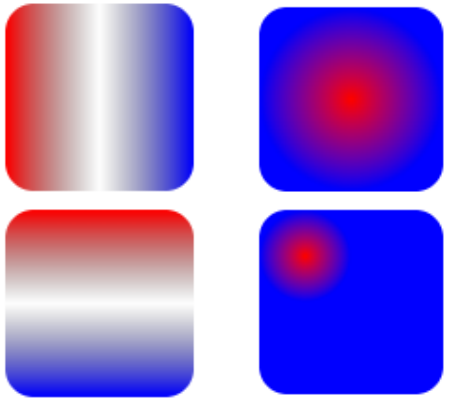
数据可视化 SVG

刘军　liujun

目录
content

线性渐变

- **SVG除了可以简单的填充和描边，还支持在填充和描边上应用渐变色。渐变有两种类型：线性渐变 和 径向渐变。**
  - 编写渐变时，必须给渐变内容指定一个 id 属性，use引用需用到。
  - 建议渐变内容定义在<defs>标签内部，渐变通常是可复用的。



- **线性渐变，是沿着直线改变颜色。下面看一下线性渐变的使用步骤：**
  - 第1步：在 SVG 文件的 defs 元素内部，创建一个<linearGradient>节点，并添加 id 属性。
  - 第2步：在<linearGradient>内编写几个<stop>结点。
    - ✓ 给<stop> 结点指定位置 offset属性和 颜色stop-color属性，用来指定渐变在特定的位置上应用什么颜色
      - ➢ offset 和 stop-color 这两个属性值，也可以通过 CSS 来指定。
    - ✓ 也可通过 stop-opacity 来设置某个位置的半透明度。
  - 第3步：在一个元素的 fill 属性或 stroke 属性中通过ID来引用 <linearGradient> 节点。
    - ✓ 比如：属性fill属性设置为url(#Gradient2)即可。

```
<linearGradient id="Gradient2" x1="0" x2="0" y1="0" y2="1">
```

  - 第4步（可选）：控制渐变方向，通过 (x1，y1) 和 （ x2，y2) 两个点控制。
    - ✓ （0, 0） （0, 1）从上到下； （0, 0） （1, 0）从左到右。
    - ✓ 当然也可以通过 gradientTransform 属性 设置渐变形变。比如： gradientTransform= “rotate(90)” 从上到下。

SVG 毛玻璃效果

- 在前端开发中，毛玻璃效果有几种方案来实现：
- 方案一：使用CSS的 backdrop-filter 或 filter 属性
  - **backdrop-filter**：可以给一个元素后面区域添加模糊效果。

    适用于元素背后的所有元素。为了看到效果，必须使元素或其背景至少部分透明。
  - **filter**：直接将模糊或颜色偏移等模糊效果应用于指定的元素。

```
.bg-cover {
    ....
    background-color: rgba(0, 0, 0, 0.1);
    backdrop-filter: blur(8px);
}
</style>
head>
ody>
<div class="box">
  <img src="../images/avatar.jpeg" alt="" />
  <div class="bg-cover"></div>
</div>
```

- 方案二：使用SVG的 filter 和 feGaussianBlur 元素（建议少用）
  - < filter>：元素作为滤镜操作的容器，该元素定义的滤镜效果需要在SVG元素上的 filter 属性引用。
    - ✓ x， y, width, height 定义了在画布上应用此过滤器的矩形区域。x， y 默认值为 **-10%**（相对自身）；width， height 默认值为 **120%** （相对自身）。
  - < feGaussianBlur >：该滤镜专门对输入图像进行高斯模糊
    - ✓ stdDeviation 熟悉指定模糊的程度
  - <feOffset>：该滤镜可以对输入图像指定它的偏移量。

```
<filter id="blueFilter">
  <feOffset dx="0" dy="0" />
  <feGaussianBlur stdDeviation="8"></feGaussianBlur>
</filter>
```

形变- transform

- **transform 属性用来定义元素及其子元素的形变的列表。**
  - 此属性可以与任何一个 SVG 中的元素一起使用。如果使用了变形，会在该元素内部建立了一个新的坐标系统。
  - 从 SVG2 开始，transform它是一个 Presentation Attribute，意味着它可以用作 CSS 属性。
  - 但是transform作为CSS 属性和元素属性之间的语法会存在一些差异。
    - ✓ 比如作为元素属性时：支持2D变换，不需单位，rotate可指定旋转原点。
- **transform属性支持的函数：**
  - translate(x， y) 平移。
  - rotate(z) / rotate(z， cx，cy)： 旋转。
  - scale（x, y） ：缩放
  - skew(x, y)： 倾斜。
  - matrix(a, b, c, d, e)： 2*3的形变矩阵
- 注意：形变会不会修改坐标系？ 会，形变元素内部会建立一个新的坐标系，后续的绘图或形变都会参照新的坐标系。

