【前端面试】CSS变量用过吗？如何使用？

前言

变量大家应该都听过吧，我们任何一门语言都有变量的存在。我们作为一名前端开发着，不仅仅需要写 JS 代码，还需要编写大量的 CSS 样式代码。在以前，我们所认知的变量仅仅是存在于 JS 中的，CSS 中是不存在变量这一说法的，即使存在也是通过 less 或 scss 等手段变相的实现的 CSS 变量。

但是随着前端的不断完善，我们原生的 CSS 也支持变量了，如果你还不会使用 CSS 变量，那赶紧一起来学一学吧！

1.基本介绍

变量有一个很大的作用就是降低维护成本，假如我们代码中很多地方都要用到了同一个值，这个时候我们就可以将该值使用变量维护起来，如果以后该值改变，我们只需要更改变量值即可。

作为前端开发者，编写大量的 CSS 样式代码是不可避免的，随着项目的增加，CSS 代码也会变得越来越难以维护，而且我们都知道，CSS 代码中有很多都是重复代码。

假如我们确定了项目的主题色，然后在各个 CSS 代码中使用该了主题色，如果后期主题色变了，那岂不是所有 CSS 代码都得查找更改一遍。假如我们用变量将该主题色存储起来，后续 CSS 代码中直接使用变量即可，需要更改主题色时，直接更改变量值即可。

上述的主题色场景是我们经常遇到的，这也是 CSS 变量运用的典型场景。

我们先来看看官网是如何解释 CSS 变量的。

官网解释：

> 自定义属性（有时候也被称作 CSS 变量或者级联变量）是由 CSS 作者定义的，它包含的值可以在整个文档中重复使用。由自定义属性标记设定值（比如： --main-color: black;），由 var() 函数来获取值（比如：color: var(--main-color);）

变量其实很好理解，我们需要知道的是如何声明变量以及如何使用变量，因为在各大语言中变量的声明和使用还有些许的不同，特别是在 CSS 代码中，它的声明和使用区别较大。

CSS 变量声明：

上面官网的解释中提到 CSS 变量又被称作为自定义属性，既然是属性，那么我们应该很熟悉，比如 color、background 等等都是 CSS 属性。那么我们的 CSS 变量就和属性一样，**代码如下：**

```
:root {
  --bgColor: #fff;
}
```

上段代码中我们定义了一个 bgColor 变量，需要注意的是我们需要在自定义属性前面加上--才算是变量，这算是 CSS 变量的声明规则。自定义属性值#fff 就是我们的变量值。上面的:root 是一个伪类，它相当于一个全局作用域，我们都知道变量是有作用域的概念的，声明在:root 伪类中的变量可以在全局 CSS 代码中使用。

CSS 变量使用：

声明了变量，那么只有使用它才能体现它的价值，CSS 变量的使用需要使用到一个 CSS 内置函数

var()，**代码如下：**

```
div {
  background-color: var(--bgColor);
}
```

我们只需要利用 var 函数将声明的变量包裹起来，就可以拿到它的值了。

变量命名规则：

我们都知道变量不能是随便命名的，比如变量名不得有关键词，特殊符号等等，各个语言中变量命名也有稍许不同，比如 JS 变量中不能以数字开头，但是我们 CSS 变量允许数字开头，比如下方代码：

```
:root {
  --1: #369;
}
div {
  background-color: var(--1);
}
```

但是，CSS 变量命名还是有自己规则的，总结如下：

- 不能包含$，[，^，(，%等字符
- 普通字符局限在只要是“数字[0-9]”“字母[a-zA-Z]”“下划线_”和“短横线-”这些组合
- 可以是中文，日文或者韩文
- 无论是变量的定义和使用只能在声明块{}里面
- 变量值只能用作属性值，不能用作属性名，比如下方代码我们是不允许的：

```
.foo {
```

```
  --side: margin-top;
  /* 无效 */
  var(--side): 20px;
}
```

2 .var函数使用

前面我们说 CSS 变量的使用要依靠 var 函数，那么 var 函数具体怎么使用呢？我们本章详细介绍一下。

官方解释：

> var()函数可以代替元素中任何属性中的值的任何部分。var()函数不能作为属性名、选择器或者其他除了属性值之外的值。（这样做通常会产生无效的语法或者一个没有关联到变量的值。）

官方解释得还是很好理解的，我们可以总结出下面两点：

- 可以替代 CSS 中任何属性的值得任何部分
- 它不能作为属性名、选择器等等，总之只能用在属性值部分

基本语法：

既然 var()是一个函数，那么它就有输入输出值，也就是参数和返回结果。var()函数比较简单，它只接收两个参数，代码如下：

```
var( <custom-property-name> , <declaration-value>? )
```

