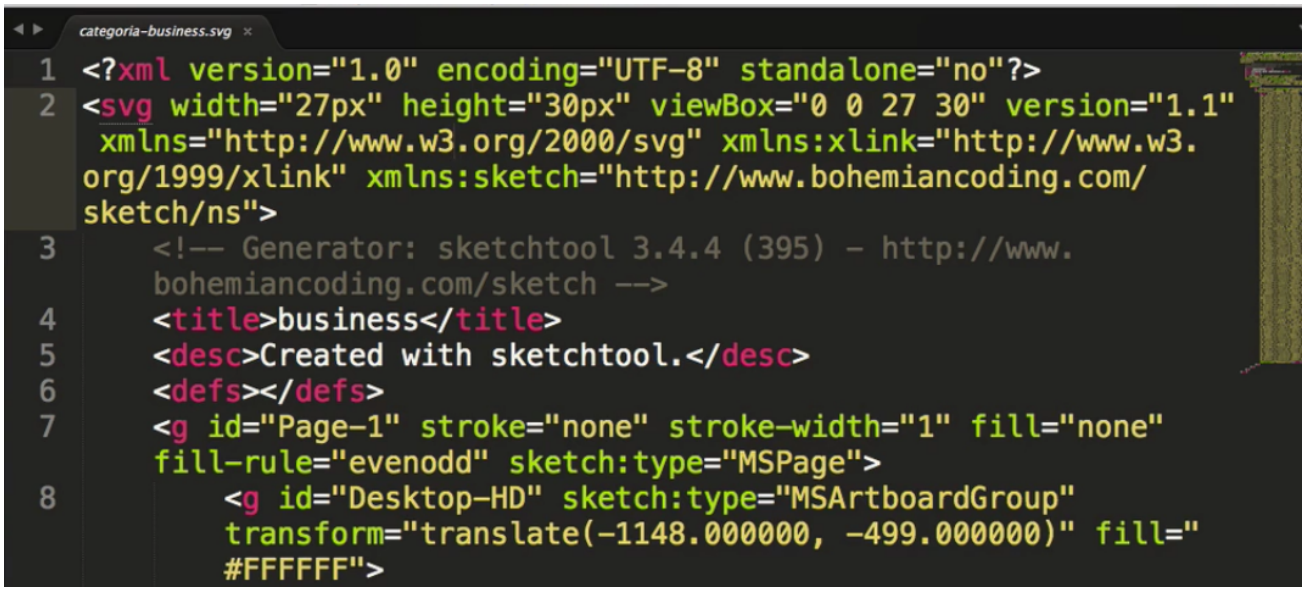
Mostramos anteriormente como lidar com imagens e ícones utilizando o mecanismo de *sprite*. Em nosso caso temos uma maior parte das imagens em *svg*. Se repararamos no projeto, todos os ícones, praticamente, são em *svg*:



O *svg* é um formato que fica melhor em *mobile*, retina e etc e ele também é um formato menor. Então, devido a essas diversas vantagens, existe uma preferência por esse tipo de formato.