形变-平移

- **平移：把元素移动一段距离， 使用transform属性的 translate()函数来平移元素。**
  - 与CSS的translate相似但有区别，这里只支持2D变换，不需单位。
- **translate(x, y)函数**
  - 一个值时，设置x轴上的平移，而第二个值默认赋值为0
  - 二个值时，设置x轴和y轴上的平移
- 平移案例：将一个矩形由默认的（0,0）点，移到点 (30,40)。

```
<rect x="0" y="0" width="10" height="10" transform="translate(30,40)" />
```

- 注意：平移会不会修改坐标系？ 会，元素内部会建立一个新的坐标系。

形变-旋转

- **旋转：把元素旋转指定的角度， 使用transform属性的 rotate(deg，cx, cy) 函数来旋转元素。**
  - 与CSS的rotate相似但有区别。区别是：支持2D变换，不需单位，可指定旋转原点。
- **rotate(deg, cx, cy) 函数**
  - 一个值时，设置z轴上的旋转的角度。

```
<svg width="40" height="50" style="background-color:#bff;">
    <rect x="0" y="0" width="10" height="10" transform="translate(30,40) rotate(45)" />
</svg>
```

- 注意：
  - 旋转会不会修改坐标系？ 会，坐标轴也会跟着旋转了
  - 如何指定旋转原点？ 直接在rotate中指定 cx ,cy（相对于自身）； 或者使用CSS样式写动画。

形变-缩放

- **缩放：改变元素尺寸，使用transform属性的 scale() 函数来缩放元素。**
  - 与CSS的scale相似但有区别，这只支持2D变换，不需单位。

- **scale(x, y)函数**
  - 二个值时：它需要两个数字，作为比率计算如何缩放。0.5 表示收缩到 50%。
  - 一个值时：第二个数字被忽略了，它默认等于第一个值。

```
<g transform="scale(2)">
  <rect width="50" height="50" />
</g>
```

- 注意：
  - 缩放会不会修改坐标系？会，坐标轴被缩放了。
  - 如何指定缩放的原点？ SVG属性实现需要 平移坐标 和 移动图形了；或者 直接使用CSS来写动画

stroke描边动画

- **stroke 是描边属性，专门给图形描边。如果想给各种描边添加动画效果，需用到下面两个属性：**
  - stroke-dasharray = “number [, number , ...]”：将虚线类型应用在描边上。
    - ✓ 该值必须是用逗号分割的数字组成的数列，空格会被忽略。比如 3，5 ：
      - ➢ 第一个表示填色区域的长度为 3
      - ➢ 第二个表示非填色区域的长度为 5
  - stroke-dashoffset：指定在dasharray模式下路径的偏移量。
    - ✓ 值为number类型，除了可以正值，也可以取负值。

```
.hy-line2 {
  stroke: red;
  stroke-width: 5;

  stroke-dasharray: 200;
  stroke-dashoffset: 200px;
  animation: dash 1s linear infinite;
}
@keyframes dash {
  100% {
    stroke-dashoffset: 0px;
  }
}
```

- **描边动画实现步骤：**
  - 1.先将描边设置为虚线
  - 2.接着将描边偏移到不可见处
  - 3.通过动画让描边慢慢变为可见，这样就产生了动画效果了。

