

# Naming Tokens in Design Systems

- 正文
- 从基础开始
- 基础 ( Base Level )
  - 类别 ( Category )
  - 属性 ( Property )
  - 概念 ( Concept )
- 调整 ( Modifiers Level )
  - 变体 ( Variant )
  - 状态 ( State )
  - 量表 ( Scale )
  - 模式 ( Mode ) : Light/Dark
- 对象 ( Objects )
  - 元素 ( Element )
  - 组件 ( Component )
  - 组件组 ( Group )
- 命名空间 ( Namespace )
  - 系统 ( System )
  - 主题 ( Theme )
  - 域 ( Domain )
- 本文小结

作者：Nathan Curtis

链接：<https://medium.com/eightshapes-llc/naming-tokens-in-design-systems-9e86c7444676>

名词解释：

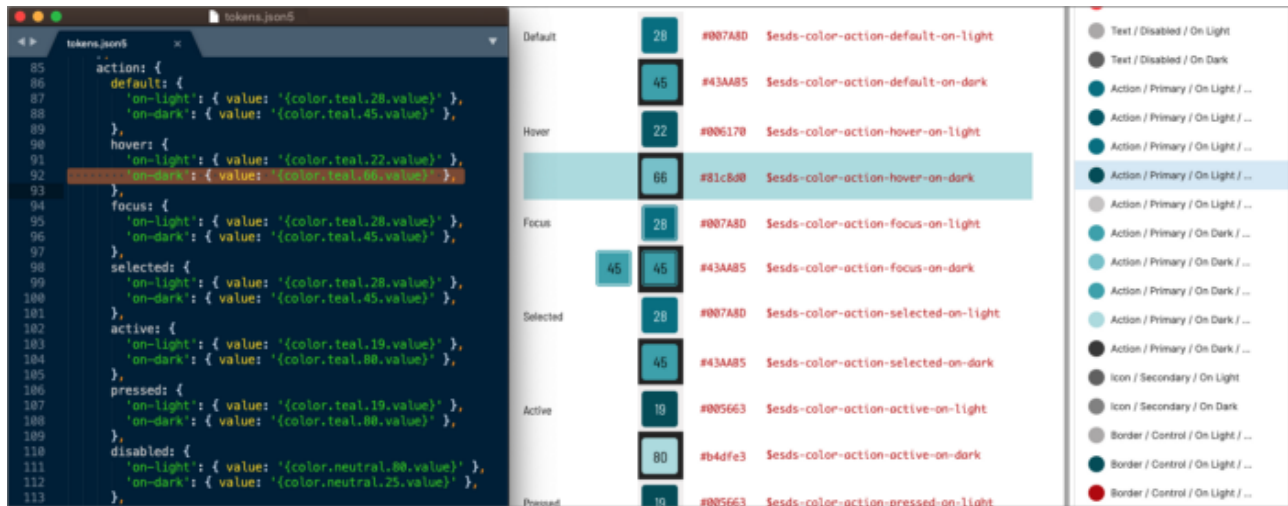
Token	Tokens 的本意是“令牌 / 指令”，我们常听说的 Design Tokens 是一种适合设计师和开发共同使用的工作思维和方法，与 Naming 连起来可以被理解为“命名变量”
Level	级别，本文指变量定义的分类级别

概念字典：

Salesforce				Nathan Curtis	
Component	组成部分，比如原子组成分子，原子就是分子的组成部分	System	在本文的语境下是类似 Namespace 命名空间的概念	Namespace	命名空间：系统、主题、域
Category	类别、种类	Category	类别、种类	Object	物体：元素、组件、模块
Property	属性	Concept	概念	Base	基础组成部分：类别、概念、属性
Attribute	性质，这个和上面的区别还不明确	Property	属性	Modifier	调整：变体、状态、量表、模式
Relationship	关系	Variant	变体		
State	状态	Scale	量表		

## 正文

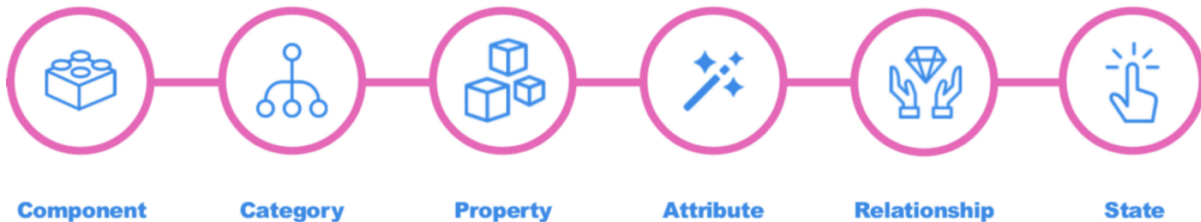
设计变量这一概念自 Salesforce 于 2014 年提出以来，已经为许多设计系统提供了视觉基础。我（Nathan Curtis）也在 2016 年就对它十分有兴趣，写过关于它的文章，在此之后我花了越来越多的精力在它上面。随着视觉风格系统在组件、平台和输出领域的广泛传播，设计变量-以及它们的命名规范-变得越来越重要。



代码（左）、文档（中）和设计（右）的设计变量。命名并不完美：你能发现不一致之处吗？

	一	~ / 二	~ / 三
代码	action	default, hover, focus, selected, active, pressed, disabled	on-light, on-dark
文档	action	default, hover, focus, selected, active, pressed, disabled	on-light, on-dark
设计	Action Primary	on Light, on Dark	Default, Hover, Focus, Selected, Active, <del>Pressed</del> , <del>Disabled</del>

随着变量变得越来越复杂，命名模式变得越来越重要。Salesforce UX 的 Brandon Ferrua 和 Stephanie Rewis 和 Shopify UX 的 Kaelig Deloumeau-Prigent 展示了他们的模型。Adobe Spectrum 和 Danny Bank 的 Style Dictionary 变量工具也记录了他们的模式。

