

Organização CSS

Danilo Vaz



Front-end na



Organizador da



Organizador do



Mestre Pokémon



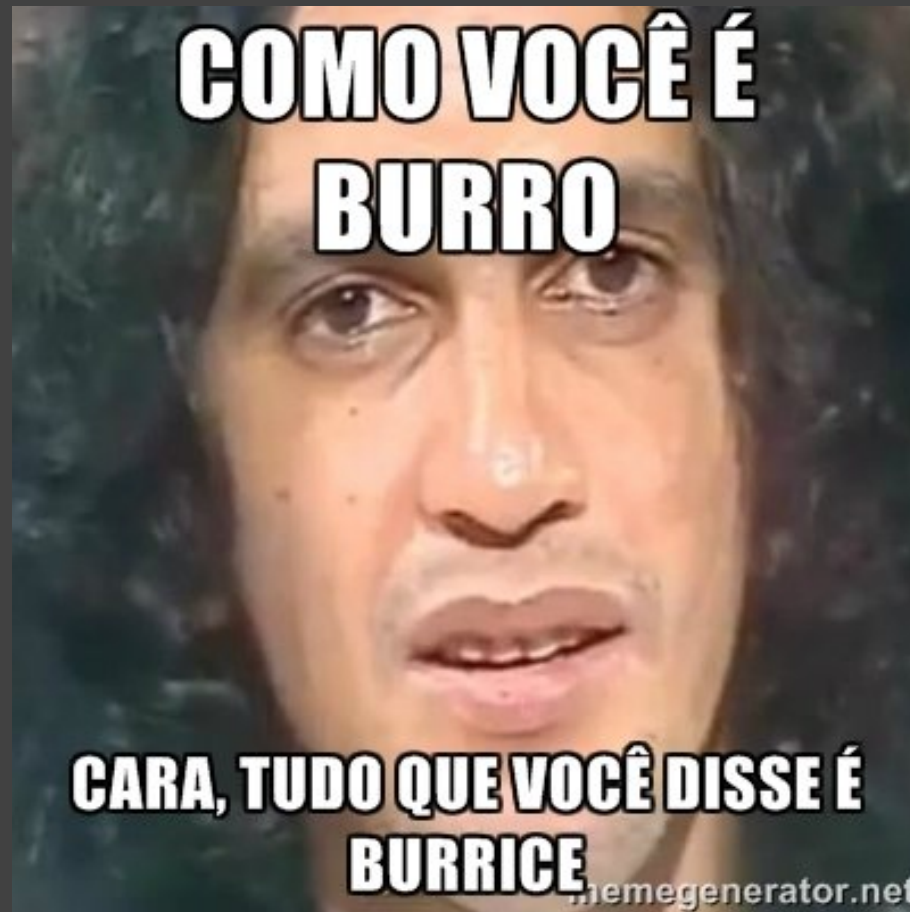
@_danilovaz



/danilovaz

Cascading **S**tyle **S**heet

- Organizado
- Manutenível
- Escalável



O seu CSS é organizado?



CSS é tão difícil de manter organizado



**Quanto mais você fuça,
mais riscos corre**



Qual a solução?

Não existe bala de prata

Mas podemos melhorar



RSCSS & ITCSS

RSCSS - O que é?

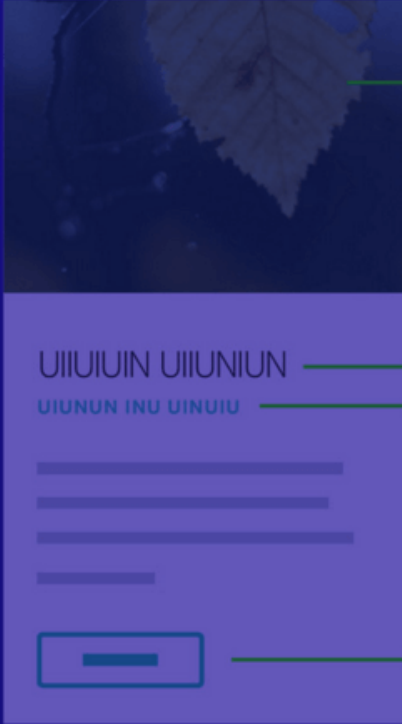
**Conjunto de regras para
ajudar na construção do
seu CSS**