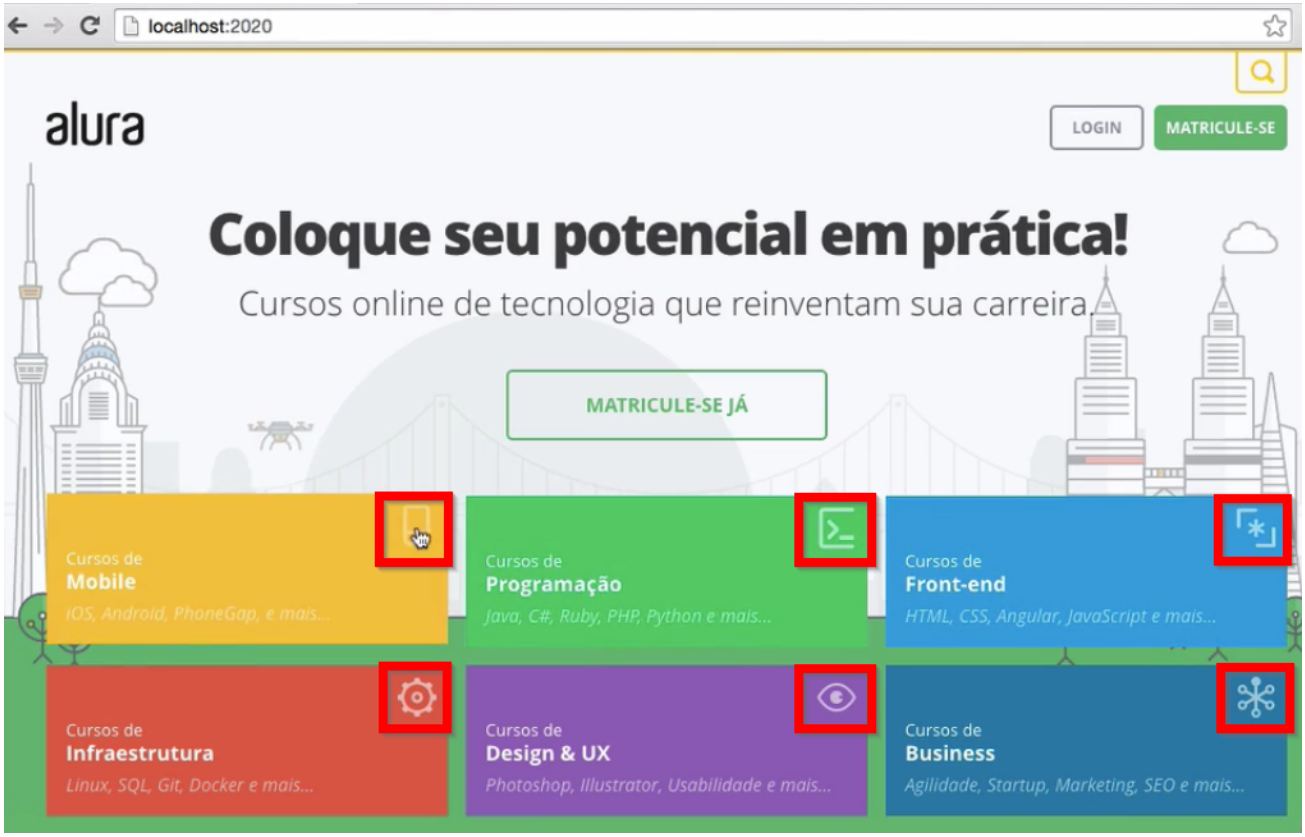
Mas, teremos o mesmo problema que antes, isto é, uma grande quantidade de arquivos. Poderíamos resolver esse problema agrupando as imagens em um mesmo arquivo e ajustando sua colocação e etc, mas existem outras maneiras de lidar com essa situação. Podemos lidar com o problema utilizando a ideia de *sprite*, só que diferente.

Quando fazemos o *sprite* na mão temos que reposicionar o *css* e isso pode ser um tanto trabalhoso. Com o *svg* temos outras possibilidades. Antes de explorar isso vamos observar o funcionamento do *categoria-business.svg* na prática. Vamos abrir um *svg* e vamos observá-lo, repare que ele nada mais é do que um *xml*. Ele possui diversos comandos que mostram como esse *svg* vai ser desenhado.



Estamos olhando isso na pasta "site", aquela que não está otimizada. Se observarmos o mesmo arquivo *svg* na pasta "dist" veremos que ele estará mais otimizado, os ícones estão menores.