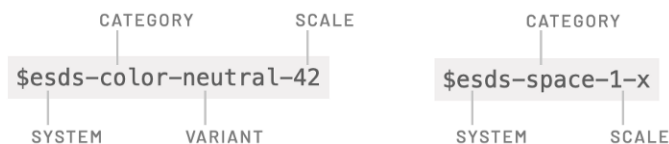


Salesforce UX 变量层次结构 组件分类属性性质关系状态

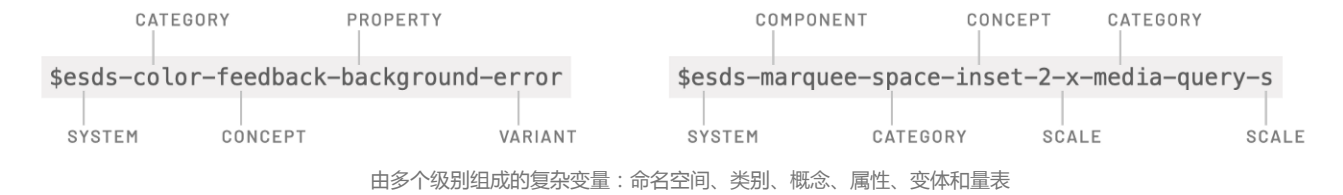
2014	Sönke Rohde - VP UX R&D at Salesforce.	Living Design System	<a href="https://medium.com/salesforce-ux/living-design-system-3ab1f2280ef7">https://medium.com/salesforce-ux/living-design-system-3ab1f2280ef7</a>
2016	Nathan Curtis	Tokens in Design Systems	<a href="https://medium.com/eightshapes-llc/tokens-in-design-systems-25dd82d58421">https://medium.com/eightshapes-llc/tokens-in-design-systems-25dd82d58421</a>
2019	Brandon Ferrua、Stephanie Rewis - Salesforce UX	Designing Standard Systems Clarity 2019	<a href="#">Designing Standard Systems Clarity 2019.pdf</a>
2020	Kaelig Deloumeau-Prigent - Shopify UX	Laying Foundations for Quality	<a href="https://www.youtube.com/watch?v=forQopU1ESE">https://www.youtube.com/watch?v=forQopU1ESE</a>

## 从基础开始

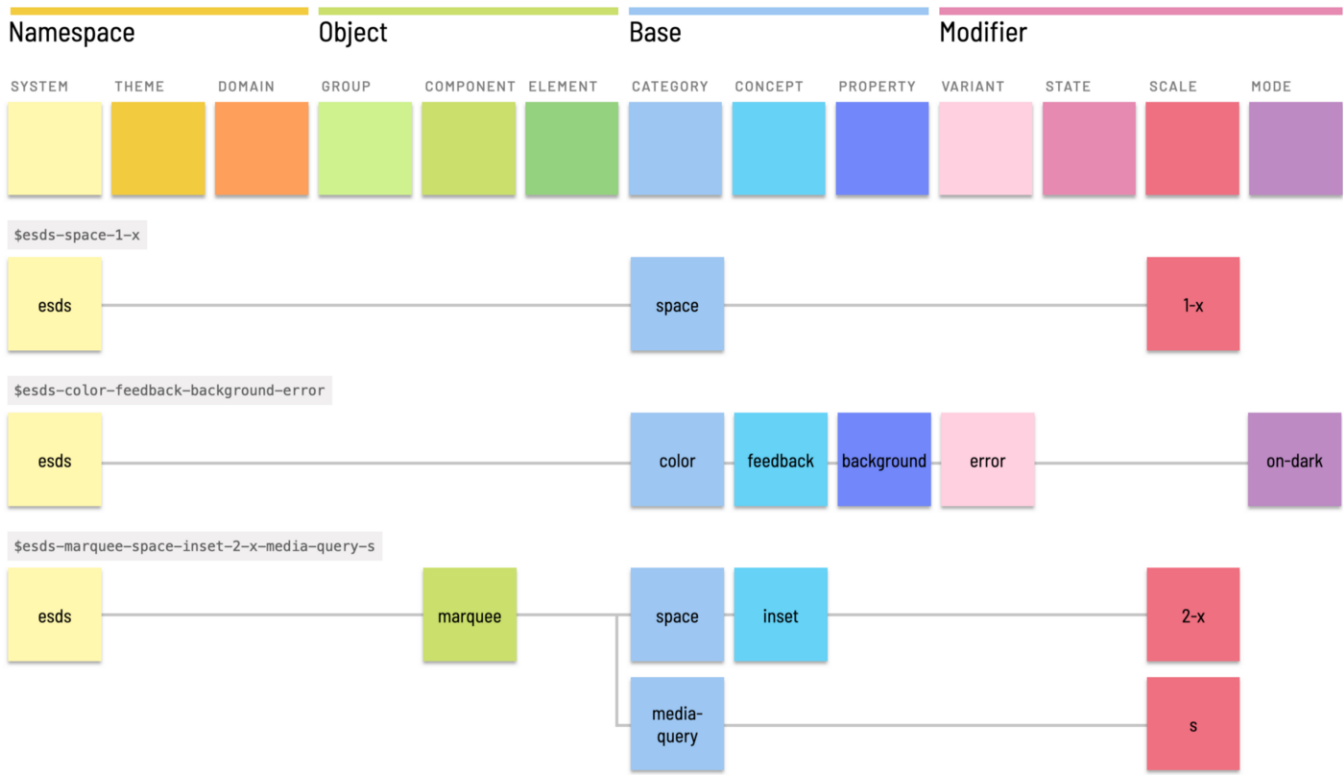
即便是最简单的设计变量也是通过级别展示命名模式。



例如，【\$esds-color-neutral-42】结合了四个级别——命名空间（system）（在本例中，esds代表“EightShapes 设计系统”）、类别（category）、变体（variant）和映射到的量表（scale）来表示颜色【#6B6B6B】。类似地，【\$esds-space-1-xnamespace）、类别（category）和量表（scale）结合起来表示的是字号【16px】



除了这些通用类型，我们也需要更多的级别来保证设计变量能够达到我们的期望，即能够集中记录并且作为在设计决策领域能广泛应用的一种方式。但是更多的级别能使记录更具体，但是也会导致更复杂，例如【\$esds-marquee-space-inset-2-x-media-query-s】这个合并的变量就包括了 2 个类别和量表。



为了描述清楚，包含了分类学和类型学的变量化语言需要很多级别来表示。不同的级别，我们可以做的就是，将它们组织成组：

- **Base**：基本级别，作为变量的主干，例如类别、概念和属性
- **Modifier**：调整级别，例如变体、状态、量表和模式
- **Object**：物体级别，从最基础的 Element（元素例如图标），到 Component（组件例如按钮）再到复杂的 Group（组件组例如表单）
- **Namespace**：命名空间级别，在极端情况下，可以包括系统、主题和域内的所有内容



不同公司内主要操作悬停颜色的变量

不同设计系统采用的级别与分类方式不同，上图是 6 个公司对于操作悬停颜色的不同表述方式，它们分别从不同的深度、特异性、顺序和意图来表示，我尽力了，但是我找不到正确的变量方式，这也是我们现在面临的问题。