Quem nunca?

```
.nav {  
  width: 100%;  
  padding: 10px 25px;  
  position: fixed;  
  
  ul {  
    max-width: 940px;  
    li {  
      display: inline-block;  
      border: 1px solid #ccc;  
      padding: 10px 50px;  
  
      a {  
        color: #333;  
  
        &:hover {  
          text-decoration: none;  
        }  
      }  
    }  
  }  
}
```

- Aumenta a especificidade
- Dependencia de tags

Nesting CSS nativo



The diagram shows a UI element, a photo card, with various components. Green lines connect these components to their corresponding CSS selectors in the code block on the right. The components are: a photo image, a heading (containing title and subtitle), a description paragraph, an action button, and a button element. The CSS code shows a deeply nested structure where each component is nested within the previous one, leading to long and complex selectors.


```
.photo-card {  
  .photo {  
    img { ... }  
  }  
  .text {  
    .heading {  
      .title { ... }  
      .subtitle { ... }  
    }  
    .description {  
      p { ... }  
    }  
    .action {  
      .btn { ... }  
    }  
  }  
}
```

Over-nesting is a common problem
with this approach, though, CSS becomes hard to read.

DEMO



BEM



BEM
Makes this a little easier.

| | |
|---------------------------------------|--|
| <code>.photo_card</code> | |
| <code>.photo_card--photo</code> | |
| <code>.photo_card--heading</code> | |
| <code>.photo_card--subheading</code> | |
| <code>.photo_card--description</code> | |
| <code>.photo_card--action</code> | |

Block

Element

DEMO



Problemas

Nesting nativo

- *Markup limpo*
- *CSS sujo*
- *Difícil de ler*

BEM

- *Markup sujo*
- *CSS limpo*
- *Fácil de ler*

RSCSS - Útil ao agradável



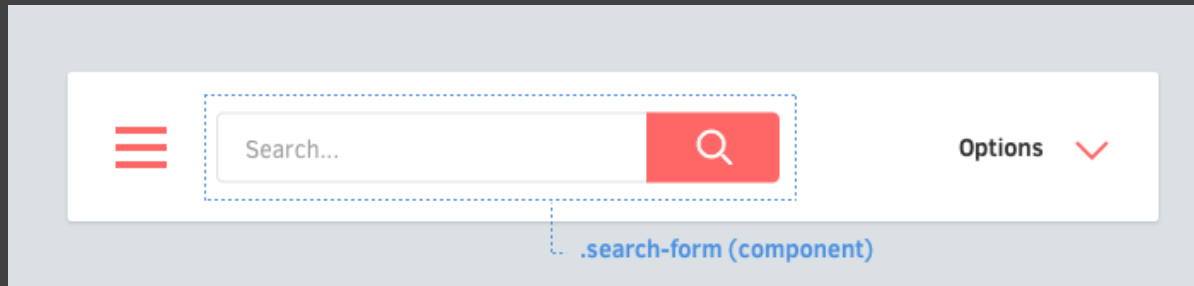
```
.photo-card {  
  > .image { ... }  
  > .heading { ... }  
  > .action { ... }  
  > .action > .btn { ... }  
}
```

RSCSS - DEMO



RSCSS - Regra de ouro

Tudo é componente



RSCSS - Nomeando Componentes

Os componentes devem ter **pelo menos duas palavras**, separadas por um traço.