Vamos conferir no *html* como os *svg* estão sendo chamados. E vamos pegar a parte das categorias. Repare que cada uma das seis categorias do site possui um ícone que é *svg*. Os ícones estão enquadrados de vermelho:



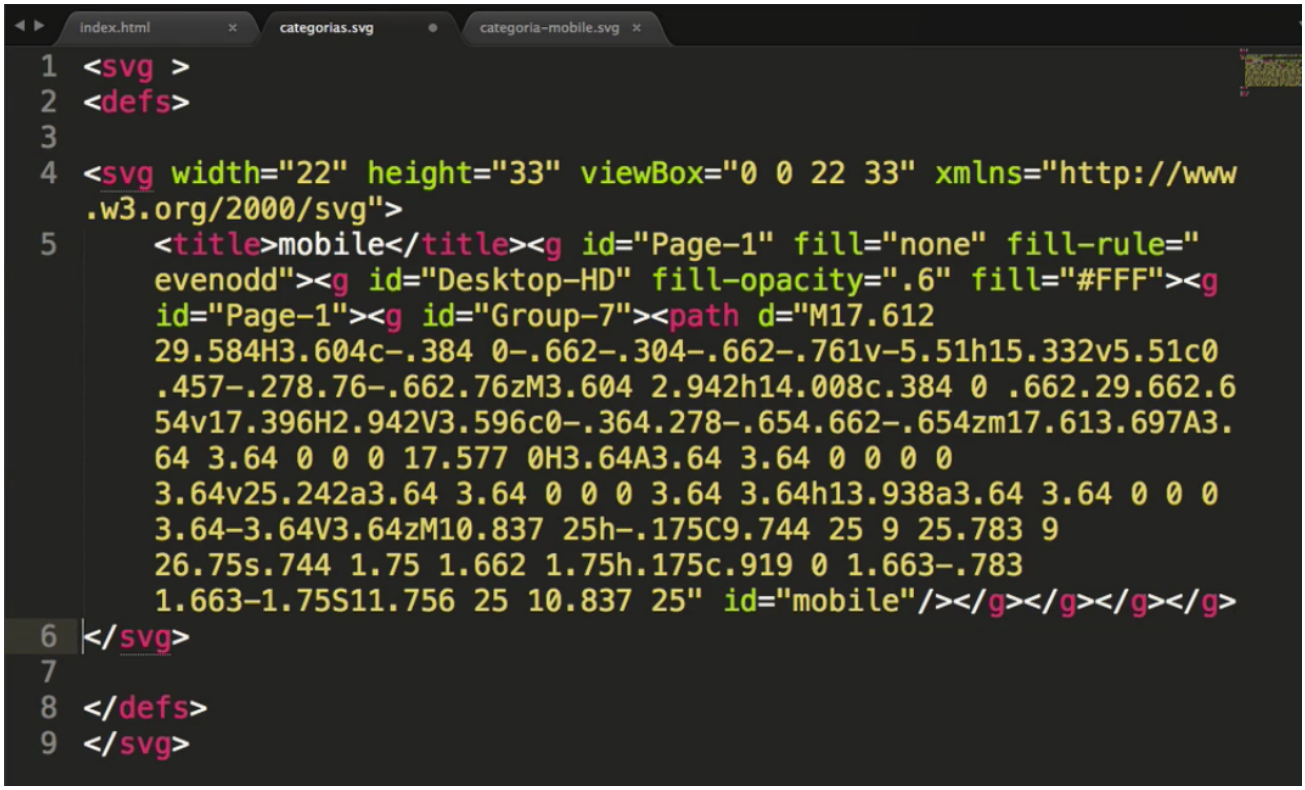
E como carregamos isso?

Vamos reparar no ícone do *Mobile*. Ele tem *img*, uma classe qualquer, *source* e categoria. E esse padrão se repete em todos os seis ícones. Imagine que queremos criar um *request* para que nele estejam os seis ícones.

Vamos fazer isso "à mão". Vamos na pasta "img" e criaremos um novo arquivo que será o "categorias" e para visualizar que não é tão difícil fazer isso.