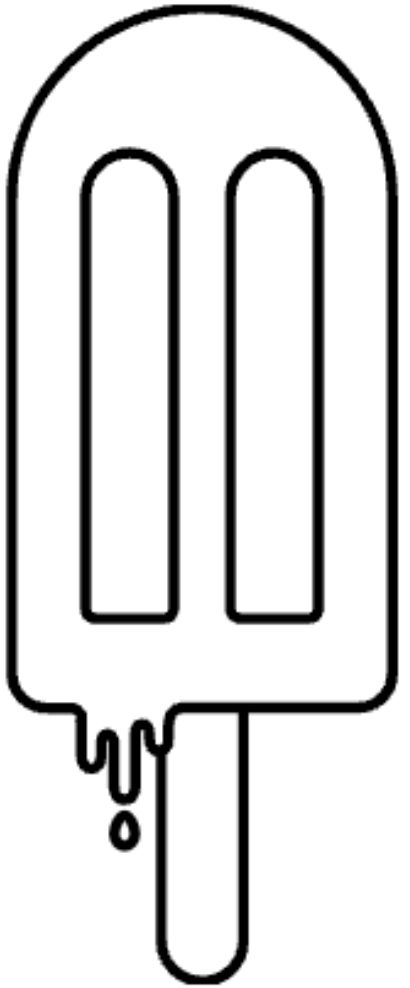
路径-描边案例

- **雪糕路径描边动画案例实现步骤：**
  - 1.找到一个雪糕的SVG图片（设计师提供 | 网站下载）
  - 2.将雪糕的每一个路径都改成虚线
  - 3.将每个路径的描边都移动到虚线的空白处（不可见）
  - 4.给每个路径添加动画，将路径描边慢慢移动到虚线填充处，即可。

```css
@keyframes offset {
  to {
    stroke-dashoffset: 0px;
  }
}
.drop,
.stick,
.inside-r,
.inside-l,
.outline {
  animation: offset 2s linear forwards;
}
```

```css
.stick {
  /* 需要大于256px */
  stroke-dasharray: 800px;
  stroke-dashoffset: 800px;

  animation-delay: 1.75s;
}
```



什么是SMIL?

- **SMIL（Synchronized Multimedia Integration Language 同步多媒体集成语言）是W3C推荐的可扩展标记语言，用于描述多媒体演示。**
  - SMIL 标记是用 XML 编写的，与HTML有相似之处。
  - SMIL 允许开发多媒体项目，例如：文本、图像、视频、音频等。
  - SMIL 定义了时间、布局、动画、视觉转换和媒体嵌入等标记，比如：<head> <body> <seq> <par> <excl> 等元素
- **SMIL的应用**
  - 目前最常用的Web浏览器基本都支持 SMIL 语言。
  - SVG 动画元素是基于SMIL实现（SVG中使用SMIL实现元素有：<set>、< animate >、< animateMotion >...）。
  - Adobe Media Player implement SMIL playback。
  - QuickTime Player implement SMIL playback。

SVG动画实现方式

- **SVG是一种基于XML的开放标准矢量图形格式，动画可以通过多种方式实现：**
  - **1.用JS脚本实现：**可以直接通过 JavaScript 在来给 SVG 创建动画和开发交互式的用户界面。
  - **2.用CSS样式实现：**自 2008 年以来，CSS动画已成为WebKit中的一项功能，使得我们可以通过CSS动画的方式来给文档对象模型(DOM) 中的 SVG 文件编写动态效果。
  - 3.用SMIL实现：一种基于SMIL语言实现的SVG动画。

SVG animation using ECMAScript [edit]

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graph
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http
width="100%" height="100%" viewBox="-4 -4 8 8" onload="rotate(evt)">
 <title>SVG animation using ECMAScript</title>
 <script type="text/ecmascript">
  function rotate(evt) {
   var object = evt.target.ownerDocument.getElementById('rot');
   setInterval(function () {
     var now          = new Date();
     var milliseconds = now.getTime() % 1000;
     var degrees      = milliseconds * 0.36; // 360 degrees in 1000 ms
     object.setAttribute('transform', 'rotate(' + degrees + ')');
    }, 20);
  }
 </script>
 <circle id="rot"
  cx="0" cy="1" r="2" stroke="green" fill="none"/>
</svg>
```

SVG animation using CSS [edit]

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
 width="100%" height="100%" viewBox="-4 -4 8 8">
 <title>SVG animation using CSS</title>
 <style type="text/css">
  @keyframes rot_kf {
    from { transform: rotate(0deg); }
    to   { transform: rotate(360deg); }
  }
  .rot { animation: rot_kf 1s linear infinite; }
 </style>
 <circle class="rot"
  cx="0" cy="1" r="2" stroke="blue" fill="none"/>
</svg>
```

SVG animation using SMIL [edit]

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
 width="100%" height="100%" viewBox="-4 -4 8 8">
 <title>SVG animation using SMIL</title>
 <circle cx="0" cy="1" r="2" stroke="red" fill="none">
  <animateTransform
   attributeName="transform"
   attributeType="XML"
   type="rotate"
   from="0"
   to="360"
   begin="0s"
   dur="1s"
   repeatCount="indefinite"/>
 </circle>
</svg>
```

SMIL动画的优势

- **SVG用SMIL方式实现动画，SMIL允许你做下面这些事情：**
  - 变动一个元素的数字属性（x、y……）
  - 变动变形属性（translation 或 rotation）
  - 变动颜色属性
  - 物件方向与运动路径方向同步等等
- **SMIL方式实现动画的优势：**
  - 只需在页面放几个animate元素就可以实现强大的动画效果，无需任何CSS和JS代码。
  - SMIL支持声明式动画。声明式动画不需指定如何做某事的细节，而是指定最终结果应该是什么，将实现细节留给客户端软件
  - 在 JavaScript 中，动画通常使用 setTimeout() 或 setInterval() 等方法创建，这些方法需要手动管理动画的时间。而SMIL声明式动画可以让浏览器自动处理，比如：动画轨迹直接与动画对象相关联、物体和运动路径方向、管理动画时间等等。
  - SMIL 动画还有一个令人愉快的特点是，动画与对象本身是紧密集成的，对于代码的编写和阅读性都非常好。

SVG animation using SMIL [edit]

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
 width="100%" height="100%" viewBox="-4 -4 8 8">
 <title>SVG animation using SMIL</title>
 <circle cx="0" cy="1" r="2" stroke="red" fill="none">
  <animateTransform
   attributeName="transform"
   attributeType="XML"
   type="rotate"
   from="0"
   to="360"
   begin="0s"
   dur="1s"
   repeatCount="indefinite"/>
 </circle>
</svg>
```