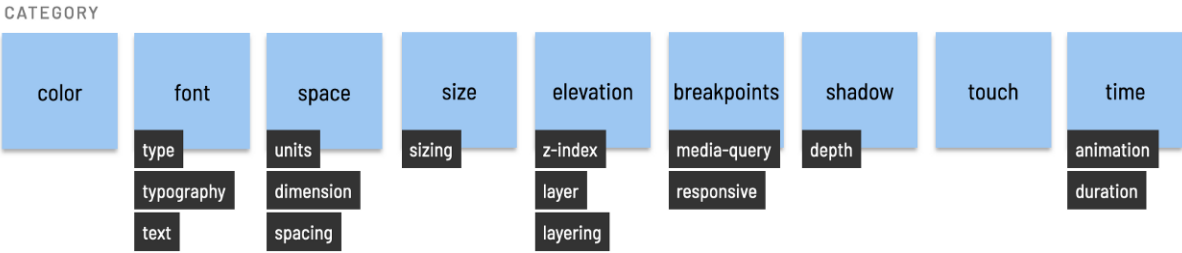
本文会对基于【Base、Modifier、Object、Namespace】的级别与分类方式来进行展开讨论，在此过程中，完整性、顺序和多层次原则是我们要面临的挑战。

## 基础 ( Base Level )

类别和属性为大多数的变量提供了一个基本出发点，但是随着集合的增长，单个类别级别被证明是不够的，因此变量子集会被组织称为概念。

## 类别 ( Category )

类别中的基础变量包括颜色、字体和间距。



相关变体术语的常见类别

类别涉及到视觉风格的各个方面并且它们经常重合，除了典型的比如颜色外，不同系统命名类别有不同的方式，常见的有：

- color
- font ( 又名 type , typographytext
- space ( 又名 units , dimensionspacing
- size ( 又名 sizing )
- elevation ( 又名 z-index , layerlayering
- breakpoints ( 又名 media-query , responsive )
- shadow ( 又名 depth )
- touch

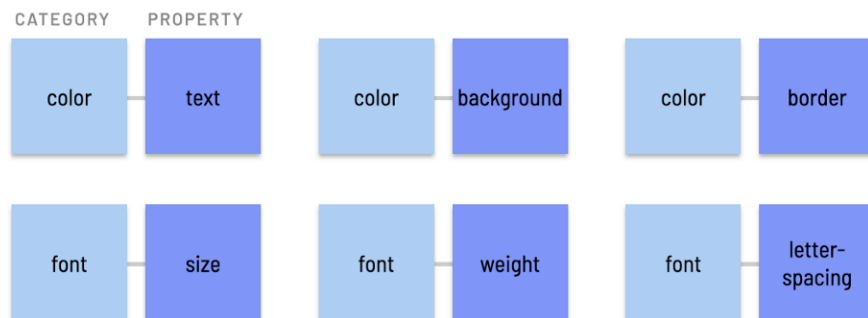
- time (又名 animation, duration)

如上图所示，优先使用开头的首选术语（比如 font），如果想要突出或者有别于其他系统，可以使用括号里的等效术语（比如 type 和 typography）。希望这些等效术语能帮助您和您的团队在形成受控词汇表时提供灵感。

**使用原则：避免同音字**，即使是首选术语也会引发艰难的选择。type 是同音异义词，被解释为许多不同的东西，例如排版的速记或类别。后者对于变量名很麻烦！类似的，一些替代品比如 text typography 也是内容和属性的同义词（稍后会详细介绍）。由于 typography 太长，团队通常最终会选择 font

## 属性 (Property)

属性可以和相关类别配对来共同定义一个变量，尽管该配对不足以定义具体的值。



类别/属性示例对

与类别 color 属性包括 text、backgroundborder fill，由于缺乏上下文只能作为基础的变量，例如：

**i** \$color-background: #FFFFFF  
\$color-text: #000000  
\$color-border: #888888

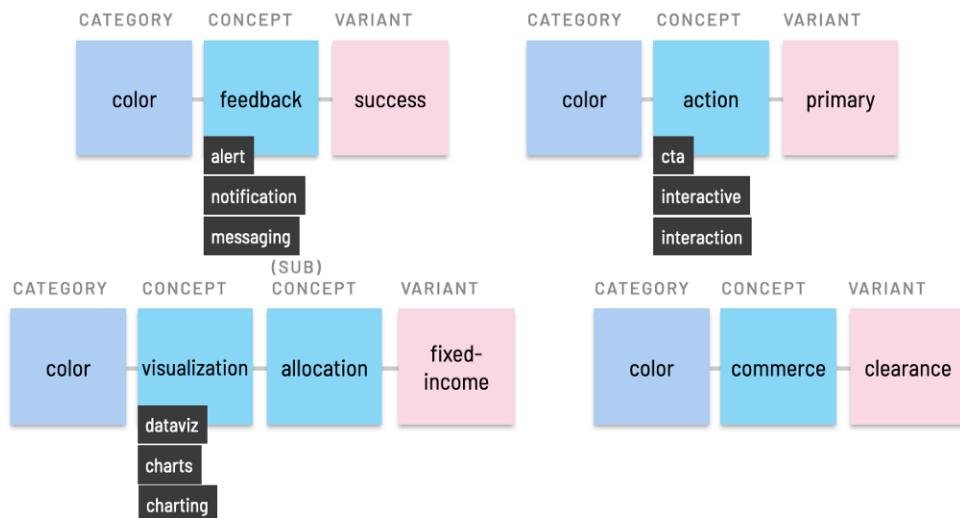
常见的 font 属性包括 size、weight、line-height 和 letter-spacing，从而产生如下变量：例如：

