前言

在第8章**变量计算**中使用整章篇幅介绍变量，变量作为CSS体系中最高逼格的特性，没有之一。随着浏览器日益完善，变量可大范围在项目中使用，无需关注其兼容性。

虽说变量可在纯CSS中起到领头羊的作用，但是变量的设计初衷是为了更便利CSS与JS间的联系。CSS使用变量有如下好处。

- 减少样式代码的重复性
- 增加样式代码的扩展性
- 提高样式代码的灵活性
- 增多一种CSS与JS的通讯方式
- 不用深层遍历DOM改变某个样式

本章的主题是**变量控件**，主要是基于变量与JS通讯简化基于JS逻辑的效果。也是本小册唯二两章结合JS完成效果的章节，常见控件有 放大镜 和 滚动渐变背景 。

放大镜

传统的放大镜效果需依赖大部分JS逻辑，移动和显示的效果均依赖JS，通过JS计算偏移量再渲染样式。

本次使用变量简化这些JS逻辑，将计算偏移量的逻辑整合到变量中，还记得第7章**函数计算**的 `calc()` 吗？ `calc()` 用于动态计算单位，是本次改造的核心用法。

基于上述需求，实时获取鼠标的 左偏移量 和 上偏移量 即可，而这两个偏移量是相对父节点的。通过 左偏移量 和 上偏移量 结合 `calc()` 即可计算放大镜显示内容相对父节点的显示位置。

`event` 提供以下八个偏移量，若不了解其概念很容易发生混淆。

- **screenX/screenY**：相对 屏幕区域左上角 定位，若发生滚动行为，则相对该区域定位
- **pageX/pageY**：相对 网页区域左上角 定位
- **clientX/clientY**：相对 浏览器可视区域左上角 定位
- **offsetX/offsetY**：相对 父节点区域左上角 定位，若无父节点则相对 `<html>` 或 `<body>` 定位

罗列出这些偏移量概念，发现 `offsetX/offsetY` 是最符合需求的，所以使用其作为放大镜显示内容相对父节点的显示位置。

```
<div class="magnifier" @mousemove="move"></div>
```

```
export default {
    methods: {
        move(e) {
            e.target.style.setProperty("--x", `${e.offsetX}px`);
            e.target.style.setProperty("--y", `${e.offsetY}px`);
        }
```

```
    }
};
```

接下来使用 sass 构建放大镜效果。放大镜显示内容其实就是将原图像放大N倍，通过上述偏移量按照比例截取一定区域显示内容。

先定义相关的 Sass变量 。设定放大倍率为2倍，那么被放大图像的宽高也是原来宽高的2倍。声明两个变量，分为为 --x 和 --y 。

```
$ratio: 2;
$box-w: 600px;
$box-h: 400px;
$box-bg: "https://static.yangzw.vip/codepen/gz.jpg";
$outbox-w: $box-w * $ratio;
$outbox-h: $box-h * $ratio;
.magnifier {
    --x: 0;
    --y: 0;
    overflow: hidden;
    position: relative;
    width: $box-w;
    height: $box-h;
    background: url($box-bg) no-repeat center/100% 100%;
    cursor: pointer;
}
```

还记得第9章**选择器**的伪元素使用场景吗？在这个场景下很明显无需插入子节点作为放大镜的容器了，使用 ::before 即可。

放大镜在使用时宽高为 100px ，不使用时宽高为 0px 。通过绝对定位布局放大镜随鼠标移动的位置，即声明 left 和 top ，再通过声明 transform:translate(-50%,-50%) 将放大镜补位，使放大镜中心与鼠标光标位置一致。由于声明 left 和 top 定位放大镜的位置，那么还需声明 will-change 改善 left 和 top 因改变而引发的性能问题。

```
.magnifier {
    &::before {
        --size: 0;
```

```
            position: absolute;
            left: var(--x);
            top: var(--y);
            border-radius: 100%;
            width: var(--size);
            height: var(--size);
            box-shadow: 1px 1px 3px rgba(#000, .5);
            content: "";
            will-change: left, top;
            transform: translate(-50%, -50%);
        }
        &:hover::before {
            --size: 100px;
        }
    }
```