SMIL动画的元素

- SVG 中支持SMIL动画的元素：
  - <set> <animate> <animateColor> <animateMotion>
  - 更多：https://www.w3.org/TR/SVG11/animate.html#AnimationElements

SVG supports the following four animation elements which are defined in the SMIL Animation specification:

'animate'	allows scalar attributes and properties to be assigned different values over time
'set'	a convenient shorthand for 'animate', which is useful for assigning animation values to non–numeric attributes and properties, such as the 'visibility' property
'animateMotion'	moves an element along a motion path
'animateColor'	modifies the color value of particular attributes or properties over time

Although SVG defines 'animateColor', its use is deprecated in favor of simply using the 'animate' element to target properties that can take color values.

Additionally, SVG includes the following compatible extensions to SMIL Animation:

'animateTransform'	modifies one of SVG's transformation attributes over time, such as the 'transform' attribute
'path' attribute	SVG allows any feature from SVG's path data syntax to be specified in a 'path' attribute to the 'animateMotion' element (SMIL Animation only allows a subset of SVG's path data syntax within a 'path' attribute)
'mpath' element	SVG allows an 'animateMotion' element to contain a child 'mpath' element which references an SVG 'path' element as the definition of the motion path
'keyPoints' attribute	SVG adds a 'keyPoints' attribute to the 'animateMotion' to provide precise control of the velocity of motion path animations
'rotate' attribute	SVG adds a 'rotate' attribute to the 'animateMotion' to control whether an object is automatically rotated so that its x–axis points in the same direction (or opposite direction) as the directional tangent vector of the motion path

Set元素

- **<set>元素提供了一种简单的方法，可以在指定的时间内设置属性的值。**
  - set元素是最简单的 SVG 动画元素。它是在经过特定时间间隔后，将属性设置为某个值（不是过度动画效果）。因此，图像不是连续动画，而是改变一次属性值。
  - 它支持所有属性类型，包括那些无法合理插值的属性类型，例如：字符串 和 布尔值。而对于可以合理插值的属性通常首选<animate>元素。
- **<set>元素常用属性：**
  - attributeName：指示将在动画期间更改的目标元素的 CSS 属性（ property ）或属性（ attribute ）的名称。
  - ~~attributeType:~~ (已过期，不推荐)指定定义目标属性的类型（值为：CSS | XML | auto）。
  - to : 定义在特定时间设置目标属性的值。该值必须与目标属性的要求相匹配。 值类型：<anything>；默认值：无
  - begin：定义何时开始动画或何时丢弃元素，默认是 0s（begin支持多种类型的值）。
- **<set>案例：**
  - **1）在3秒后自动将长方形瞬间移到右边**
  - **2）点击长方形后，长方形瞬间移到右边**

```
<rect id="hyRect" width="100" height="100">
    <set attributeName="x" to="100" begin="hyRect.click" />
    <set attributeName="x" to="100" begin="3s" />
</rect>
```