**i** \$font-weight: normal  
\$font-size: 14px  
\$font-line-height: 1.2

类别与属性的配对非常普遍，并没有故意为之。我们需要概念和调整级别。

## 概念 (Concept)

通过添加一个或多个概念，我们可以按类别对变量进行分组。

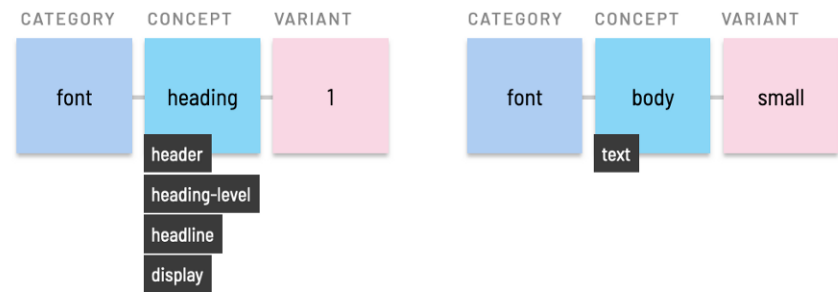


概念级别变量示例

例如，color 可以分为以下概念：

- feedback ( 又名 notificationmessaging、alert ) 带有successwarning 和error
- action ( 又名 ctainteractiveinteraction ) 来划分颜色，提供号召性用语（链接、按钮.....）和选定项目（如选项卡、导航项、复选框、单选按钮和过滤器）。
- visualization ( 又名datavizchartingcharts。金融晨星设计系统甚至包括子概念 visualizationfor correlation、valuation、performance 和 asset-allocationcolors 以及默认的可视化颜色 order
- commerce带有sale、clearance、inventory 和 urgency 变体的颜色 timing

概念与变体结合形成如【\$color-feedback-success\$color-action-primary和【\$color-visualization-performance-positive



概念与变体结合示例

类似地，typography 变量通常分为 heading ( 又名 header、heading-levels、headline、display ) 和 body ( 又名 text 一同音词！ ) 之类的概念。

eyebrow 标题或 lead lede、deck、subheadersubhead) 之类的特殊情况与 headings ( 1, 2, 3, ...) 和 body ( s, m, l ) 概念不同。这样的命名挑战暗示了变体和量表级别的分类也需要获得足够具体的名称。

**原则：内同质，间异质，**在各个层面，尤其是在**概念**上，都力求类内的同质性（如 visualization）和类间的异质性（如visualizationVS commerce）。为了保持低概念数量，您可以将 sale clearance 融入 visualization。但是，适用于电子商务流程 visualization 中的 sale 和 clearance。则因为给定了不同的含义，它们被分成了不同的概念。

## 调整 ( Modifiers Level )

通过对**变体**、**状态**、**量表**和**模式**的调整来达到效果，调整可以独立使用或者协同使用，与**类别**、**概念**、**属性**级别配对使用，形成有使用价值与决策意义的文本模式。

### 变体 ( Variant )

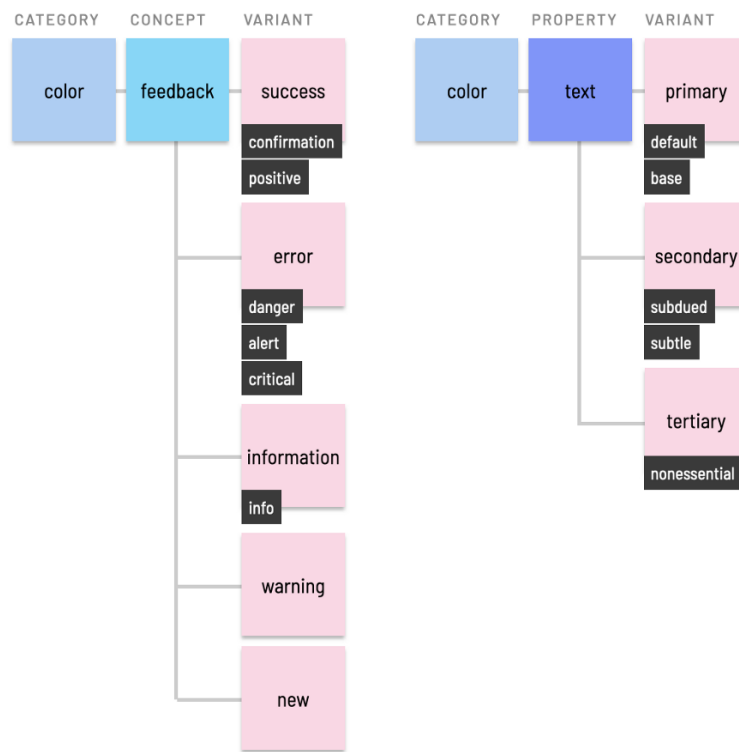
变量变体用量区分不同的使用例子。例如，设计系统通过不同的层次结构和对比度来创建不同的文本颜色：

- primary ( 又名 defaultbase )
- secondary ( 又名 subduedsubtle )
- tertiary ( 又名 nonessential )

同样的，通过界面反馈的方式去提醒用户：

- success ( 又名 confirmationpositive )
- error ( 又名 dangercriticalalert )
- information ( 又名info )
- warning
- new

\$color-text-primary\$color-background-warning\$color-fill-new将类别/属性对与变体组合在一起的例子。



调整级别变量示例

## 原则：更灵活还是更有指向性？

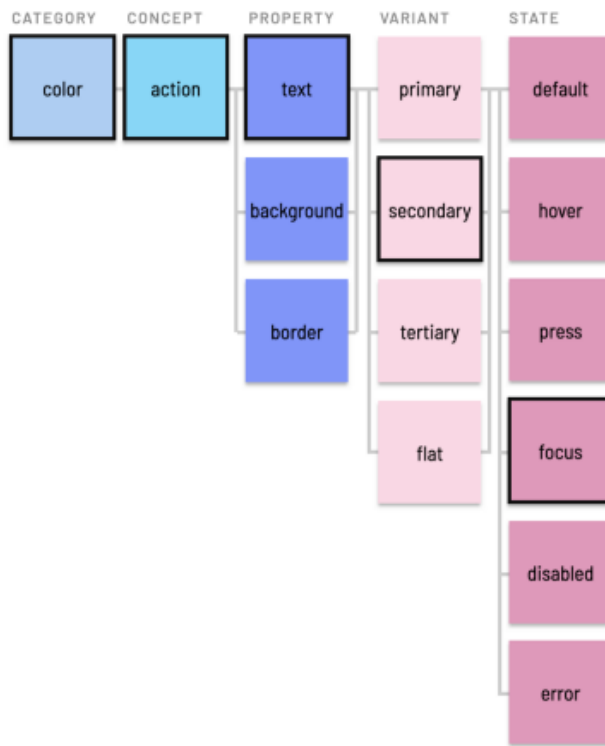
诸如【`$color-success`】将类别 (color) 和变体 (success) 组合为适用于许多场景的变量。用户可以在任何 `background` 或 `text` 上使用 `$color-success`

```
-- success textbackground success textbackground border$color-background-success$color-text-success
```

## 状态 ( State )

变量状态可以基于不同的交互状态来指定属性，例如：

- default
- hover 当指针位于对象上方时
- press/ active, 在用户按下和释放对象的时间之间
- focus 当一个对象能够接受输入时
- disabled 当一个对象不能接受输入时
- visited 用于已访问时的替代链接显示
- error 当对象处于错误状态时



