02

Combatendo o Prefix HELL!

Começando deste ponto? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/gulp/stages/06-capitulo.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo. Dentro da pasta **projeto**, não esqueça de executar no terminal o comando **npm install** para baixar novamente todas as dependências.

Nosso script de `gulpfile.js` cada vez nos torna mais produtivos. Será que está tudo pronto, falta alguma coisa? Vamos dar uma passeada pelos arquivos do projeto, que tal começarmos abrindo o `projeto/src/css/estilos.css` ?

Bem, CSS aparentemente normal, mas espere pouco. Vejamos a seguinte propriedade CSS:

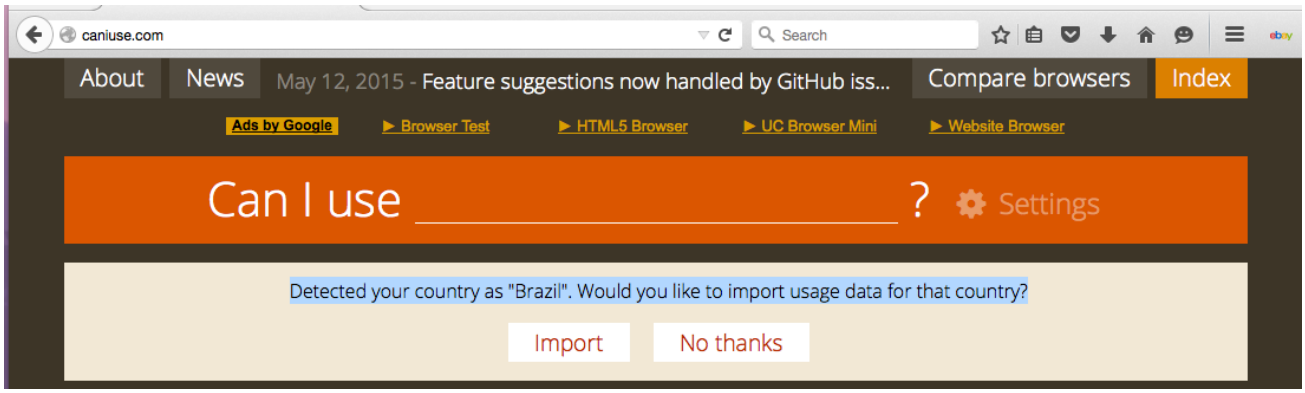
estilos.css

```
.painel li:hover {
  box-shadow: 0 0 5px #333;
  transition: box-shadow 0.7s;
}

.painel li:hover {
  background-color: rgba(255,255,255,0.8);
  transition: all 0.7s;
}

.painel li:hover {
  transform: scale(1.2);
}

.painel li:hover {
  transform: scale(1.2) rotate(-5deg);
}

.painel li:nth-child(2n):hover {
  transform: scale(1.2) rotate(5deg);
}

.painel li {
  transition: all 0.7s;
}

.painel li:hover {
  transition: all 0.7s ease-in;
}
.painel li {
  transition: all 0.7s ease-out;
}
```