Animate元素

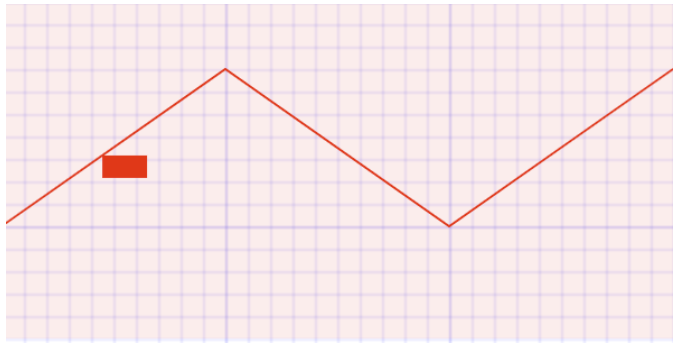
- **<animate>元素给某个属性创建过度动画效果。需将animate元素嵌套在要应用动画的元素内。**
- **<animate>元素常用属性：**
  - attributeName：指将在动画期间更改目标元素的 property （CSS 属）或 attribute的名称。
  - 动画值属性：
    - ✓ from：在动画期间将被修改的属性的初始值。没有默认值。
    - ✓ to :在动画期间将被修改的属性的最终值。没有默认值。
    - ✓ values：该属性具有不同的含义，具体取决于使用它的上下文（没有默认值） 。
      - ➢ 它定义了在动画过度中使用的一系列值，值需要用分号隔开，比如：values= “2 ; 3; 4; 5” 。
      - ➢ 当values属性定义时，from、to会被忽略。
  - 动画时间属性：
    - ✓ begin：定义何时开始动画或何时丢弃元素。默认是 0s 。
    - ✓ dur：动画的持续时间，该值必须，并要求大于 0。单位可以用小时 (h)、分钟 (m)、秒 (s) 或毫秒 (ms) 表示。
    - ✓ fill：定义动画的最终状态。 freeze（保持最后一个动画帧的状态） | remove（保持第一个动画帧的状态）
    - ✓ repeatCount：指示动画将发生的次数：<number> | indefinite。没有默认值。

animateTransform元素

- **< animateTransform >元素**
  - 指定目标元素的形变（transform）属性，从而允许控制元素的平移、旋转、缩放或倾斜动画（类似于 CSS3 的形变）。
  - 在一个动画元素中，只能用一个< animateTransform >元素创建动画；存在多个时，后面会覆盖前面的动画。
- **< animateTransform >元素常用属性：**
  - attributeName：指示将在动画期间更改的目标元素的 CSS 属性（ property ）或属性（ attribute ）的名称。
  - type：一个指定类型的属性，在不同的使用场景下，有不同的意思：
    - ✓ 在<animateTransform>元素，只支持 translate(x, y) | rotate(deg, cx, cy) | scale(x, y) | skewX(x) | skewY(y) 。
      - For a type="translate", each individual value is expressed as <tx> [,<ty>].
      - For a type="scale", each individual value is expressed as <sx> [,<sy>].
      - For a type="rotate", each individual value is expressed as <rotate-angle> [<cx> <cy>].
      - For a type="skewX" and type="skewY", each individual value is expressed as <skew-angle>.
    - ✓ 在 HTML 中的 <style > 和 <script > 元素，它定义了元素内容的类型。
  - 动画值属性：from、to 、values
  - 动画时间属性：begin、dur、fill、repeatCount

animateMotion元素

- **< animateMotion > 定义了一个元素如何沿着运动路径进行移动。**
  - 动画元素的坐标原点，会影响元素运动路径，建议从（0, 0）开始。
  - 要复用现有路径，可在<animateMotion>元素中使用<mpath>元素。



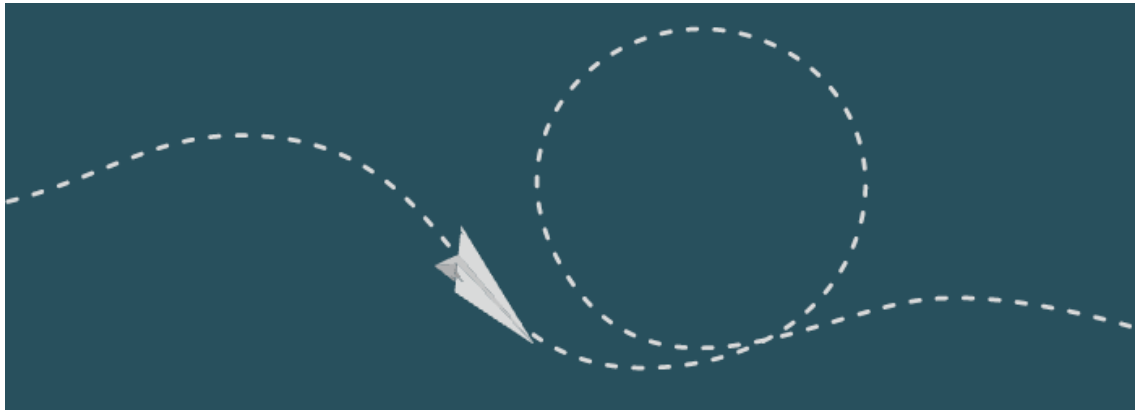
- **< aniamteMotion >元素常用属性：**
  - path：定义运动的路径，值和< path >元素的 d 属性一样，也可用 href 引用 一个 <path>。
  - rotate ： 动画元素自动跟随路径旋转，使元素动画方向和路径方向相同，值类型： <数字> | auto | auto-reverse; 默认值： 0
  - 动画值属性： from、to 、values
  - 动画时间属性： begin、dur、fill、repeatCount

```
<path d="M 0 100, 100, 30, 200 100, 300 30"
    fill="transparent"
    stroke="red"
></path>

<rect x="0" y="0" width="20" height="10" fill="red">
    <animateMotion
        path="M 0 100, 100, 30, 200 100, 300 30"
        dur="10s"
    ></animateMotion>
</rect>
```

