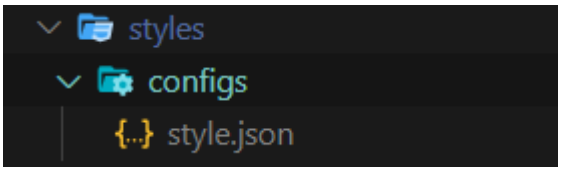
VTEX Styleguide

Olá!
Para iniciarmos o processo de estilização de componentes, o primeiro passo é entender qual é o pré-processador que articula nosso código.

A VTEX utiliza o framework **Tachyons**, um pré-processador que opera com pré-definições de estilos para que a customização geral use CSS o mínimo possível. Antes de colocar a mão no CSS, podemos definir as gramaturas de fontes, tamanhos, cores e padrões de espaçamentos.

Com base nas pré-definições no arquivo styles/configs/style.json, segundo o storetheme da VTEX, iniciaremos as nossas estilizações, definindo os padrões do projeto. Lá encontraremos definição de espaçamentos e diagramação:



Como padrão de unidade de medida, o Tachyons utiliza a propriedade REM.

we.digi

```
{
    "spacing": [
        0.125,
        0.25,
        0.5,
        0.75,
        1,
        1.5,
        2,
        3,
        4,
        8,
        16
    ]
}
```

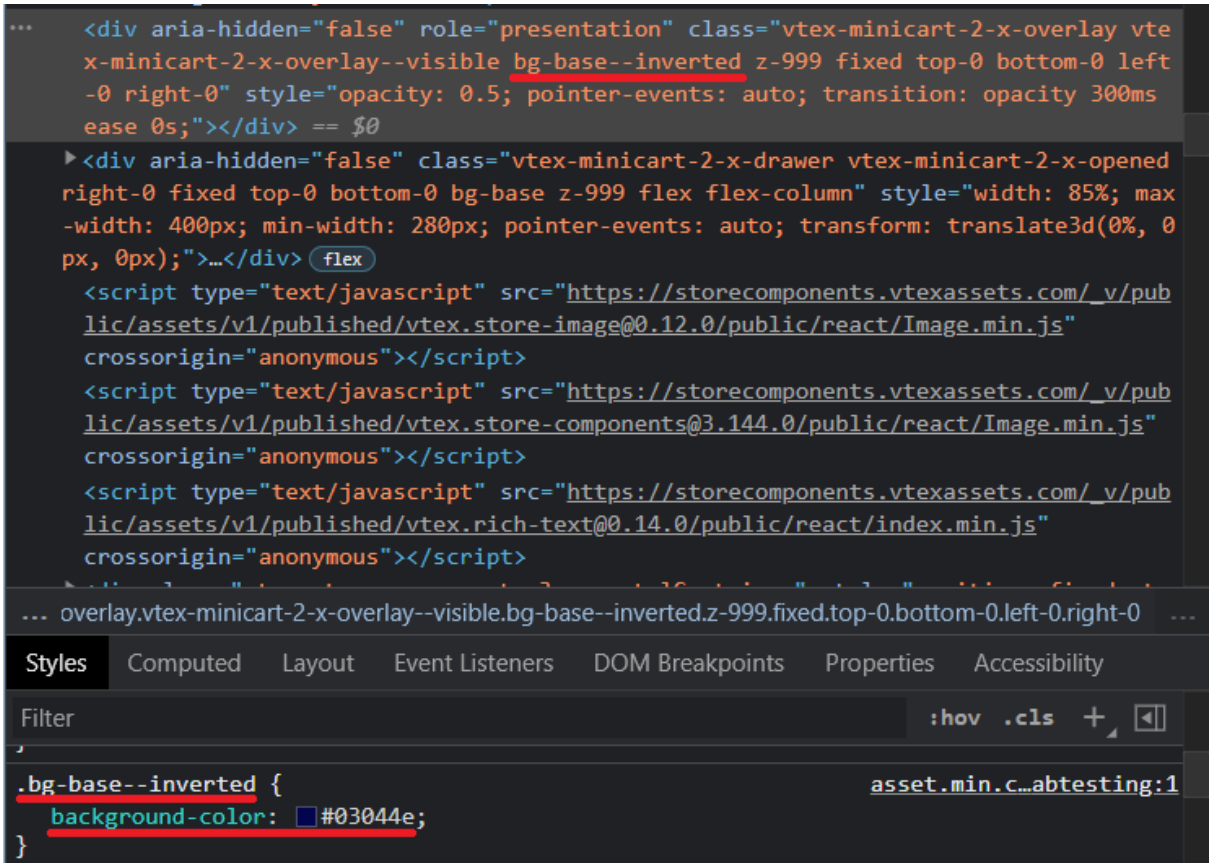
Definições de cores:

```
{
    "semanticColors": {
        "background": {
          "base": "#ffffff",
          "base--inverted": "#fff",
          "action-primary": "#000",
          "action-secondary": "#fff",
          "emphasis": "#000",
          "disabled": "#f2f4f5",
          "success": "#8bc34a",
          "success--faded": "#eafce3",
          "danger": "#ff4c4c",
          "danger--faded": "#ffe6e6",
          "warning": "#ffb100",
          "warning--faded": "#fff6e0",
          "muted-1": "#727273",
          "muted-2": "#979899",
          "muted-3": "#cacbcc",
          "muted-4": "#e3e4e6",
          "muted-5": "#f2f4f5"
        }
    }
}
```