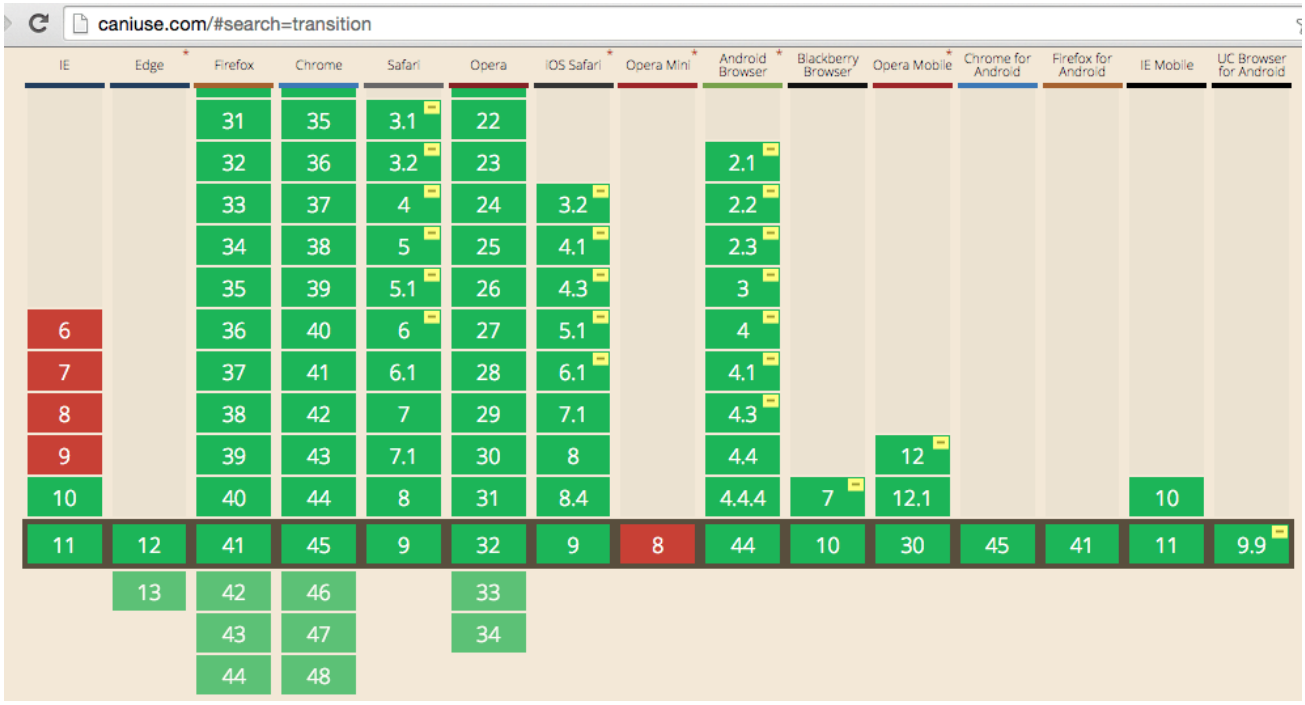
Esse trecho de código é o que anima os produtos usando a propriedade **transition** e **transform** do CSS3. Mas o que tem de errado? Bem, uma consulta ao site **http://www.caniuse.com (http://www.caniuse.com)** pode lançar uma luz no problema.

Propriedades prefixadas

No site do `http://www.caniuse.com` podemos pesquisar por propriedades CSS e seu suporte nos mais diversos navegadores. Porém, sempre que o acessamos ele pergunta se queremos importar as estatísticas de navegadores do nosso país (detectado automaticamente). É uma boa escolha clicar em `importar` :



Depois desse processo, podemos pesquisar por propriedades CSS, em nosso caso, buscaremos por `transition` . Será apresentada uma tabela que mostrará se a propriedade não é suportada, suportada ou parcialmente suportada pelos navegadores mais usados do mercado:



Podemos saber ainda mais clicando em `Show All` , que levará em consideração versões ainda mais antigas do navegador. Ainda sobre a propriedade `transition` , vemos que até o chrome 25 é necessário usar o prefixo `-webkit` e que no android é necessário usar o mesmo prefixo até a versão 4.3. Há mais detalhes de outros navegadores, mas que raios de prefixo é esse?

No passado (porque hoje essa prática é evitada), quando uma propriedade nova era lançada, muitos navegadores a suportavam experimentalmente, isto é, a propriedade, mesmo em fase de teste, poderia ser usada. Porém, essas propriedades experimentais não são habilitadas por padrão, sendo necessário utilizar um prefixo que difere de navegador para navegador.

Vejamos a propriedade transition prefixada:

```
.painel li:hover {
  background-color: rgba(255,255,255,0.8);

  -webkit-transition: all 0.7s;
     -moz-transition: all 0.7s;
      -ms-transition: all 0.7s;
       -o-transition: all 0.7s;
          transition: all 0.7s;
}
```

Usamos `-webkit` para Chrome, Android e versões mais atualizadas do opera. Usamos `-moz` para o Firefox, `-ms` para Internet Explorer e `-o` para versões mais antigas do Opera. No exemplo acima, se estivermos em uma versão antiga do Chrome, como por exemplo a versão 21, todas as outras propriedades serão ignoradas, exceto a prefixada. O interessante é que se a propriedade deixar de ser experimental a partir de uma versão específica, apenas a propriedade não prefixada será aplicada.