SVG + SMIL动画

- 案例1：飞机沿轨迹飞行动画



- 案例2：加载进度动画

SVG + CSS3动画

- SVG + CSS3动画

Snap.svg

- **什么是Snap.svg?**
  - Snap.svg 是一个专门用于处理SVG的 JavaScript 库（ 类似jQuery ）。
  - Snap 为 Web 开发人员提供了干净、直观、功能强大的API，这些API专门用来操作SVG。
  - Snap 可用于创建动画，操作现有的 SVG 内容，以及生成 SVG 内容。
- **为什么选择Snap.svg?**
  - Snap 是由 Dmitry Baranovskiy 从零开始编写，专为现代浏览器（IE9 及更高版本、Safari、Chrome、Firefox 和 Opera）而设计。并且Snap可以支持遮罩、剪辑、图案、全渐变、组等功能。
  - Snap 还有一个独特功能是能够与现有的 SVG一起工作。意味着 SVG 内容不必使用 Snap 生成，就可使用 Snap 来处理它。
    - ✓ 比如可以在 Illustrator 或 Sketch 等工具中创建 SVG 内容，然后使用 Snap 对其进行动画处理和操作。
  - Snap 还支持动画。提供了简单直观的与动画相关的JavaScript API，Snap 可以帮助你的 SVG 内容更具交互性和吸引力
  - Snap.svg 库处理 SVG 就像 jQuery 处理 DOM 一样简单，并且 Snap 是 100% 免费和 100% 开源的。

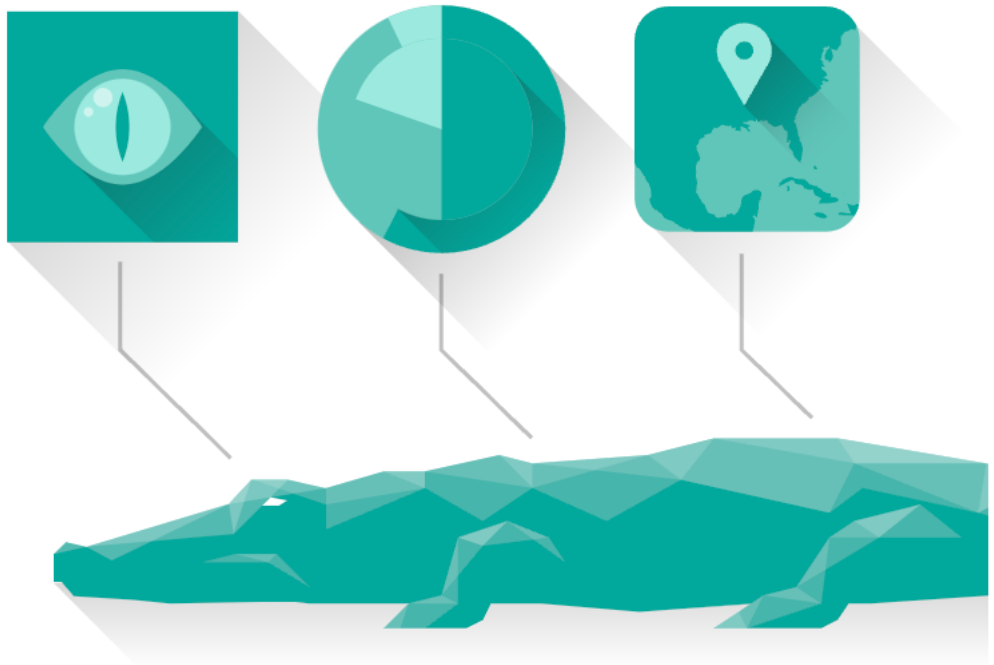
Snap.svg初体验

- **Snap.svg绘制一个圆， 如右图所示：**
- **Snap.svg常用的API：**
  - Snap： 工厂函数，创建或获取SVG
    - ✓ Snap(w, h) 、Snap(selector)....
  - Paper: 纸张 | SVG画布
    - ✓ circle、rect、line、path、text....
  - Element：元素
    - ✓ animate、attr、select、before、after...
  - mina：通常用到的一些动画时间函数。
    - ✓ mina.linear、mina.easeIn、mina.easeOut....
- Snap更多的API文档： http://snapsvg.io/docs/

```
<div id="app"></div>
<script src="./libs/snap.svg-min.js"></script>
<script>
  // 1.创建一个svg画布
  var svg = Snap(200, 200);
  // 2.拿到一张白纸
  var paper = svg.paper;
  // 3.在白纸上绘制一个圆
  paper.circle(50, 50, 50);
  // 4.将圆添加到页面上
  document.getElementById("app").appendChild(svg.node);
</script>
```