可以看到 var()函数接收两个参数，具体表示啥意思我们一起来看看。

参数解释：

- <custom-property-name>：自定义属性名，也就是我们自定义的变量。
- <declaration-value>：可选参数，如果我们自定义的变量值无效的话，该参数就作为回退值使用，也就是 var 函数的默认值。

简单示例：

当我们的变量值有效时，会直接采用变量值。

```
<head>
  <style>
    :root {
      --bgColor: rgb(82, 70, 70);
    }
    div {
      background-color: var(--bgColor, green);
      width: 100px;
      height: 100px;
    }
  </style>
```

```
</head>
<body>
  <div></div>
</body>
```

输出结果：

当我们变量值无效时，采用默认值，**代码如下：**

```
<head>
  <style>
    div {
      background-color: var(--bgColor, green);
      width: 100px;
      height: 100px;
    }
  </style>
</head>
<body>
  <div></div>
</body>
```

输出结果：

3.变量的作用域

通常来说，变量声明后都会有作用域的，不可能我们随便声明一个变量，然后整个系统都可以用吧！这样整个系统就乱套了！我们的 CSS 变量同样也有作用域的概念。

3.1 全局变量

前面我们将 CSS 变量声明在了:root 伪类中，这个时候的变量就是全局变量，即所有 CSS 样式代码中都可以使用到该变量。

代码如下：

```
:root {
  --bgColor: rgb(82, 70, 70);
}

div {
  background-color: var(--bgColor);
  width: 100px;
  height: 100px;
}
```

上段代码中--bgColor 就是全局变量

3.2 局部变量

我们也可以将变量的声明放在某一个选择器内部，这个时候该变量就是一个局部变量，属于该元素的所有子元素都可以使用到该变量。

代码如下：

```
<head>
  <style>
    div {
      --color: blue;
      width: 100px;
      height: 100px;
    }
    p {
      color: var(--color);
    }
    span {
      color: var(--color);
    }
  </style>
</head>

<body>
  <div>
    <p>小猪课堂</p>
  </div>
  <!-- 无法使用 div 内声明的变量 -->
  <span>我在 div 外面</span>
</body>
```

输出结果：

小猪课堂

我在div外面

我们可以看到 span 标签无法使用到我们定义的--color 变量，因为该变量属于 div 这个作用域，只能作用域 div 以及它的子元素。

3.2 变量优先级

当我们有多个 CSS 变量的名称一致时，该如何取变量的值呢？我们这里可以简单给个规则：

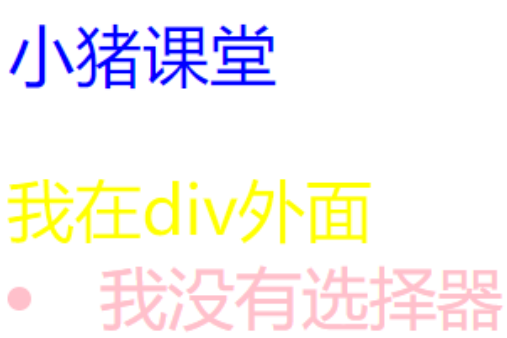
> 如果在多个 CSS 变量命名相同，且都能读取到的情况下，根据 CSS 选择器的优先级来选择合适的变量值。

示例代码如下：

```
<head>
  <style>
    :root {
      --color: yellow;
    }
    * {
      color: pink;
    }
    .classp {
      --color: blue;
      color: var(--color);
    }
    span {
      color: var(--color);
    }
  </style>
</head>

<body>
```

```
  <p class="classp">小猪课堂</p>
  <!-- 无法使用 div 内声明的变量 -->
  <span>我在 div 外面</span>
  <li>我没有选择器</li>
</body>
```

实现效果：



上段代码中我们使用了通配符*给所有的字体颜色都设置为了 pink，但是最终只有 li 标签的字体颜色为 pink，是因为 li 标签没有单独设置样式，而其它标签都利用选择器单独设置了样式，所以颜色不一样。

上段代码还需要注意的是.classp 选择其中的--color 变量将:root 中的变量覆盖掉了，这就和 css 样式优先级一个道理。

总之大家判断 CSS 变量优先级的时候参考 CSS 样式的层级规则就可以了。

4.关于变量的拼接

有时候我们定义的变量可能是数字，可能是字符串等等，当我们使用变量的时候可能需要做一些处理，比如数字要与字符串拼接等等，这个时候我们需要注意什么呢？这里给出一些常见的需求以及功能点供大家参考。