状态级别变量示例

状态通常将对象( button) 或类别( color)、概念( action) 和属性( text) 元组与变体( secondary) 相关联。这会产生一个完全体的变量，如【\$color-action-text-secondary-focus

## 量表 ( Scale )

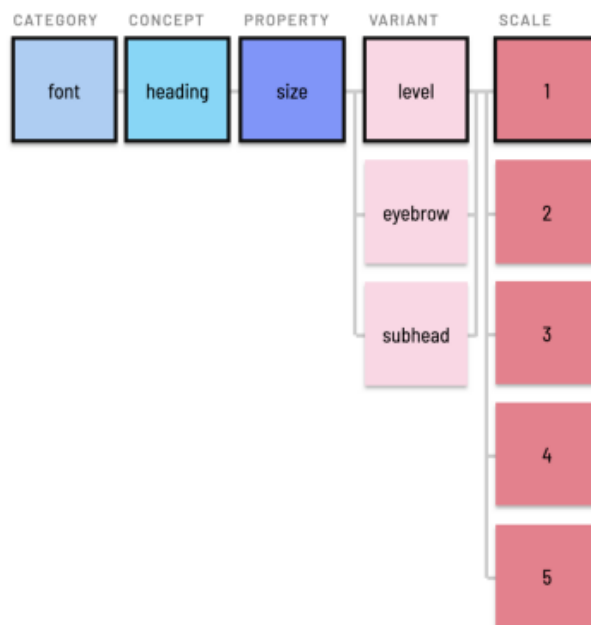
变量量表应用于事物和事物之间的不同大小、空间和其他描述事物间关系的场景。常见的量表类型包括：

- 枚举值，如标题级别 1、2、3、4和5
- 有序值，如Google Material 颜色级别。50100900
- 有界刻度，例如 HSL 的 0 到 100 亮度值，以改变色调的深浅，例如石板灰色的 slate-42、slate-90 和 slate-95。
- 比例，往往建立一个基数1-x，相对增长 ( 2-x, 4-x, ... ) 和缩小 ( half-x, quarter-x, ... )。
- T 恤尺寸，以 small(variants: s)、medium (variants: m, standard, base, default) 和 large(variant: l) 开头，然后扩展为xlxs和xxxl。专业提示：尺寸≠空间，所以尽管我在四年前说过，请考虑使用比例而不是 T 恤来获得空间。

量表同时体现在通用和具体的变量中。例如，大多数系统定义了通用（又名原始）间隔符，例如【\$esds-space-2-xfor 32px例如【\$esds-color-neutral-42for 颜色 #6B6B6B，它们被称为空间和颜色的具体使用。在这种情况下，2-x 42 坐在比例和亮度的尺度上。

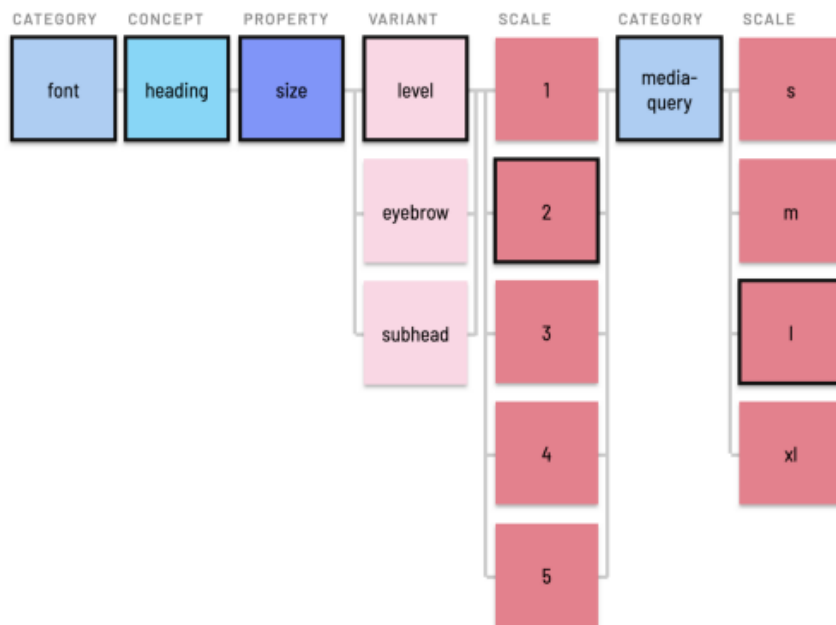
具体的明确的变量在类别（ font ）、概念（ heading ）和属性（ size ）的上下文中明确量表（ level-1 ），比如【\$esds-font-size-heading-level-1 \$esds-font-size-body-small





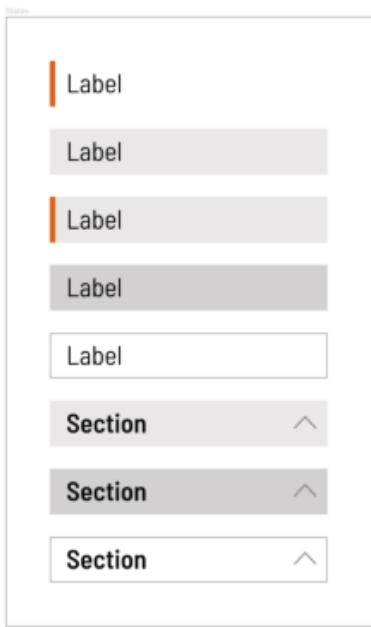
量表级别变量示例

不太常见的是，明确的变量描述需要将概念/量表对链接到一个变量中。例如，响应式排版系统可以将 heading 枚举 level(1, 2, 3, ...) 与媒体查询断点 (media-query) 组合为 T 恤尺寸 (m, l, ...) 来存储 45px