SVG + Snap动画

- SVG + Snap动画



GSAP动画库

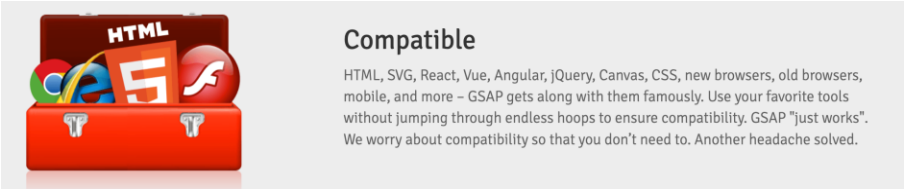
什么是GSAP

- GSAP全称是（ GreenSock Animation Platform）GreenSock 动画平台。
- GSAP 是一个强大的 JavaScript 动画库，可让开发人员轻松的制作各种复杂的动画。

GSAP动画库的特点

- 与Snap.svg不一样，GSAP无论是HTML 元素、还是SVG、或是Vue、React组件的动画，都可以满足你的需求。
- GSAP的还提供了一些插件，可以用最少的代码创建令人震惊的动画，比如：ScrollTrigger插件和MorphSVG插件。
  - ✓ https://greensock.com/scrolltrigger
- GSAP 的核心是一个高速的属性操纵器，随着时间的推移，它可以极高的准确性更新值。它比 jQuery 快 20 倍!
- GSAP 使用起来非常灵活，在你想要动画的地方基本都可以使用，并且是零依赖。