Exemplos:

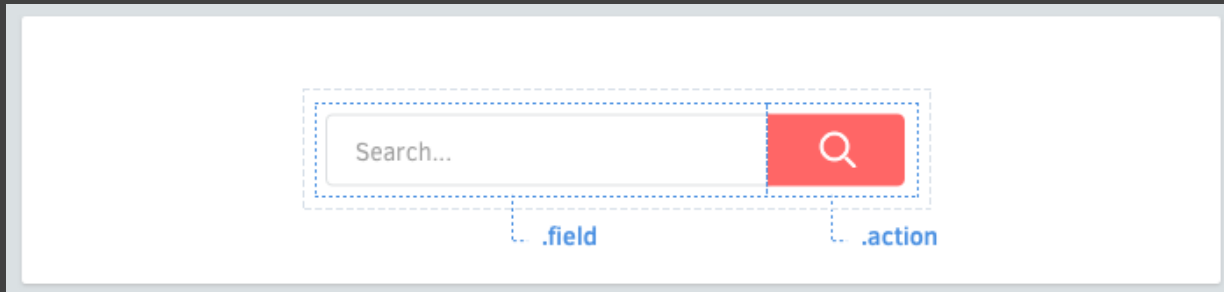
Um botão de share = .share-button

Um form de busca = .search-form

Um preview de post = .preview-post

RSCSS - Elementos

Elementos são os filhos do seu componente



RSCSS - Nomeando Elementos

Os elementos devem ter classes **com uma palavra**. Exemplos:

```
.search-form {  
  > .field {...}  
  > .action {...}  
}
```

RSCSS - Elementos Seletores

Use o seletor filho > sempre que possível.

- Previne que a propriedade passe para componentes internos indesejados
- Performa melhor que seletores descendentes

RSCSS - Elementos Multi-palavras

Elementos que precisam de uma ou mais palavras, concatene sem traços ou underscore

```
.profile-box {  
  > .firstname { /* ... */ }  
  > .lastname { /* ... */ }  
  > .avatar { /* ... */ }  
}
```

RSCSS - Tags

Evite usar tags como seletores, pois não são descritivas e há perda de performance

```
.article-card {  
  > h3      { /* x evite */ }  
  > .name { /* ✓ melhor */ }  
}
```

RSCSS - Variações

Componentes e elementos podem ter variações

The image displays three variations of a search form component, each with a label on the left and a visual representation on the right:

- .search-form**: A standard search form with a text input field containing the placeholder text "Search..." and a red square button with a white magnifying glass icon.
- .search-form .-prefixed**: A search form with a grey rectangular label "This project:" positioned to the left of the text input field, which contains the placeholder text "Search...". The red search button is to the right of the input field.
- .search-form .-compact**: A compact search form where the text input field and the red search button are combined into a single, wider rectangular element. The input field contains the placeholder text "Search..." and the button has a white magnifying glass icon.

RSCSS - Variações em componentes

Classes de variações devem ser prefixadas por um traço

```
.like-button {  
  &.-wide { /* ... */ }  
  &.-short { /* ... */ }  
  &.-disabled { /* ... */ }  
}
```

RSCSS - Variações em elementos

Também podemos utilizar traços prefixados e não há problema em colocá-lo adjacente

```
.shopping-card {  
  > .title { /* ... */ }  
  > .title.-small { /* ... */ }  
}
```



The logo features a large, inverted white triangle centered on a red background. Inside the triangle, there are three white circles: one at the top-left corner, one at the top-right corner, and one at the bottom vertex. A white horizontal trapezoidal bar is positioned across the middle of the triangle, containing the text "ITCSS" in a bold, red, sans-serif font.