Gramatura de títulos:

```
{
    "typography": {
        "styles": {
            "heading-1": {
                "fontFamily": "
Poppins, sans-serif",
                "fontWeight": "700",
                "fontSize": "3rem",
                "textTransform": "initial",
                "letterSpacing": "0"
            },
            "heading-2": {
                "fontFamily": "
Poppins, sans-serif",
                "fontWeight": "700",
                "fontSize": "2.25rem",
                "textTransform": "initial",
                "letterSpacing": "0"
            }
        }
    }
}
```

Para que esse documento não fique muito extenso, mas mesmo assim, seja possível compreender o funcionamento dessas definições no mundo real, veja o exemplo a seguir:



Os componentes podem utilizar essas classes pré definidas do Tachyons para estilizar como podemos ver aquilo que foi definido dentro de configs, ao declarar a cor dentro de background -> base--inverted.

É possível utilizar a classe **.bg-base--inverted** (Sendo “bg” abreviação de “background”, são padrões fornecidos pelo Tachyons), que a renderização da propriedade será efetivada background-color: #03044e.

we.digi

```
{
    "background": {
        "base": "#ffffff",
        "base--inverted": "#03044e",
        "action-primary": "#000",
        "action-secondary": "#fff",
        "emphasis": "#000",
        "disabled": "#f2f4f5",
        "success": "#8bc34a",
        "success--faded": "#eafce3",
        "danger": "#ff4c4c",
        "danger--faded": "#ffe6e6",
        "warning": "#ffb100",
        "warning--faded": "#fff6e0",
        "muted-1": "#727273",
        "muted-2": "#979899",
        "muted-3": "#cacbcc",
        "muted-4": "#e3e4e6",
        "muted-5": "#f2f4f5"
    }
}
```

Essas propriedades - que são o Styleguide da VTEX - são ótimas para criar componentes em React e não depender somente da memória para lembrar dos hexadecimais, dos tamanhos, entre outros componentes que serão usados naquele contexto. Assim, é possível importar a classe correta e obter as cores e tamanhos já padronizados no nosso código.

Porém, é importante entender o funcionamento do Styleguide, pois a vantagem de contar com o Framework VTEX IO, está na gama de componentes prontos para todos os propósitos possíveis dentro de um site e-commerce.

Com as fontes configuradas, gramaturas, cores e espaçamentos, temos algo muito mais próximo da identidade de uma marca. Porém, algumas situações exigem soluções mais customizadas, e aí que entra a “**BlockClass**”.

Para o estudo dessa propriedade, utilizaremos o componente “**rich-text**”, que consideramos ser o mais básico e simples de entender.

Ao declararmos um bloco de texto dentro de store, é possível ver o retorno diretamente no navegador.

```
{
    "rich-text#hello-world": {
        "props": {
            "text": "# Hello World!"
        }
    }
}
```

Retorno no navegador:

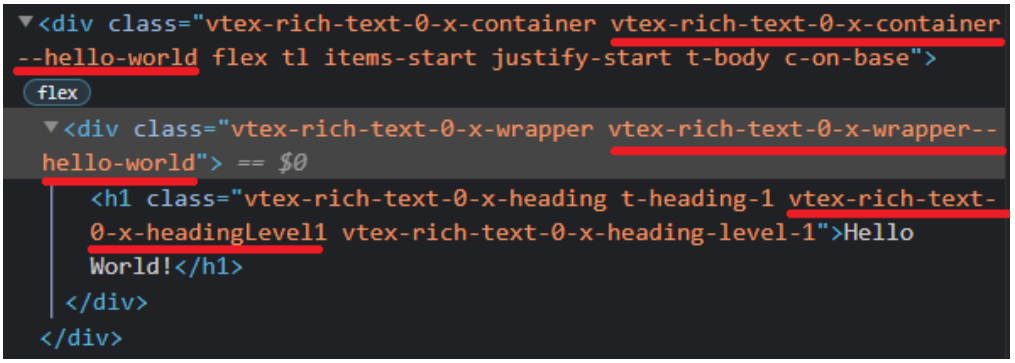
Hello World!

