01

Imagens Vetoriais

Transcrição

Nesta aula continuaremos com o assunto de *imagens responsivas*. A grande questão na aula passada era a adaptação das imagens e seus pixeis flexibilizando-os para diversas plataformas.

Se os pixeis nos mostraram ser um grande problema, por que usá-los? Vamos trabalhar aqui com ***Desenhos Vetoriais***. Uma das vantagens é que um desenho vetorial é composto por coordenadas matemáticas e, independentemente da modificação que fizermos, ele nunca perde a qualidade. Sua única desvantagem é que esse conceito não serve para fotografias, apenas para logotipos, gráficos, ícones, etc.

O foco desta aula é trabalharmos com esses *desenhos vetoriais* na *Web*. Perceba que nos utilizando deles sempre teremos a melhor qualidade possível em imagens, tanto em telas comuns como de retina.

CSS

Vamos tentar utilizar CSS o máximo possível, pois ele é um mecanismo vetorial. Tudo o que ele desenha é feito dinamicamente no navegador, dependendo da resolução e do tamanho da tela. Veja alguns exemplos:

- Bordas arredondadas:

```
.panel {
  border-radius: 10px;
}
```

- Sombreamentos:

```
.container {
  box-shadow: 3px 3px 10px rgba(0,0,0,.3);
}
```

- Gradientes:

```
.main {
  background: linear-gradient(#F90, #FF0);
}
```

"Texto como Texto"

Antigamente, quando queríamos usar um texto um pouco diferente, como fazer um *banner*, tínhamos que montar a imagem através de um editor externo e inserí-la no HTML já com o texto. O problema de imagens com texto é que podemos perder a qualidade e ter que fazermos todos aqueles processos de *imagem responsiva* que vimos na aula passada.

Por sua vez, se o texto do *banner* estiver "como texto" no código, não teremos esse tipo de problema. Hoje em dia na *Web* podemos usar fontes customizadas que já criam esses elementos automaticamente, pois texto é sempre vetorial.

Icon Fonts

É tão bom trabalhar com texto que usamos fontes como ícones, por suas propriedades vetoriais. Muitas vezes é preferível utilizar uma fonte que possua desenhos (em vez de letras), a utilizar uma imagem. Veja uns exemplos de *icon fonts*:

Retirado de: https://icomoon.io/app/#/select (https://icomoon.io/app/#/select)

Cada desenho desses é um caractere de uma fonte especial.

Com as *icon fonts* podemos, por exemplo, com a propriedade "*color*" do CSS, modificar a cor do ícone:

Nesse caso o ícone da Caelum foi exportado como fonte.

Mesmo com todas essas possibilidades, existem algumas desvantagens ao trabalhar com fontes.

- Complexidade do ícone: a cor é sempre uma só para o ícone inteiro, não conseguimos criar desenhos coloridos. Um caractere não pode ter mais de uma cor.
- Renderização: a forma como cada navegador ou sistema operacional renderiza as fontes é diferente. O posicionamento e a grossura das linhas dos caracteres variam ligeiramente. Tais diferenças podem ou não ser graves, causando desalinhamentos.

Em contrapartida, as *icon fonts*:

- são muito simples
- são muito leves
- têm suporte de navegadores até muito antigos

SVG

O *SVG* é a melhor solução para gráficos vetoriais na *Web*, resolvendo quase 100% dos problemas com cenários de desenhos vetoriais. Ele permite fazermos desenhos muito mais complexos.

Ele é o formato vetorial padrão da *Web* e pode ser exportado de diversas ferramentas de desenho, como o *Illustrator*.

É um formato de imagem como qualquer outro, então podemos usá-lo com a *tag* "img":

```
<img src="logo.svg">
```

O SVG é um formato muito familiar para os programadores em HTML porque ele é baseado em *tags*, o que nos permite escrevê-las misturadas com as *tags* do HTML:

```
< doctype html>
<html>
<body>
```

```
    <h1> Meu teste de SVG</h1>

    <svg>
        <circle cx="50" cy="50" r="40" fill="#F90">
    </svg>
</body>
</html>
```

Neste exemplo estamos fazendo uma imagem simples desenhando um círculo laranja usando a *tag* "svg".

Além de baixarmos um recurso externo com "img", podemos escrever a propria imagem direto na página.

Uma outra possibilidade é usarmos o CSS:

```
.logo {
    background: url(logo.svg) no-repeat;
}
```

Todas essas são possibilidades em SVG na *Web* de hoje. O problema desse formato é o seu suporte para *Internet Explorer 8* e anteriores e *Android 2* e anteriores.

Suporte SVG

Se necessitamos de suporte para navegadores antigos, precisaremos para cada um dos ícones, além de sua versão SVG, da sua versão PNG. para exportar é bem simples: basta ir no editor de imagem e fazer para ambos os formatos.

Para os navegadores antigos, entregamos a versão PNG. Para os novos a SVG. Na prática temos duas maneiras:

1. O navegador tentará baixar a versão SVG, se houver erro ele troca para a versão PNG.

```
<img src="logo.svg"
 onerror="this.src='logo.png'; this.onerror=null">
```

   Navegadores sem suporte não muito antigos, como o IE8, baixarão de qualquer forma o imagem SVG e só a substituirão quando ela der erro. Ou seja, há download duplicado. Como o IE8 está saindo do mercado, esse problema é mínimo. O mais importante é o suporte. De qualquer jeito, ainda existe uma segunda possibilidade usando o CSS.

2. No CSS poderíamos acrescentar uma classe para verificar se o navegador tem suporte ao SVG ou não:

```
.svg .logo {
    background: url(logo.svg) no-repeat;
}

.no-svg .logo {
    background: url(logo.png) no-repeat;
}
```

Um *script* que detecta esse suporte é o *Modernizr*, uma biblioteca JavaScript que automaticamente insere tais *tags*.