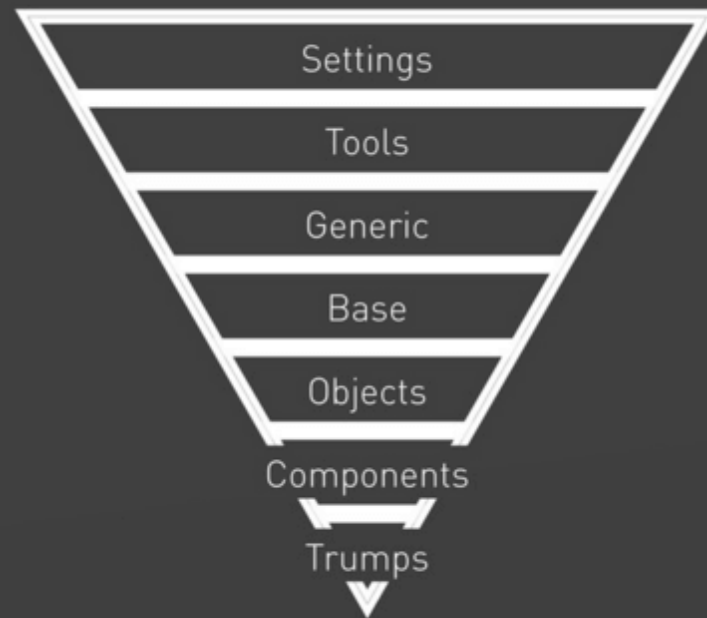
ITCSS

ITCSS - O que é?

**Conjunto de regras para
uma estrutura CSS sadia
e escalável**

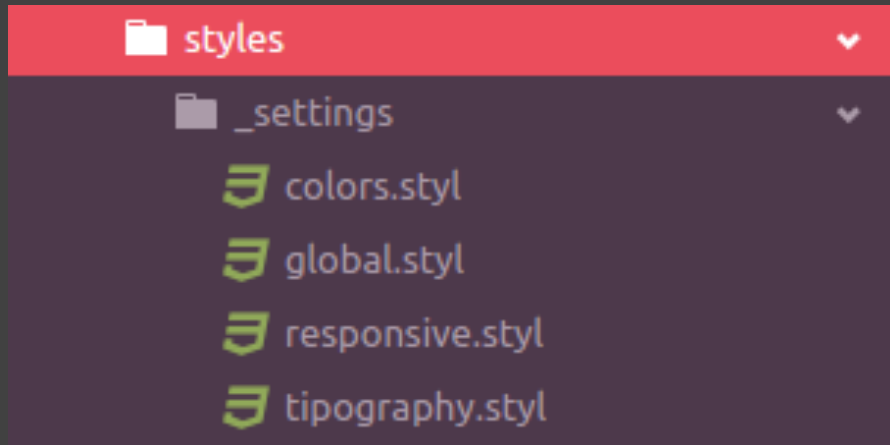
ITCSS - Como funciona?

Inverted Triangle CSS



ITCSS - Settings

Aqui ficam as variáveis globais que vão definir cores, espaçamentos e outras configurações desejadas para o funcionamento do seu projeto.



```
// Brand Channels
$color-facebook = #3a589b
$color-instagram = #956c54
$color-twitter = #28aae1
$color-linkedin = #0077b5
$color-youtube = #bd1b19
$color-google-plus = #d6492f
$color-pinterest = #b81d21
$color-analytics = #ffa500
```

ITCSS - Tools

Aqui ficam os mixins e funções necessárias para a construção do layout .

```
@mixin font-brand() {  
    font-family: "Prime", sans-serif;  
    font-weight: 400;  
}
```

ITCSS - Generic

Camada destinada às propriedades mais genéricas (dã).
Aqui ficam seletores com a menor especificidade possível,
como os famosos resets.

```
* {  
    box-sizing: border-box;  
}
```

ITCSS - Base

Estilizações básicas diretamente em tags. Geralmente utilizada para estilizar as tags h1-h6. Mas apenas **estilizações básicas**

```
ul {  
    list-style: square outside;  
}
```

ITCSS - Objects

Pequenos pedaços da interface **reutilizáveis** por todo o projeto, como botões por exemplo. Aqui só pode usar classes, nada de IDs e tags. Abstraia o máximo que puder

```
.base-list {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  
    > .item {  
        padding: $spacing-unit;  
    }  
}
```

ITCSS - Components

Aqui teremos que ser o mais específicos possível, mas sempre lembrando que um componente deve ser **reutilizável**. Então nada de positions e magins.

```
.products-list {  
  @extend font-brand();  
  border-top: 1px solid $color-ui;  
  
  > .item {  
    border-bottom: 1px solid $color-ui;  
  }  
}
```

ITCSS - Trumps



OPS!