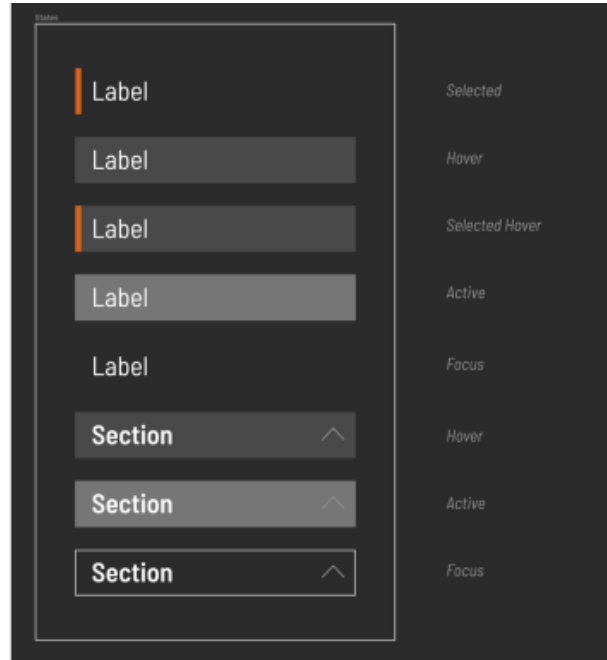


## 模式 ( Mode ) : Light/Dark

模式量表用来区分元素出现的两个或多个表面/背景设置中的值。这启用了不同的 light 模式和 dark ，也可以延伸到其他比如 brand-color EightShapes

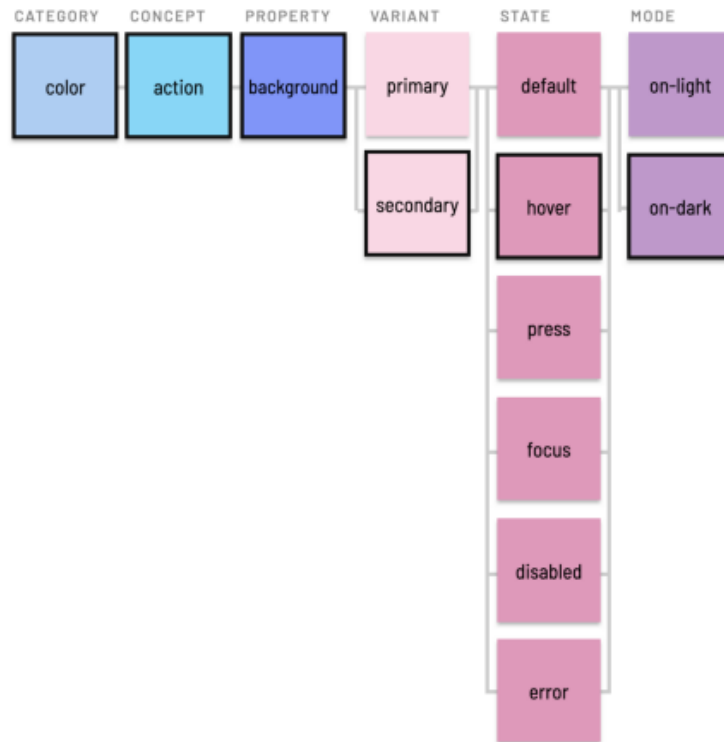


Selected  
Hover  
Selected Hover  
Active  
Focus  
Hover  
Active  
Focus



UI

例如，您可能需要两个变量【`$color-action-background-secondary-hover-on-light`】和【`$color-action-background-secondary-hover-on-dark`】来区分垂直导航、垂直过滤器和水平选项中项目的悬停背景颜色。

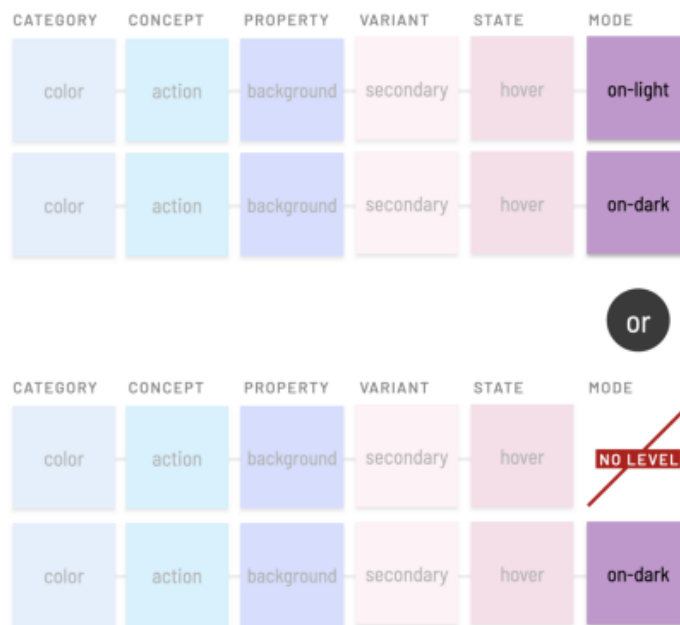


颜色模式级别变量示例

这增加了变量的数量，但在实践中仅限于小的子集，可以通过将更简单的值（比如 `$color-accent-hover-on-dark`）重复使用来降低其的复杂性。

**原则：默认值显示还是默认值截断？**

模式变量可以假定默认背景（通常是“浅色”或白色），这样只用将需要“深色”的地方加上 `on-dark` 变量，以免在模式变量不相关的地方也要加上 `on-light`。但是另一方面，使用这种在语言上有关联关系的变量比如 `on-light`、`on-dark`、和 `on-brand`



命名对比：包括还是排除默认值

原则：包括还是排除修改条款？

通过在诸如 dark light -on Nathan Curtis Cap Watkin Sliding Scale



命名对比：包括还是排除默认值

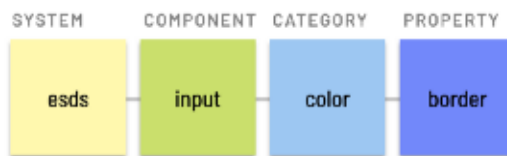
## 对象（Objects）

变量通过对分类名的使用来定义本身，然而有时候某一变量只需要在少数或者甚至一个变量里面使用。在组件或者组件组中的元素，我们可以用对象变量来定义它们，这种使用方式对类似于表单这种有一系列组件集群的场景下通常有帮助。

比如说，一个输入框组件存在着多种变量，比如说用【`$sds-color-text-primary`】来定义文本的色值，但是对于另外一个变量集合来说，可能就不需要用到输入框的边框颜色与圆角，使用通用的变量比如【`$sds-color-neutral-70`】或者直接显示值比如 **40PX** 就能满足了，不是吗？