接下来使用 background 实现放大镜显示内容。依据放大倍率为2倍，那么声明 size:$outbox-w $outbox-h ，通过声明 position-x 和 position-y 移动背景即可，最终连写成 background:#333 url($box-bg) no-repeat $scale-x $scale-y/$outbox-w $outbox-h ，而 $scale-x 和 $scale-y 对应 position-x 和 position-y ，用于随着鼠标移动而改变背景位置。

```
水平方向偏移量 = offsetX * 倍率 - 放大镜宽度 / 倍率
垂直方向偏移量 = offsetY * 倍率 - 放大镜高度 / 倍率
```

基于第10章**背景与遮罩**的 background-position 正负值问题，上述两条公式还需乘以 -1 ，则变成以下公式。

```
水平方向偏移量 = 放大镜宽度 / 倍率 - offsetX * 倍率
垂直方向偏移量 = 放大镜高度 / 倍率 - offsetY * 倍率
```

此时将两条公式代入到 $scale-x 和 $scale-y 两个 Sass变量 中，若在 calc() 中使用 Sass变量 ，需使用 #{} 的方式包含 Sass变量 ，否则会按照字符串的方式解析。

```
$scale-x: calc(var(--size) / #{$ratio} - #{$ratio} * var(--x));
$scale-y: calc(var(--size) / #{$ratio} - #{$ratio} * var(--y));
```

最终的 scss 文件如下。

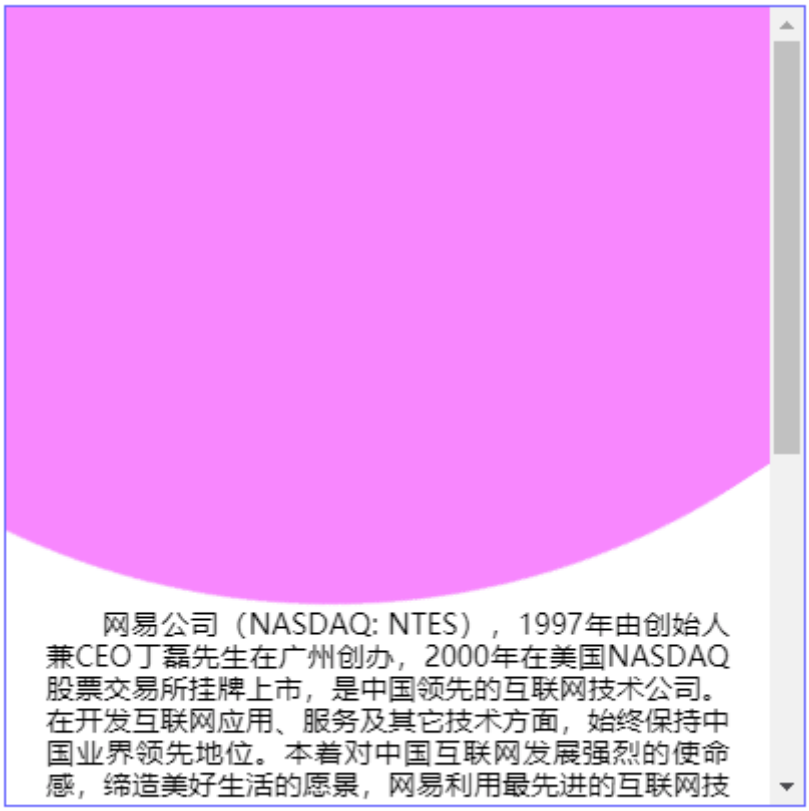
```scss
$ratio: 2;
$box-w: 600px;
$box-h: 400px;
$box-bg: "https://static.yangzw.vip/codepen/gz.jpg";
$outbox-w: $box-w * $ratio;
$outbox-h: $box-h * $ratio;
.magnifier {
    --x: 0;
    --y: 0;
    overflow: hidden;
    position: relative;
    width: $box-w;
    height: $box-h;
    background: url($box-bg) no-repeat center/100% 100%;
    cursor: pointer;
    &::before {
        --size: 0;
        $scale-x: calc(var(--size) / #{$ratio} - #{$ratio} * var(--x));
        $scale-y: calc(var(--size) / #{$ratio} - #{$ratio} * var(--y));
        position: absolute;
        left: var(--x);
        top: var(--y);
        border-radius: 100%;
        width: var(--size);
        height: var(--size);
        background: #333 url($box-bg) no-repeat $scale-x $scale-y/$outb
        box-shadow: 1px 1px 3px rgba(#000, .5);
        content: "";
        will-change: left, top;
        transform: translate(-50%, -50%);
    }
    &:hover::before {
        --size: 100px;
```

```
        }
    }
```