ITCSS - Trumps



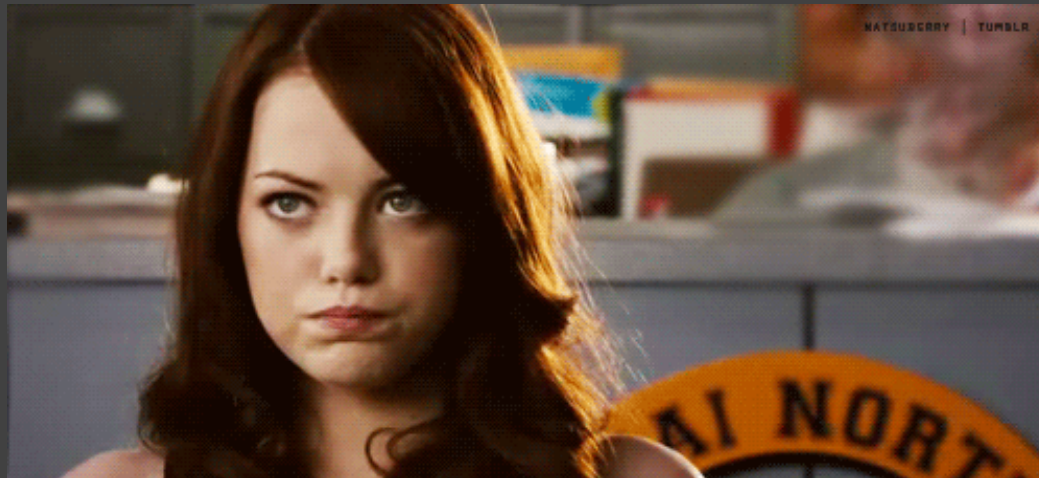
Camada responsável pela maior especificidade de todas, inclusive aqui é **plausível** o uso de **!important**

```
.hidden {  
    display: none !important;  
}
```

ITCSS - Recapitulando

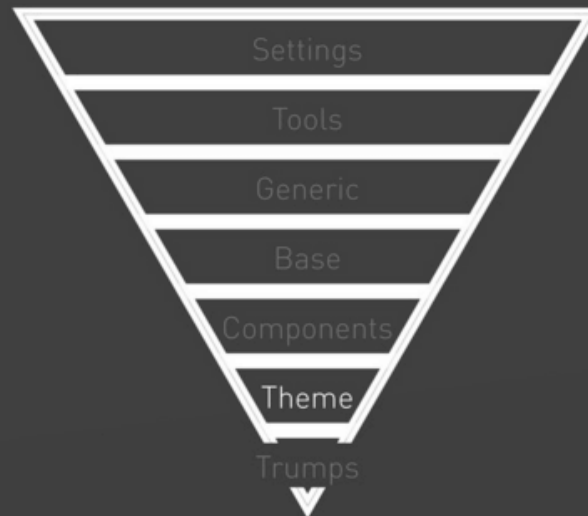
- A especificidade vai crescendo linearmente e lentamente de camada a camada;
- Cada camada é mais detalhada e explícita que a anterior;
- Se você cria um objeto genérico, ele pode ser estendido/reutilizado;
- Tudo que você cria tem um lugar específico para ficar.

