VETORES E ANIMAÇÕES COM SVG

SVG são basicamente desenhos feitos com base em cálculos que você irá definir no código. Por se tratar de contas e não de pixels, acaba que a imagem, o desenho não perderá sua qualidade independente de um zoom que estiver sendo direcionado ao mesmo e etc.

No HTML, o SVG segue como um texto de marcação, ou seja, segue a maneira como se é estruturado os elementos como é feito no HTML, usando TAGs.

Para definirmos os cálculos para algum desenho devemos iniciar com a TAG <svg></svg>, e dentro dela inserir outros elementos, como por exemplo, se você quiser desenhar um círculo. Para isso você precisará das seguintes TAGs:

<circle> do qual dentro dos “<” “>”, você irá especificar a coordenada X (cx), a coordenada Y (cy), o raio (r), o stroke – a borda -, stroke-width – largura da borda -, a cor de preenchimento do círculo (fill)

Além do círculo, podemos fazer quadrados, retângulos, triângulos, polígonos, linhas de diversas formas e etc. Cada uma com suas especificações de tags.

Path é outra TAG do SVG bem importante, pois nele podemos fazer desenhos mais complexos e diferentes, misturando linhas, curvas e etc. Ele é mais chatinho de fazer, principalmente por conta de possuir algumas especificações que ditam o que fazemos em código, e são essas:

M = move até L = linha até H = linha horizontal até V = linha vertical até C = curva até S = curva suave até Q = curva de Bézier T = curva de Bézier suave A = arco elíptico Z = fecha path

O XML é uma "metalinguagem" que permite ao usuário a criação de suas próprias linguagens de marcação, baseadas nesse modelo. Ou seja, é permitida a criação de outros elementos (ou tags) além das já definidas, por exemplo, no HTML.

Para importamos o SVG para dentro do nosso HTML podemos usar dois jeitos:

1º Através da TAG <img></img>, colocando a src da imagem e etc.

2º Trazendo todo o conteúdo da imagem, os códigos para dentro do HTML com a TAG <SVG>

Seguindo o 2º método, se torna possível modificar os SVGs no CSS, expandindo o leque de possibilidades para nossa página.

Animações

Para fazermos a animação de algum objeto, podemos usar do CSS. Para isso, iremos precisar de um seletor (id, classe, etc) do obj desejado e conhecer algumas propriedades que geraram o movimento.

Para começo, precisamos o usar o @keyframes que falará que trabalharemos com uma sequência de frames, ou seja, o passo-a-passo do movimento.

Feito isso atribuíremos um nome a esse keyframe e abriremos chaves ({ })

Dentro das chaves colocaremos a animação desejada. Iniciamos com o to { } simbolizando que o objeto saíra de seu ponto inicial PARA um novo ponto. Se quisermos o contrário, ou seja, dizendo onde o elemento começa, usamos o from { }.

E novamente dentro das chaves colocaremos a animação, que pode ser por exemplo de um simples movimento com a propriedades transform: translate(nova coordenada, nova coordenadaY). Assim estabelecemos o novo ponto que nosso objeto irá.

Claro que fazendo só isso, não adiantará muito, então precisamos linkar esse movimento com o obj. fazemos isso chamando o id ou classe do elemento e abrindo chaves, aí então usamos a propriedade animation que poderá receber diversos parâmetros, eles são:

-Nome do keyframes

-tempo da animação

-tipo do movimento

-Quantas vezes será repetida

-Maneira como irá transitar o fim com o começo da animação

Usando apenas o “shorthand” animation podemos colocar todas as suas propriedades de maneira mais rápida, porém se quisermos ser mais específicos podemos usar as seguintes propriedades:

* Animation-name (nome para "chamarmos" a animação)
* Animation-duration (duração em segundos ou milissegundos)
* Animation-timing-function (controla a velocidade da animação: a mesma durante a animação toda, mais lenta no início, mais lenta no final, etc)
* Animation-delay (o tempo de pausa antes de iniciar a animação, em segundos ou milissegundos)
* Animation-iteration-count (quantas vezes a animação será reproduzida em sequência)
* Animation-direction (a direção da animação: direção normal, de trás para frente, alternando normal e trás para frente, etc)
* Animation-fill-mode (podemos aplicar estilos diferentes na animação enquanto ela executa)
* Animation-play-state (especifica se a animação está sendo executada ou pausada)

Na “shorthand” fica mais ou menos assim:

animation: [name] [duration] [timing-function] [delay] [iteration-count] [direction];

Viewport: É toda a área visível do site, tudo o que podemos ver nos sites, são as dimensões

Viewbox: É uma parte do viewport, é tipo quando damos um zoom em um site, a parte que aparece nesse zoom é tipo um viewbox da página.