- [x] 在线演示：Here
- [x] 在线源码：Here

滚动渐变背景

各位同学使用移动端APP应该会发现某些页面在滚动时，顶部背景颜色会发生细微的变化，该变化随着滚动距离的增大而导致背景颜色变淡。



有了上述示例作为铺垫，可清楚知道变量结合 鼠标事件 能完成更多的酷炫效果，而关键点是把鼠标的某些参数(例如 偏移量)赋值到变量上，让变量随着鼠标参数的变化而变化。主要了解该技巧，就能开发出很多变量与JS通讯的 动画关联 和 事件响应 。

由于本示例与滚动有关，那么毫不犹豫地想起了 `event.target.scrollTop` ，监听滚动事件并将 `event.target.scrollTop` 赋值到变量上即可。另外，当滚动距离超过一定时需做一些限制，例如背景颜色不再发生变化。

```
<div ref="bg" class="dynamic-bg">
    <header></header>
    <main @scroll="scroll">
        <div>
            <p>网易公司（NASDAQ：NTES），1997年由创始人兼CEO丁磊先生在广州创办，
            <p>网易公司推出了门户网站、在线游戏、电子邮箱、在线教育、电子商务、在线
            <p>网易2019全年财报显示，网易公司2019年全年净收入为592.4亿元；基于非
            <p>2019年，网易深入推进战略聚焦，坚守内容消费领域，积极布局游戏、教育、
        </div>
    </main>
</div>
```

```
.dynamic-bg {
    --scrolly: 250;
    overflow: hidden;
    position: relative;
    border: 1px solid #66f;
    width: 400px;
    height: 400px;
    header {
        --0: calc(var(--scrolly) * 2deg);
        --size: calc(1500px - var(--scrolly) * 2px);
        --x: calc(var(--size) / 2 * -1);
        --y: calc(var(--scrolly) * -1px);
        --ratio: calc(50% - var(--scrolly) / 20 * 1%);
        position: absolute;
        left: 50%;
        bottom: 100%;
        margin: 0 0 var(--y) var(--x);
        border-radius: var(--ratio);
        width: var(--size);
        height: var(--size);
        background-color: #3c9;
        filter: hue-rotate(var(--0));
        animation: rotate 5s linear infinite;
```

```
        }
        main {
            overflow: auto;
            position: relative;
            width: 100%;
            height: 100%;
            div {
                padding: 300px 20px 50px;
            }
            p {
                line-height: 1.2;
                text-align: justify;
                text-indent: 2em;
            }
        }
    }
    @keyframes rotate {
        to {
            transform: rotate(1turn);
        }
    }


    export default {
        mounted() {
            this.bgStyle = this.$refs.bg.style;
        },
        methods: {
            scroll(e) {
                const top = e.target.scrollTop;
                if (top <= 250) {
                    this.bgStyle.setProperty("--scrolly", 250 - top);
                } else {
                    this.bgStyle.setProperty("--scrolly", 0);
                }
            }
```

```
        }
    };
```

- [x] 在线演示：Here
- [x] 在线源码：Here