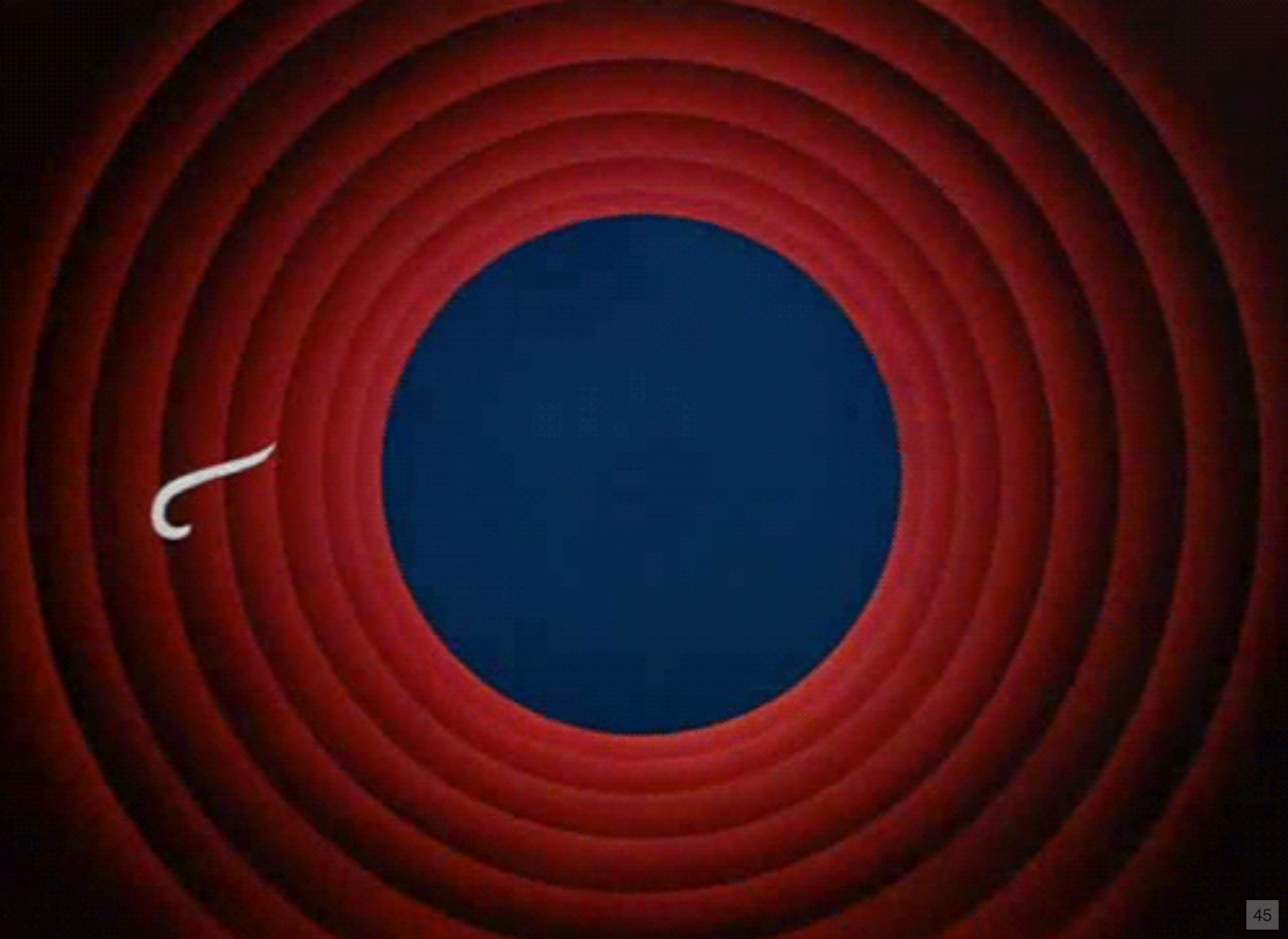
WAAAAAAAAAIIIT!
TEM MAIS



ITCSS - Temas

Se você utiliza **Testes A/B**, pode criar uma camada extra chamada **Theme**, entre Componentes e Trumps.





Perguntas?

