Técnicas de detecção de funcionalidades do HTML 5 por JavaScript.

Pode ser que o usuário não utilize um browser que suporta específicas utilidades do HTML5 e CSS3. Neste caso, você pode redirecioná-lo para uma versão do site mais simples, ou talvez apenas mostrar uma mensagem alertando o usuário sobre a importância da atualização do browser. Para isso temos algumas técnicas de detecção para conferir se o browser suporta ou não HTML5 ou CSS3.

Quando um browser renderiza uma página, carrega um conjunto de objetos para o documento (Document Object Model – DOM) representando todos os objetos HTML presentes na página. Todos os elementos são representados nesse conjunto de objetos por um objeto diferente.

Todos os objetos DOM compartilham um conjunto de propriedades, mas alguns objetos possuem outras. Em browsers que suportam HTML5, há alguns tipos de objetos que possuem propriedades específicas, que não estão disponíveis nos browsers que não oferecem mesmo suporte. Um exame desses objetos permite determinar que *features* do HTML5 estão disponíveis e quais não estão.

Para detectarmos se o browser que estamos utilizando suporta Canvas, por exemplo, tentamos instanciar um elemento desse tipo e verificamos a disponibilidade do método “getContext”. Observe:

```
function supportsCanvas() {
    return !!document.createElement('canvas').getContext;
}
```

- Criamos um elemento Canvas;
- Checamos a presença do método getContext que só está disponível caso o Canvas esteja disponível;
- Usamos uma dupla negativa (!!) para forçar um resultado boolean.

Utilizando o Modernizr

O Modernizr (http://www.modernizr.com/) é uma biblioteca de detecção que lhe permite verificar o suporte da maioria das características do HTML5 e CSS3.

O Modernizr é um script javascript que roda automaticamente assim que você o adiciona no head do documento. Assim, se você quiser verificar se o browser suporta Geolocalização, por exemplo, basta inserir este script na página:

```
if(Modernizr.geolocation) {
  alert("Feature suportada pelo browser");
} else {
  alert("Feature não suportada pelo browser");
}
```

Canvas API

Uma das principais mudanças com a chegada do HTML5 foi a implementação de uma API gráfica interpretada sem plug-ins, manipulada pela tag canvas. A canvas API permite a você desenhar na tela do navegador via Javascript. Veja como inserir o elemento Canvas numa página:

```
<canvas id="x" width="300" height="300"></canvas>
```

Esse código vai exibir um retângulo vazio. Para desenhar nele, primeiro obtemos o contexto do desenho, com Javascript:

```
context=document.getElementById('x').getContext('2d');
```

Agora que temos um contexto, podemos desenhar nele. Vamos começar com um simples retângulo.

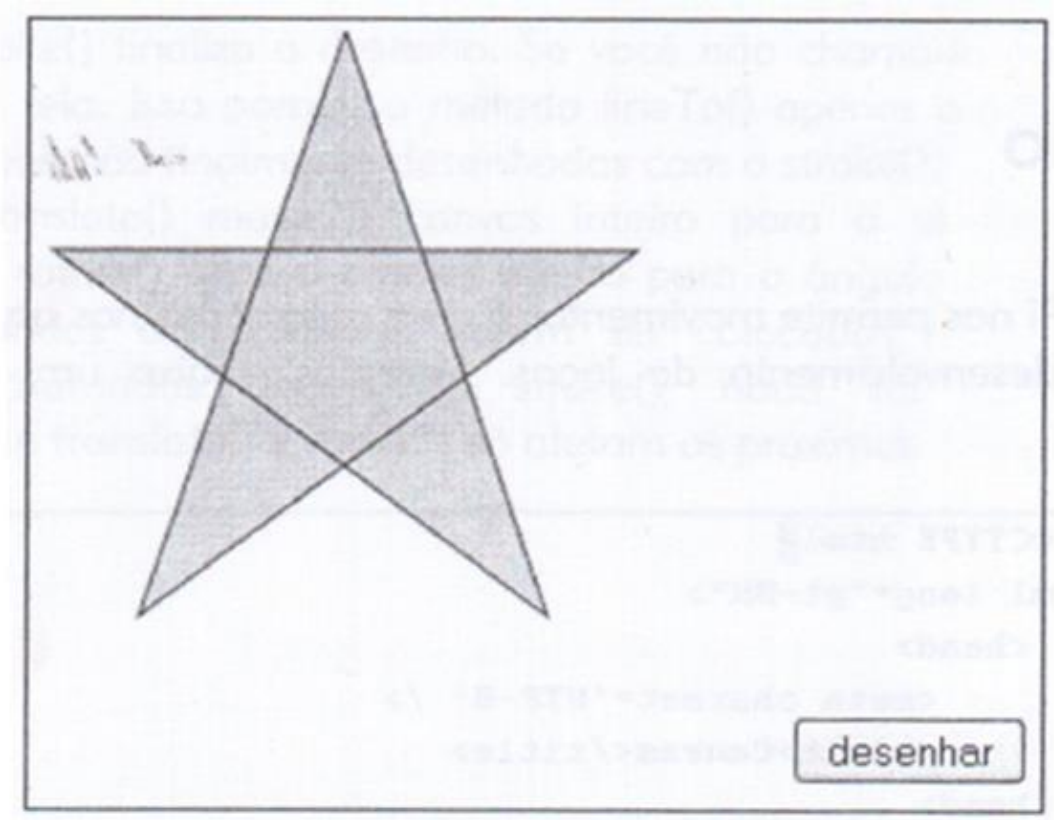
```
context.fillRect(10, 10, 50, 150);
```