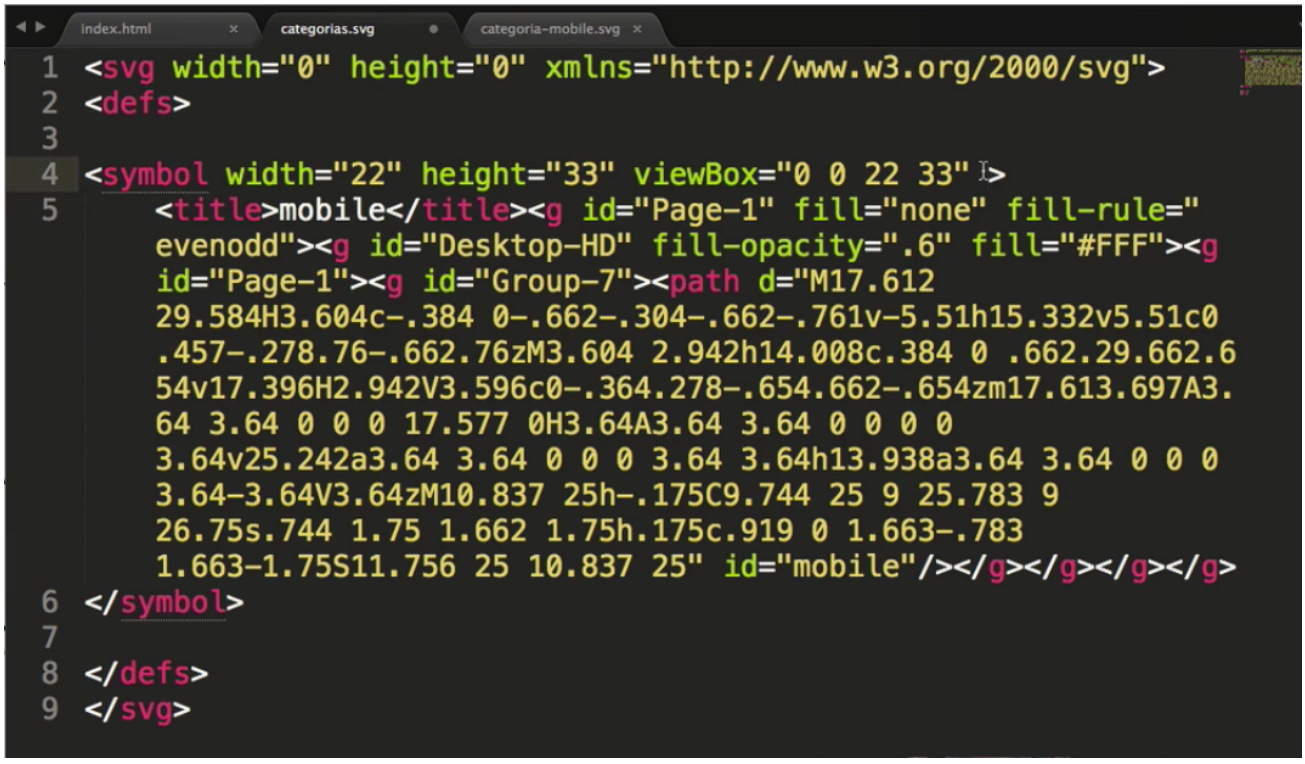
Vamos pegar a versão já minimizadas das categorias que fica mais fácil de vermos. Temos um ícone "categoria-mobile" e queremos colocar ele dentro do *svg*. Primeiro, criaremos um novo arquivo *svg*. Dentro dele, teremos uma tag chamada `defs` de *definition* e dentro dele teremos o que chamamos de símbolos que são os ícones. Dentro do `defs` colaremos os *svg* do *mobile*.

Vamos quebrar com espaços esse código para podermos entender o que temos nele:



Vamos trocar a *tag svg* por uma chamada *symbol*. A ideia é que dentro do *defs* tenhamos vários *symbol*. Como acabamos de alterar o nome, vamos apagar o *xmlns* dele, pois não é mais necessário. E movemos o *xmlns* para o *svg* restante, que é a primeira *tag* do nosso arquivo.

Vamos, ainda, acrescentar depois da *tag svg* `width="0" height="0"` com isso estamos dizendo que não queremos que a imagem seja exibida e sim que sejam mostrados os símbolos individualmente.



Repare, é o mesmo código, só trocamos para *symbol*. E podemos introduzir diversas coisas aqui.