变量是数值时

当我们的变量是数值时，且使用时需要与字符串拼接，这是我们需要做一些处理才可以，**代码如下：**

```
:root {
  --color: yellow;
  --height: 100;
}

/* 2 种错误的写法 */
div {
  height: var(--height)px;
```

```
  height: var(--height) + 'px';
}

/* 正确的写法 */
div {
  height: calc(var(--height) * 1px);
}
```

变量值是字符串

如果我们的变量值本身是一个字符串，那么它是可以和其它字符串拼接的，**代码如下：**

```
--bar: 'hello';
--foo: var(--bar)' world'; // hello world
```

变量值带有单位时

变量值带有单位是比较常见的，很多小伙伴可能觉得带有单位就应该当做字符串处理，其实不是，因为 css 内部本身本身就能解析单位，我们不必再使用引号将单位包装成字符串，**代码如下：**

```
/* 错误的写法 */
div {
  --foo: '20px';
  font-size: var(--foo);
  /* 无效 */
}

/* 正确的写法 */
div {
  --foo: 20px;
  font-size: var(--foo);
}
```

5.换肤原理

换肤是 CSS 变量实践的一个典型场景，我们通常使用 Less 或者 Sass 与 CSS 变量结合的方式来实现换肤，我们这里直接使用 CSS 变量来演示换肤最简单的原理。

代码如下：

```
<head>
  <style>
    :root {
      --theme: yellow;
    }
    .box {
      width: 100px;
      height: 100px;
      background-color: var(--theme);
    }
```

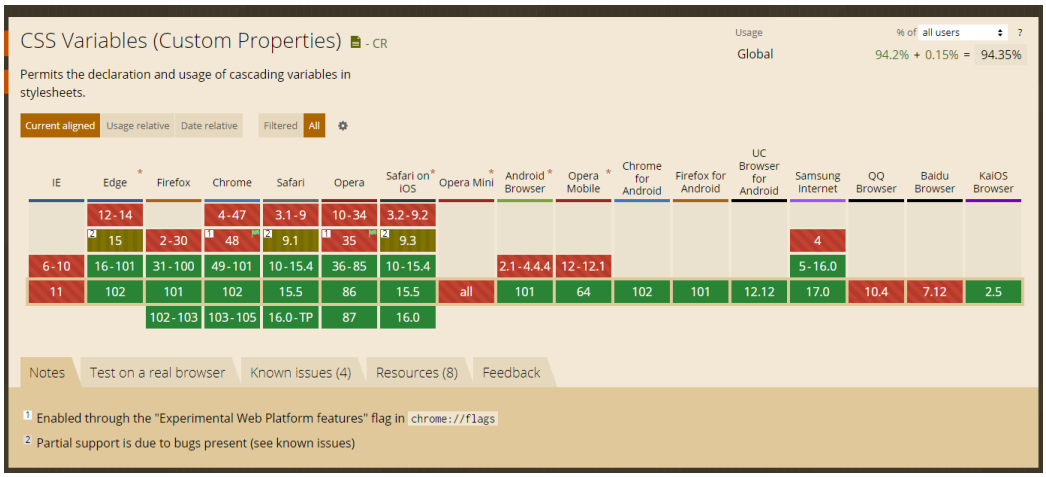
```
  </style>
</head>

<body>
  <div class="box"></div>
  <button id="btn1">切换黑色主题</button>
  <button id="btn2">切换黄色主题</button>
</body>
<script>
  let box = document.getElementsByClassName('box')[0];
  let btn1 = document.getElementById('btn1');
  let btn2 = document.getElementById('btn2');
  btn1.addEventListener('click', () => {
    box.style.setProperty('--theme','#000');
  })
  btn2.addEventListener('click', () => {
    box.style.setProperty('--theme','yellow');
  })
</script>
```

实现效果：

上段代码虽然很简单，但是它基本上解释了换肤的原理是什么，我们通过点击按钮实现 div 背景色的变化。如果我们再很多元素上都使用了设置的 theme 变量，那么更改--theme 属性值的时候，就相当于切换了主题。

6.兼容性

由于 css 变量出现的时候并不是那么早，所以可能存在一些兼容性问题。但是最近 IE 浏览器都倒闭了，我们何必还要担心兼容性问题呢。



总结

虽然我们 CSS 变量可以实现换肤功能，但是我们在实际项目中通常采用的是 CSS 变量与 less 或者 scss 结合方式，因为我们的项目通常是 Vue 或者 react 项目，所以我们需要合理的借助一些工具。