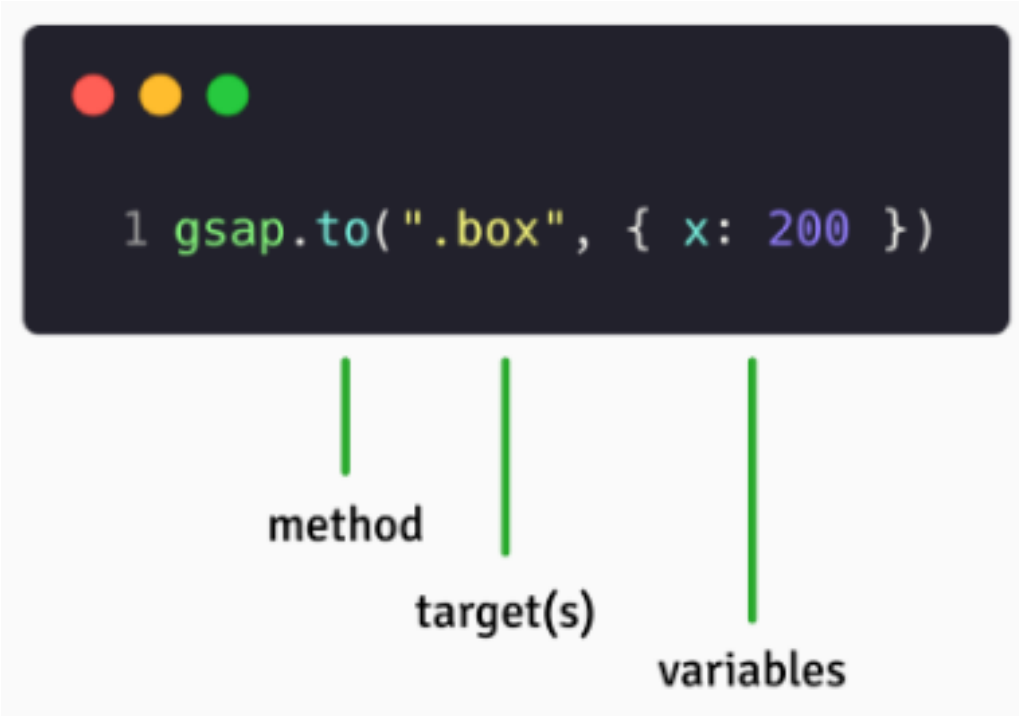
GSAP官网：https://greensock.com/



GSAP初体验

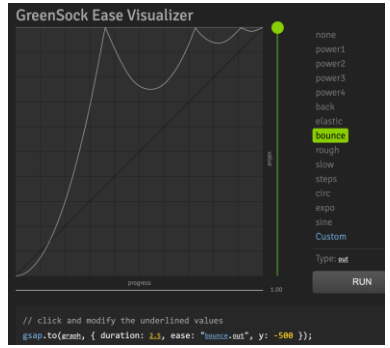
- **GSAP初体验：移动SVG中的一个矩形**
  - 引入 gsap.js 动画库（CDN，本地，npm）。
  - 调用 gsap.to 方法来执行 tween（补间/过度）动画。

```
<script src="./libs/gsap.min.js"></script>
<script>
    function moveRect() {
        // 1.一个单个tween（补间）动画称为
        gsap.to(".hy-rect", { x: 200 });
    }
</script>
```



GSAP 补间动画（Tween）

- **GSAP的Tween动画有4中类型：**
  - gsap.from(targets | selector, vars) - 元素从from定义的状态过度到元素当前的状态。
    - ✓ targets | selector： 需动画的元素对象，支持字符串的选择器
    - ✓ vars: 需过度动画的属性和GSAP扩展的duration、ease、transformOrigin、repeat、delay、yoyo、stagger、onComplete 等
    - ✓ 官网gsap.form文档：https://greensock.com/docs/v3/GSAP/gsap.from()
  - gsap.to(targets | selector, vars) - 元素从当前的状态过度到to状态。
  - gsap.fromTo(targets | selector, fromVars， toVars) -元素从from定义状态过度到to定义的状态
  - gsap.set(targets | selector, vars) - 立即设置属性（无过度效果）。
    - ✓ 本质上是一个 duration = 0 的 to 补间动画。

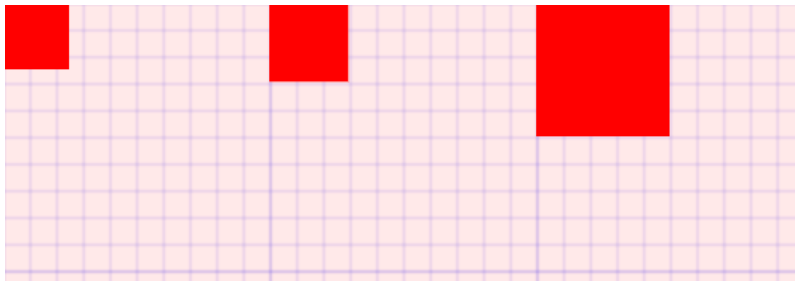


- **哪些属性可以设置动画?**
  - GSAP几乎可以为任何属性制作动画
    - ✓ 包括 CSS 属性、元素属性、自定义对象属性。
    - ✓ 甚至 CSS 变量和复杂的字符串。
    - ✓ 最常见的动画属性、变换和不透明度等。
  - GSAP还专门给CSS形变（transform）相关属性提供了简写，如右图所示：
    - ✓ 官网形变文档：https://greensock.com/get-started/#transformShorthand

GSAP	CSS	Explanation
x: 100	transform: translateX(100px)	Move horizontally (px or SVG units)
y: 100	transform: translateY(100px)	Move vertically (px or SVG units)
xPercent: -50	transform: translateX(-50%)	Move horizontally (percentage of element's width)
yPercent: -50	transform: translateY(-50%)	Move vertically (percentage of element's height)
rotation: 360	transform: rotate(360deg)	Rotate (degrees)
scale: 2	transform: scale(2, 2)	Increase or decrease size
transformOrigin: "0% 100%"	transform-origin: 0% 100%;	The center of translation, this will rotate around the bottom left.

GSAP 动画时间线（TimeLine）

- **什么是动画时间线（TimeLine）：**
  - 时间线（TimeLine）是用来创建易于调整、有弹性的动画序列。
  - 当我们将补间添加到时间线（Timeline）时，默认情况下，它们会按照添加到时间轴的顺序一个接一个地播放。
- **TimeLine的使用步骤：**
  - 第一步：通过gsap.timeline(vars) 拿到时间线对象
    - ✓ timeline文档： https://greensock.com/docs/v3/GSAP/Timeline
  - 第二步：调用时间线上的 Tween 动画方法，比如：form、to 等。



```
let timelint = gsap.timeline({
  repeat: 2,
  yoyo: true,
});
timelint.to(["#rectangle1"], {
  scale: 0.5,
  duration: 1,
});
timelint.to("#rectangle2", {
  scale: 0.5,
  duration: 1,
});
timelint.to("#rectangle3", {
  scale: 0.5,
  duration: 1,
});
```

SVG + GSAP动画

- SVG+GSAP动画