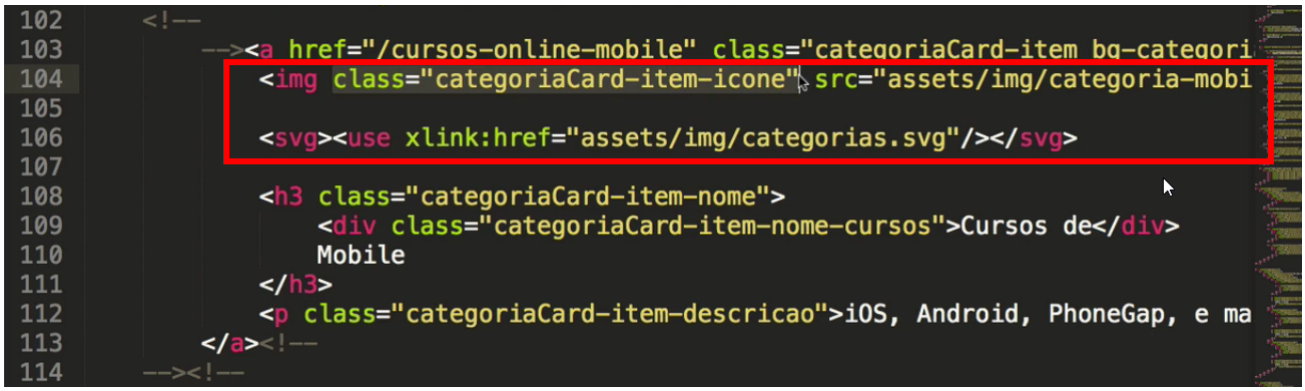
A ideia é que coloquemos as seis categorias.

E como usamos isso?

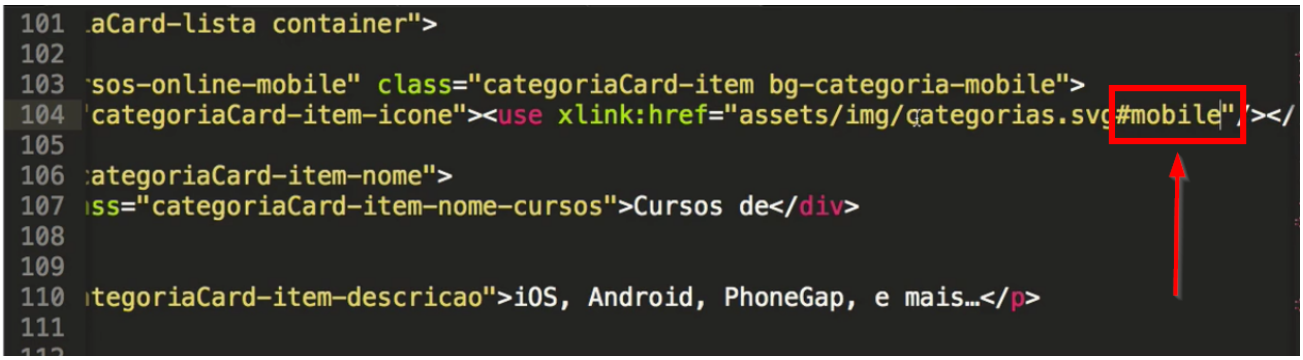
Vamos no *html* e para usar um *svg* dessa maneira precisamos usar a própria *tag svg*. Usaremos a *use* que é uma *tag* que recebe um *href* onde podemos passar o nome do *svg* que possui todos os ícones.

```
<svg><use xlink:href="assets/img/categorias.svg"/></svg>
```