Veremos alguns elementos do JavaScript que nos ajudam no que se trata estruturar o código e realizar animações:

Exportarmos ou importamos algo de um arquivo js para outro, usamos:

Import { o que queremos importar } from ‘onde está o arquivo’

E para exportarmos usamos:  
export { o que queremos exportar }

No JS podemos criar uma classe do HTML/CSS através do comando class nome da classe, abrir chaves e escrever tudo o que essa classe representa, seja funções, ações, animações, que posteriormente podemos usar para transitar esse conjunto de código de um arquivo ao outro.

Como falamos de estruturação, usamos uma função chamada constructor () { } para deixarmos dentro dela tudo que precisamos pegar do HTML e trazer para o JS, através de variáveis, deixando nossos códigos mais organizados e limpos.

Para criarmos uma função, não necessariamente precisamos do function, podemos apenas colocar o nome da mesma, seguida por () e uma abertura de chaves.

This e Bind

This: refere a um objeto global, porém pode ter diferentes valores, dependendo de onde está sendo utilizada, como os casos:

* Em um método, refere-se ao objeto ao qual o método está ligado (por exemplo, obj.metodo()).
* Sozinha ou em uma função, refere-se ao objeto global (no caso do navegador, o objeto window).
* Em uma função em strict mode, é undefined.
* Em um evento, refere-se ao elemento que recebeu o evento.
* Quando falamos de objetos (criados com class ou não), this refere-se ao objeto

Exemplo:

function Pessoa(nome, sobrenome) {

this.nome = nome;

this.sobrenome = sobrenome;

this.nomeCompleto = function() {

console.log(`Meu nome é ${this.nome} ${this.sobrenome}`);

}

}

const pessoa = new Pessoa("Leia", "Organa");

pessoa.nomeCompleto() // 'Meu nome é Leia Organa'

const outraPessoa = new Pessoa("Luke", "Skywalker");

outraPessoa.nomeCompleto() // 'Meu nome é Luke Skywalker'

const nomeOutraPessoa = pessoa.nomeCompleto.bind(outraPessoa);

// Cria uma nova função onde o valor de `this` se liga ("bind") ao objeto "outraPessoa"

nomeOutraPessoa(); // 'Meu nome é Luke Skywalker'

BIND: usamos para "ligar" this a um contexto específico para que o this não fique "perdido" ao ser invocado fora de seu objeto.

Alguns atributos do JS:

-getBoundingClientRect()[]: serve para localizar em que ponto, ou no eixo x ou y (isso se especifica entre os colchetes), a janela está mostrando. É como se fosse um ponto cartesiano em que se mostrará a coordenada exata de acordo com o que está sendo mostrado

-const: usamos para criar uma instância, juntamente com o new seguido com a function que queremos criar a instância.

FILTROS

Para usar os filtros é simples, basta usar a tag filter e dentro dela se jogar no imenso mar de possibilidades proporcionado pelos filtros.

Nos links abaixo você verá tudo sobre isso:  
<https://developer.mozilla.org/pt-BR/docs/Web/SVG/Element/filter>

<https://tympanus.net/codrops/2019/01/15/svg-filters-101/>

<https://www.w3schools.com/graphics/svg_filters_intro.asp>

Tags DEFS

Essas servem quando queremos definir algo do HTML sem que isso seja mostrado automaticamente no site, sendo só ativado quando for chamado.

Também é possível aplicar gradientes em nossos elementos SVG. Vemos isso no artigo abaixo:

<https://www.w3schools.com/graphics/svg_grad_linear.asp>