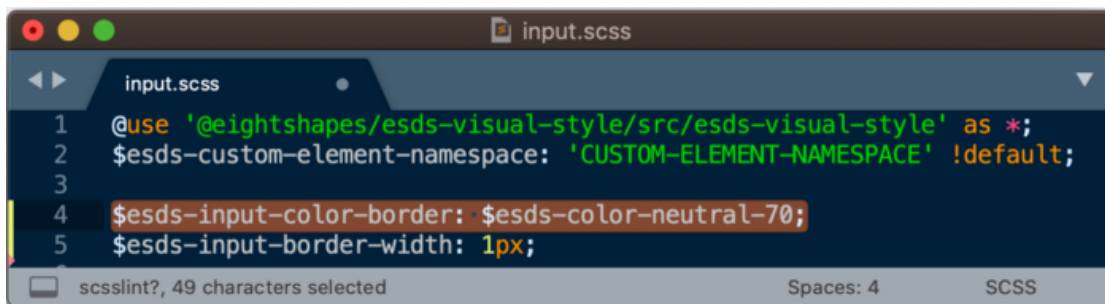
## 元素（Element）

边框颜色可能在很多地方都需要使用，但是我们并不能保证能定义出到底是否要用的规则，在这种不确定的情况下，我们希望有一个地方来记录着这特殊的元素组件，【`$sds-input-color-border`】，但是我们要在哪里放它呢？



元素级别变量示例

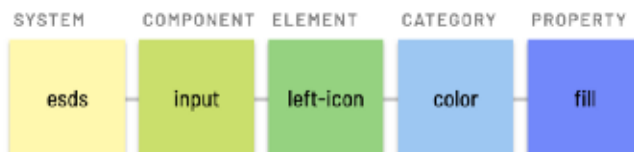
一个全局级别的变量不是一个好的方式，可以替代的方式是，将其记录在在特定于该组件的位置，例如输入框的设计规范或者类似【input.scss文件的标题。这些地方用便于使用的传统名称来记录使用方式，这也对于之后类似的情况具有参考意义



input.scss

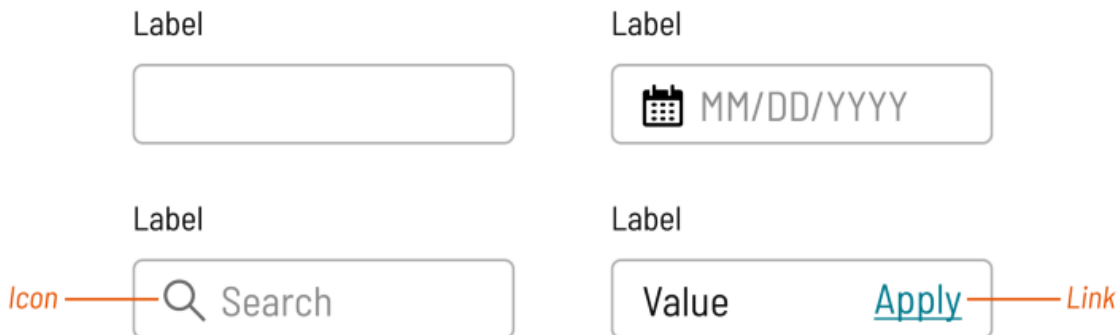
## 组件 (Component)

即便是像输入框着这原子类的组件最终也可能包含图标和链接等嵌套元素，这也是需要重复使用的候选者。



组件级别变量示例

特定于嵌套元素使用的变量可能包含组件名称和元素名称，就像 BEM CSS 方法论一样，为元素制定的变量，比如【\$esds-input-left-icon-color-fill】【\$esds-input-left-icon-size\$esds-input-inline-link-color-text】也可以放置在本地文本例如设计规范或者类似【input.scss



## 组件组 (Group)

即便是像表单（又名UI-控制器，或者表单控制器）这种组件组里面的变量，不仅自己会使用，在其他相关的组件组里面也会使用，这些使用方式共同构成该变量的意义。比如，选择器、复选框和单选框的描边也会用的到【\$esds-color-neutral-70】的变量描述。

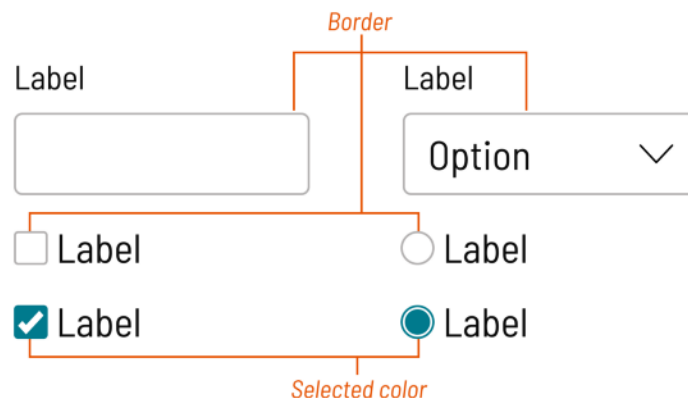


组件组级别变量示例

由于变量的写法往往取决于多个组件（我的建议是使用3个或者以上的组件），我们可以这样做

1. 添加【\$esds-forms-color-border】为全局变量

2. 将【`$esds-input-color-border`】替换为【`$esds-forms-color-border.`】
3. 在【`input.scss`】文件里删除【`$esds-input-color-border`】
4. 将变量【`$esds-input-color-border`】应用在选择器、复选框和单选框上



原则：从局部开始，慢慢延伸到跨组件

在探索实践过程我们发现，在推广命名方式的时候从局部到整体是比较合适的方式。

原则：不要太快的决定全局变量

不同组件之间享有相同的元素类型是可以遇见的一件事情，这些元素被放在一个分类下面，并且通常是可以很快被约定下来的。

但是有一个情况不是那么明显，比如当我们来定义组件 `tooltip` 时，很容易就想到接下来是组件 `popover` 和组件 `meun`。一个系统大概率会使用阴影和凹口圆角，但是也不能完全保证。在这种情况下，在局部使用 `tooltip` 特指型的变量，并在之后 `popover` 和 `meun` 需要使用它时引用。这可以避免烦人的主观辩论（“这些是【`notched-layers!`”），过早地打乱我们的全局命名空间规则。

## 命名空间（Namespace）

在小规模团队工作中，命名空间级别的变量无需担心它和其他级别变量的协作与交互，但是，考虑到变量的存在就是为了之后在多平台和各种范围的空间里面传播的，在系统、主题或者域之上加上命名空间是有必要的。

### 系统（System）

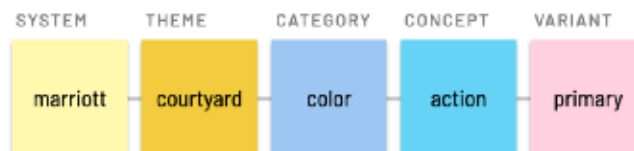
许多系统使用以下命令预设方式命名空间级别：

- **系统命名**：像是【`comet-`】和【`orbit-`】、简短的系统名称（例如五个字符或更少）通常效果很好。否则，.....
- **长名称的首字母缩写**：像是【`slids-`】（for Salesforce Lightning Design System）和【`mds-`】（for Morningstar Design System）。