Canvas permite desenhar na página e atualizar dinamicamente estes desenhos, por meio de scripts e atendendo às ações do usuário. Tudo isso dá possibilidades de uso tão grandes com as que dispomos com o plugin de Flash, no que se refere á renderização de conteúdos dinâmicos. As aplicações podem ser grandes como possamos imaginar, desde jogos, efeitos dinâmicos em interfaces de usuário, editores de código, editores gráficos, aplicações, etc.

Atualmente, alguma aplicações para a Web já utilizam Canvas para seu funcionamento, na qual se destacam Bespin, um editor de código do Mozilla, ou Google Wave.

Veremos agora um exemplo de Canvas atributo a um botão Javascript:

```
<canvas id="x" width="300" height="300"></canvas>
<button onclick="desenhar()">desenhar</button>
<script>
function desenhar(){
  // Obtemos o contexto
  context=document.getElementById('x').getContext('2d')

  //Iniciamos um novo desenho
  context.beginPath()

  //Movemos a caneta para o inicio do desenho
  context.moveTo(150,50)

  //Desenhamos as linhas
  context.lineTo(220,250)
  context.lineTo(50,125)
  context.lineTo(250,125)
  context.lineTo(80,250)
  context.lineTo(150,50)

  //O desenho não é de verdade enquanto você
  //não mandar o contexto pintá-lo.

  //Vamos pintar o interior de amarelo
  context.fillStyle='#ff0'
  context.fill()

  //Vamos pintar as linhas de vermelho.
  context.strokeStyle='#f00'
  context.stroke()

}
</script>
```



Animação

O Canvas API nos permite movimentar e criar animações nos objetos da tela, por isso é utilizado no desenvolvimento de jogos. Veremos abaixo um simples exemplo de animação.

```
<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <meta charset="UTF-8" />
        <title>Canvas</title>
    </head>
    <body>
        <canvas id="canvas" width="960" height="480">
        </canvas>
        <script type="text/javascript" charset="utf-8">
            var canvas = document.getElementById("canvas");
            var paper = canvas.getContext("2d");

            function draw() {
                paper.clearRect(0, 0, 960, 480);
                paper.translate(30, 30);
                paper.rotate(0.1);
                paper.beginPath();
                paper.moveTo(10, 0);
                paper.lineTo(20, 30);
                paper.lineTo(0, 30);
                paper.closePath();
                paper.stroke();
            }

            setInterval(draw, 200);
        </script>
    </body>
</html>
```

No exemplo acima, estamos criando um desenho mais customizado. O método beginPath() define ao Canvas que vamos começar um desenho usando as APIs do PATH, que podemos associar a um “lápis” na tela.

O moveTo() move nosso “lápis” para o ponto 10,0(x,y). A partir do ponto 10,0, desenhamos uma linha no método lineTo() para o ponto 20,30, Neste momento, é como se fosse desenhada uma linha no mesmo. O mesmo fazemos depois para o ponto 0,30. O método closePath() fecha automaticamente nosso desenho para o ponto inicial, no caso 10,0.

O método stroke() finaliza o desenho. Se você não chama-lo ao final, seu desenho não aparecerá na tela. Isso, porque o método lineTo() apenas adiciona as coordenadas numa fila, que depois são finalmente desenhadas com o stroke().

O método translate() move o canvas inteiro para a posição x,y passada nos argumentos. Já o rotate(), gira o canvas inteiro para o ângulo passado (em radianos).

Essas duas chamadas dos métodos devem ser colocadas antes de beginPath(). Se colocarmos as chamadas depois do stroke(), nada vai acontecer, porque as transformações com translate() e rotate() só afetam os próximos desenhos.

Localização do usuário com API Geolocation

Com a API Geolocation que está sendo implementada com o HTML5, saber a localização de quem visita seu site se tornou bem simples. A API Geolocation prevê a captura da posição geográfic, ou seja, latitude e longitude. E isso já resolve tudo. Com as coordenadas geográficas é possível integrar com outras ferramentas e obter o logradouro e até mesmo integrando com o Google Maps gerar um mapa com a localização do usuário.

Obtendo as coordenadas geográficas

Para entender o funcionamento da API vamos a um simples exemplo para obter as coordenadas geográficas.

```
<script>
navigator.geolocation.getCurrentPosition(showpos);

function showpos(position){
  lat=position.coords.latitude
  lon=position.coords.longitude
  alert('Sua posição: '+lat+','+lon)
}
</script>
```