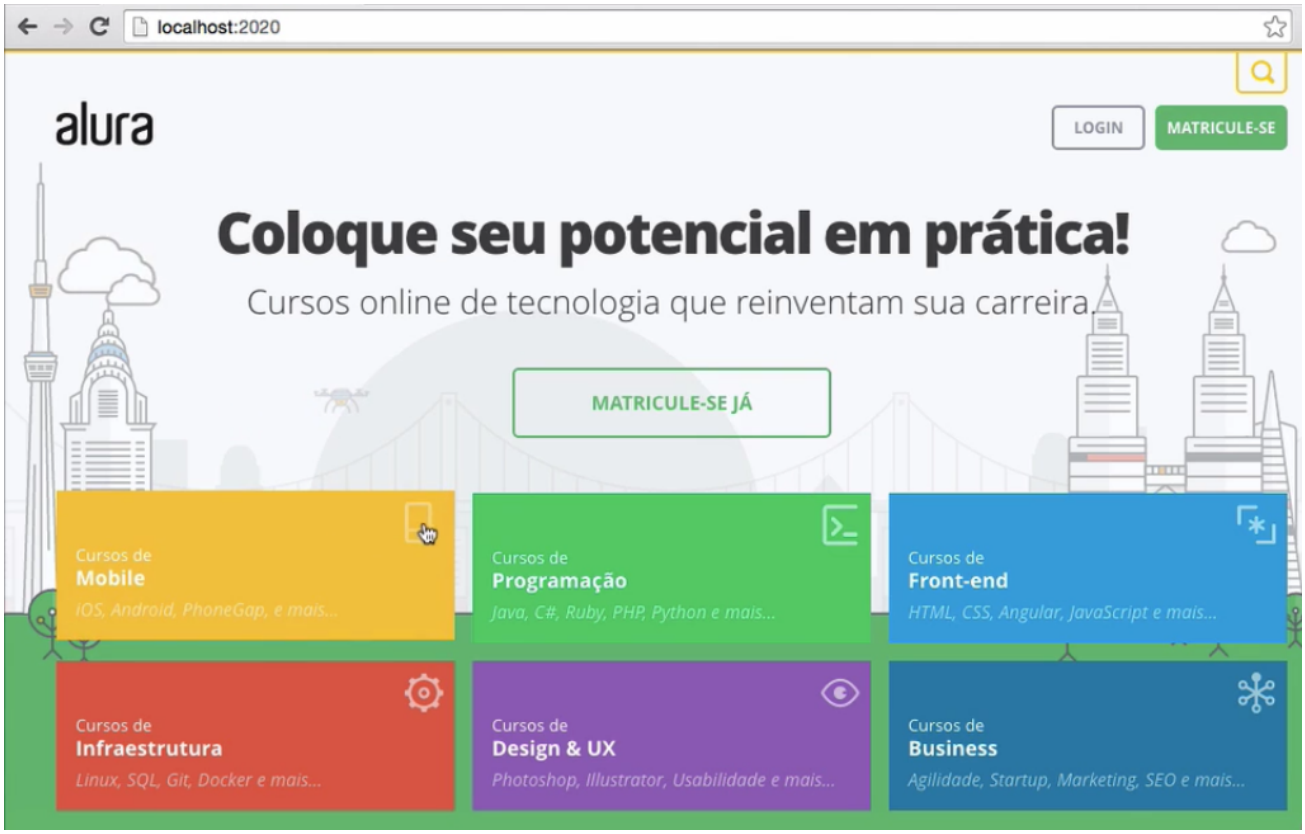
Note que na imagem abaixo temos o que fazíamos e o que estamos fazendo agora:



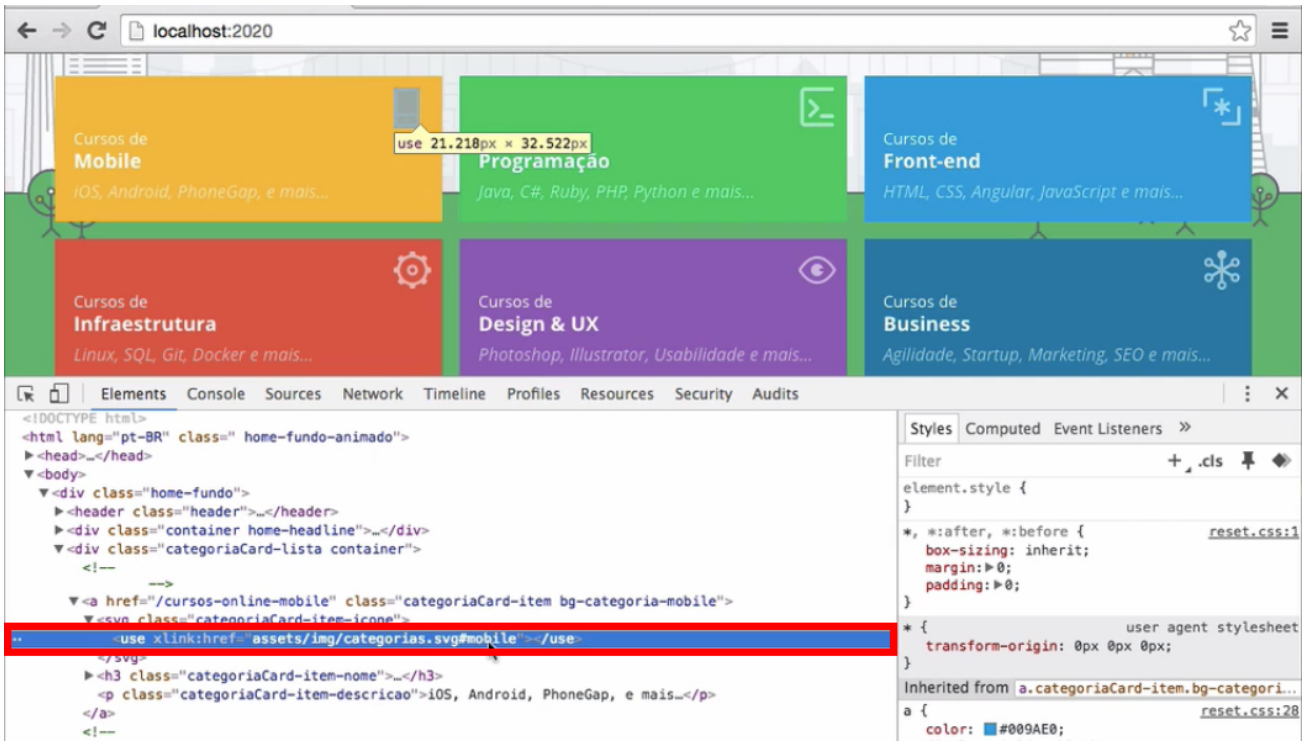
E agora, como ele sabe que ícone carregar? A ideia é que no categoria *svg* cada um dos símbolos que definimos pode ter um *id* que os diferencie, então, nomearemos o primeiro com o `id="mobile"`, `id="front-end"` e podemos continuar nomeando. E nesse momento, podemos ainda, no "index.html" adicionar no final do `use`, `#mobile`. Com isso estamos dizendo que queremos que isso seja carregado e lá dentro deve ser buscado o símbolo que contêm `#mobile`.



Vamos ver se funciona? Vamos atualizar nosso navegador e, aparentemente, constatamos que tudo está em ordem:



Podemos dar um *inspector* e confirmar que tudo está funcionando nos conformes:



E a mesma coisa se repete para os próximos itens, podemos fazer o mesmo com o `front-end` adicionando `#front-end`.

Repare, que fazendo isso estamos economizando *requests*. A criação desse arquivo é muito simples, a ideia é juntar os arquivos e usar a tag *symbol*.

E o que fazemos? Mantemos os ícones separados e escrevemos na mão?

Podemos fazer uma geração automática utilizando algumas ferramentas. Podemos procurar no *google*, o *svg sprite*. Acessaremos o link (https://github.com/jkphl/svg-sprite)https://github.com/jkphl/svg-sprite

(https://github.com/jkphl/svg-sprite). Ele mostra diversas configuração e opções, podemos fazer isso automaticamente, temos, inclusive, um *pluggin* para o *gulp*. Ou, podemos criar isso nos próprios editores que existem para *svg* que suportam exportar direto para a sintaxe do *symbol*. Por exemplo, podemos utilizar o *Inkscape* para desenhar ícones e também podemos utilizá-lo para visualmente rotulá-lo de símbolo, assim, ele já gera a sintaxe automaticamente, sem a dificuldade de fazer isso a mão.

Então, temos essas opções: (1) usamos um gerador qualquer; (2) fazemos isso a mão (3) usamos um editor visual;