从结果来说，命名空间【`esds`】在【`$esds-color-text-primary`】和【`$esds-font-family-serif`】上的应用可以很好的让我们追踪与区分相关团队的变量。

### 主题（Theme）

系统通常会提供一个主题，主题会在组件目录中改变颜色、版式或其他样式。比如像万豪这样的组织可能会为 JW Marriott、Renaissance、W、Courtyard 和其他酒店定义主题。主题的目的主要是在现有的变量主题上提供额外的可扩展能力使之能够快速的覆盖其他同类情况。对于万豪庭院来说，这可能就是在按钮、复选框和被决定好的选项卡（突出显示的）等组件上应用可变品牌颜色。（比如 `#a66914`）



主题级别变量示例

主题变量的命名和可变动约定是一个相当复杂的问题，这超出了本文的范围。不同的团队以不同且通常非常复杂的方式设置工具、编译和命名结构。但是我在这里阐明一些“关于主题命名”基本原则。

比如说，假设我们现在有一个手机应用，由【\$aads-动画应用程序设计系统提供了相关的 ocean、sands、mountain 和 sunset 。每个主题都同时在局部或者全局的环境下有特殊的定义调整。

i

```
// General tokens to alias thematic colors to many contexts
$aads-ocean-color-primary
$aads-sands-color-primary
$aads-sunset-color-secondary// Specific token overrides and extensions
$aads-ocean-color-heading-text-1
$aads-sands-color-heading-text-1
$aads-mountain-color-alert-background-success
```

一种方式是将主题 ( ocean aads\$aads-ocean ocean

i

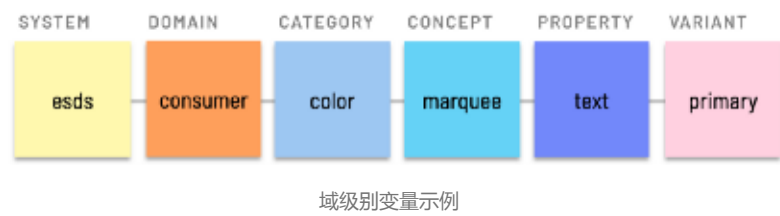
```
$aads-color-text-primary
= $aads-ocean-color-primary;
$aads-color-forms-text-metadata
= $aads-ocean-color-primary;
$aads-color-background-secondary-on-dark
= $aads-ocean-color-primary;
```

原则：主题不等于模式

一个主题可能最后会要求能应用在 on-light 和 on-dark

域 ( Domain )

虽然我还没有在实践中观察到这一点，但是我预料设计变量会拓展到可以支持设计系统层。一个域 ( 又名业务单元 ) 级别是命名空间下的一种类型，以便于为能按照系统的核心集合来创建、隔离和分发不同组的变量。



例如，一个包括选取框、卡片系统和其他促销组件的可能需要大量变量网络的 consumer 团队，它的命名空间可能是：

i

```
$esds-consumer-color-marquee-text-primary
$esds-consumer-color-promo-clearance
$esds-consumer-font-family-marquee
$esds-consumer-space-tiles-inset-2-x
```


这种域级别的变量命名方式会应用在多个主题 ( 不同的产品线 ) 和颜色模式 ( 客户的深/浅模式 ) 上，但是每个组织都是不同的，我们没有一个所谓的模版。一家银行按照 credit-cardbank 和 loan sales servicing consumer business publicpartner internal

本文小结

无论你按照什么级别来组织你的设计变量，命名的完整性、顺序和多层次结构都是你需要考虑好的。

完整性

没有一种命名方式可以包括所有的级别，一些级别 - domain、theme、element 它们比较少被使用到，另外一些 - 定量尺度与定性变体 它们往往是相互排斥的。避免教条地包括所有可能的级别或冗余地复制变量。相反，只需要保证级别的使用能充分的描述和对相似的变量做出区别即可。

```
 // Good
$esds-shape-tile-corner-radius
$esds-shape-tile-shadow-offset

// Bad, redundant
$esds-shape-tile-corner-radius-default-on-light
$esds-shape-tile-corner-radius-default-on-dark
$esds-shape-tile-shadow-offset-default-on-light
$esds-shape-tile-shadow-offset-default-on-dark
```

## 顺序


正如我在之前说到的，没有一个所谓的模版，但是我们有一些可以信赖的原则：

- **基础级别**（类别、属性、概念）是中间的支柱
- **基础级别**中的分类要保证**分层严格性**（color-interactive-background）和**可读性**（interactive-background-color），并且基于**类别**（category）和**属性**（property）的对应关系做好配对（color-background-interactive）
- **命名空间**（system, theme, domain）应该被优先定义
- **修饰级别**（variant、state、scale、mode）往往最后确定
- **对象级别**（组件组、组件和嵌套元素）从属于**命名空间**，并且作为**基础级别**和**修饰级别**的主体
- **修饰级别**中的顺序并不固定，尽管**模式**通常是最后的（鉴于其是框架的“开始”并且仅限于对颜色的约束），即使这样，也只有在有模式区别时才使用


此处显示的级别顺序是一个建议，但它不是唯一的选项。您的系统级别顺序取决于您使用的级别、系统需要什么以及每个团队成员的不同观念。

## 多级别结构的考虑

概念、类别、变体和其他层次可以重叠和互换。例如，“红色错误”既可以是**概念变体**（color-feedback-error，也可以是**对象变体**（ui-controls-color-text-error

 我将这个变量放在什么级别？  
可以将相同的变量放在两个不同的位置吗？  
如果两个不同的变量表现形式几乎相同，应该是 1 个还是 2 个？

对于 **error** 的概念集合，**color-feedback** 和 **ui-controls-color-text** 都有其他变体（分别是 `warningsuccess`、`info` 和 `label`、`value` 和 `helper-text`），即使实际的红色值相同，我们也更看重两组的完整性。因此，我会考虑将一个**对象变体**别名为另一个**概念变体**。

 \$sui-controls-color-text-error = \$color-feedback-error  
(= \$color-red-36)  
(= #B90000)

考虑这么多的可能性，变量的命名方式可能会让人望而生畏。但是当您读完本文，再去思考您的设计变量系统时，本文给您提供了一套新的观念，您可以为您的团队制定词汇表来共享或者共同打造一个合适的变量的命名方式。祝您顺利！