Note que o componente está sem estilo, apenas apresentando o que foi pré-definido dentro de configs. Para melhorar, basta recorrer à propriedade **BlockClass**. Nela iremos adicionar uma classe única para que possamos customizá-la dentro de um documento css.

Adicionada a propriedade BlockClass:

```
{
    "rich-text#hello-world": {
        "props": {
            "text": "# Hello World!",
            "blockClass": "hello-world"
        }
    }
}
```

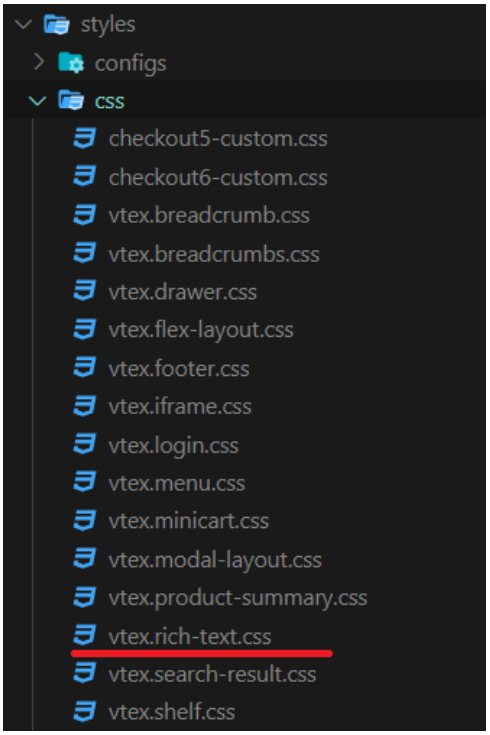
No navegador:



É possível perceber que todo componente irá renderizar o componente com o próprio nome antes da classe. **É assim que o Styleguide funciona:** antes de renderizar uma classe, ele irá referenciar o componente em questão e irá nos disponibilizar uma classe a ser customizada via CSS.

Essa é uma maneira de termos um código mais organizado e funcional, colocando os estilos de um componente específico em seu respectivo documento.

Ainda na pasta styles, clicaremos no documento .css do componente que está sendo usado para o exemplo em questão:

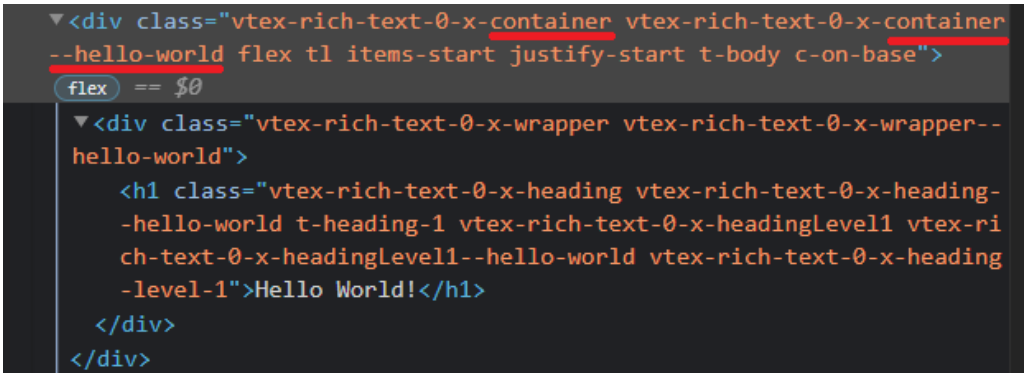


Dentro desse documento, você encontrará duas opções:

1 - referenciar todos os componentes que sejam um rich-text
2 - ou todos que pertencem a uma mesma classe.

Ao utilizar a classe que criamos, é possível reutilizá-la quantas vezes necessário, podendo também criar styles genéricos para tudo que for rich-text.

Para exemplificar, veja renderização do componente no inspetor do Browse:



Dessa forma, é possível usar a classe genérica ou a específica declarada dentro de BlockClass.

Em uma situação hipotética, onde o objetivo é que tudo que for rich-text seja da cor vermelha e, especificamente, a que possui a classe hello-world seja em itálico:

Hello World!

Segundo texto

Eis o código:

```
.container .heading {
  color: red;
}
.container--hello-world .heading {
  font-style: italic;
}
```