Prefixar e repetir, eis a questão!

A ideia é utilizarmos a prefixação para ampliarmos o número de navegadores que suportam a propriedade. Há aqueles que suportam apenas os dois últimos navegadores do mercado (aliás, essa é a política da Google), há outros que suportam os quatro últimos.

Imagine prefixar o restante das propriedades do nosso arquivo CSS? Há propriedades também como `box-shadow` que seria interessante prefixar. Com certeza teremos muito código duplicado e se precisarmos alterar algum valor de determinada propriedade teremos que alterar em todas as propriedades prefixadas. Será que tem solução?

Automatizar, prefixar e aproveitar!

Podemos automatizar esse processo de prefixação no momento de criação do projeto para deploy. O plugin gulp-autoprefixer (https://github.com/sindresorhus/gulp-autoprefixer) pode nos ajudar. Instalando-o através do terminal:

```
npm install gulp-autoprefixer@3.0.2 --save-dev
```

E como de costume, vamos importar o módulo em nosso `gulpfile.js` :

```
var gulp = require('gulp')
  ,imagemin = require('gulp-imagemin')
  ,clean = require('gulp-clean')
  ,concat = require('gulp-concat')
  ,htmlReplace = require('gulp-html-replace')
  ,uglify = require('gulp-uglify')
  ,usemin = require('gulp-usemin')
  ,cssmin = require('gulp-cssmin')
  ,browserSync = require('browser-sync')
  ,jshint = require('gulp-jshint')
  ,jshintStylish = require('jshint-stylish')
  ,csslint = require('gulp-csslint')
  ,autoprefixer = require('gulp-autoprefixer');
```

Integrando com tarefas existentes

A primeira coisa que podemos pensar é na criação de uma tarefa, mas podemos fazer com que o `usemin` chame o `autoprefixer` colocando-o no array de módulos da propriedade `css` :

```
// código anterior omitido
gulp.task('usemin', function() {
  return gulp.src('dist/**/*.html')
    .pipe(usemin({
      js: [uglify],
      css: [autoprefixer, cssmin]
    }))
    .pipe(gulp.dest('dist'));
});
// código posterior omitido.
```

Agora precisamos testar. Vamos rodar nossa tarefa padrão no terminal:

```
npm run gulp
```

Vamos dar uma olhada em `projeto/dist/css/index.min.css` . Teremos um pouco de dificuldade em ver o trabalho do `autoprefixer` , porque nosso arquivo está minificado. Então, temporariamente, vamos remover a chamada do `cssmin` deixando apenas o `autoprefixer` :

```
// código anterior omitido
gulp.task('usemin', function() {
  return gulp.src('dist/**/*.html')
    .pipe(usemin({
      js: [uglify],
      css: [autoprefixer]
    }))
    .pipe(gulp.dest('dist'));
});
// código posterior omitido
```

Agora sim, abrindo nosso arquivo `projeto/dist/css/index.min.css` que não está minificado, vemos as propriedades como:

projeto/dist/css/index.min.css

```
.painel li:hover {
  box-shadow: 0 0 5px #333;
  -webkit-transition: box-shadow 0.7s;
          transition: box-shadow 0.7s;
}

.painel li:hover {
  background-color: rgba(255,255,255,0.8);
  -webkit-transition: all 0.7s;
          transition: all 0.7s;
}
```

```
.painel li:hover {
  -webkit-transform: scale(1.2);
      -ms-transform: scale(1.2);
          transform: scale(1.2);
}

.painel li:hover {
  -webkit-transform: scale(1.2) rotate(-5deg);
      -ms-transform: scale(1.2) rotate(-5deg);
          transform: scale(1.2) rotate(-5deg);
}

.painel li:nth-child(2n):hover {
  -webkit-transform: scale(1.2) rotate(5deg);
      -ms-transform: scale(1.2) rotate(5deg);
          transform: scale(1.2) rotate(5deg);
}

.painel li {
  -webkit-transition: all 0.7s;
          transition: all 0.7s;
}

.painel li:hover {
  -webkit-transition: all 0.7s ease-in;
          transition: all 0.7s ease-in;
}
.painel li {
  -webkit-transition: all 0.7s ease-out;
          transition: all 0.7s ease-out;
}
```

Suportando browsers mais antigos

Não há prefixação do `Opera` no resultado final. Podemos indicar para o `autoprefixer` que ele deve considerar os 4 últimos navegadores. Para isso, precisamos criar o arquivo **browserslist** em `projeto` . Ele terá uma linha apenas e seremos radicais: vamos pedir suporte aos 40 últimos browsers!

browserslist

```
Last 40 versions
```

O conteúdo é esse mesmo, parece uma frase em inglês e é uma frase!

Agora, basta rodarmos novamente. Em nosso `index.min.css` final teremos:

```
.painel li:hover {
  -webkit-box-shadow: 0 0 5px #333;
     -moz-box-shadow: 0 0 5px #333;
          box-shadow: 0 0 5px #333;
  -webkit-transition: -webkit-box-shadow 0.7s;
     -moz-transition: -moz-box-shadow 0.7s;
```

```
      -o-transition: box-shadow 0.7s;
         transition: box-shadow 0.7s;
}

.painel li:hover {
  background-color: rgba(255,255,255,0.8);
  -webkit-transition: all 0.7s;
     -moz-transition: all 0.7s;
       -o-transition: all 0.7s;
          transition: all 0.7s;
}

.painel li:hover {
  -webkit-transform: scale(1.2);
     -moz-transform: scale(1.2);
      -ms-transform: scale(1.2);
       -o-transform: scale(1.2);
          transform: scale(1.2);
}

.painel li:hover {
  -webkit-transform: scale(1.2) rotate(-5deg);
     -moz-transform: scale(1.2) rotate(-5deg);
      -ms-transform: scale(1.2) rotate(-5deg);
       -o-transform: scale(1.2) rotate(-5deg);
          transform: scale(1.2) rotate(-5deg);
}

.painel li:nth-child(2n):hover {
  -webkit-transform: scale(1.2) rotate(5deg);
     -moz-transform: scale(1.2) rotate(5deg);
      -ms-transform: scale(1.2) rotate(5deg);
       -o-transform: scale(1.2) rotate(5deg);
          transform: scale(1.2) rotate(5deg);
}

.painel li {
  -webkit-transition: all 0.7s;
     -moz-transition: all 0.7s;
       -o-transition: all 0.7s;
          transition: all 0.7s;
}

.painel li:hover {
  -webkit-transition: all 0.7s ease-in;
     -moz-transition: all 0.7s ease-in;
       -o-transition: all 0.7s ease-in;
          transition: all 0.7s ease-in;
}
.painel li {
  -webkit-transition: all 0.7s ease-out;
     -moz-transition: all 0.7s ease-out;
       -o-transition: all 0.7s ease-out;
          transition: all 0.7s ease-out;
}
```

Quem vai decidir a quantidade de navegadores suportado normalmente são as estatísticas de acesso ao site. Google Analytics, por exemplo, consegue identificar quais navegadores acessam o site e essa informação pode nos ajudar a calibrar a configuração do `autoprefixer`. Podemos até querer suportar todos os navegadores, mas nosso arquivo CSS ficará maior e sabemos que tamanho de arquivos impactam na largura de banda.

Agora podemos voltar com o `cssmin`, que será processado após o `autoprefixer`.

```
gulp.task('usemin', function() {
  return gulp.src('dist/**/*.html')
    .pipe(usemin({
      js: [uglify],
      css: [autoprefixer, cssmin]
    }))
    .pipe(gulp.dest('dist'));
});
```

Vamos para os exercícios?