

# Chakra-UI

现代化 React UI 框架 Chakra-UI

# 1. Chakra-UI 介绍

Chakra UI 是一个简单的, 模块化的易于理解的 UI 组件库. 提供了丰富的构建 React 应用所需的UI组件.

文档: <https://next.chakra-ui.com/docs/getting-started>

1. Chakra UI 内置 Emotion, 是 CSS-IN-JS 解决方案的集大成者
2. 基于 Styled-Systems <https://styled-system.com/>
3. 支持开箱即用的主题功能
4. 默认支持白天和黑夜两种模式
5. 拥有大量功能丰富且非常有用的组件
6. 使响应式设计变得轻而易举
7. 文档清晰而全面. 查找API更加容易
8. 适用于构建用于展示的给用户的界面
9. 框架正在变得越来越完善

# 2. Chakra-UI 快速开始

## 2.1 下载 chakra-ui

```
npm install @chakra-ui/core@1.0.0-next.2
```

# 2. Chakra-UI 快速开始

## 2.2 克隆默认主题

Chakra-UI 提供的组件是建立在主题基础之上的, 只有先引入了主题组件才能够使用其他组件.

```
npm install @chakra-ui/theme
```

## 2. Chakra-UI 快速开始

### 2.3. 引入主题

```
import { ChakraProvider } from '@chakra-ui/core';  
import theme from "@chakra-ui/theme";  
  
<ChakraProvider theme={theme}>  
  <App />  
</ChakraProvider>
```

## 2. Chakra-UI 快速开始

### 2.4 引入 CSS 重置组件

```
import { CSSReset } from '@chakra-ui/core';  
  
<ChakraProvider theme={theme}>  
  <CSSReset />  
  <App />  
</ChakraProvider>
```

# 3. Style Props 样式属性

Style Props 是用来更改组件样式的, 通过为组件传递属性的方式实现. 通过传递简化的样式属性以达到提升开发效率的目的.

样式属性	css属性	主题
m, margin	margin	space
mx	margin-left & margin-right	space
p, padding	padding	space
py	padding-top & padding-botton	space
bg	background	colors
bgColor	background-color	colors
color	color	colors
border	border	borders
textAlign	text-align	none
w, width	width	sizes
boxSize	width & height	sizes
d, display	display	none
pos, position	position	none
left	left	space
shadow, boxShadow	box-shadow	shadows

# 3. Style Props 样式属性

Style Props 是用来更改组件样式的, 通过为组件传递属性的方式实现. 通过传递简化的样式属性以达到提升开发效率的目的.

```
import { Box } from '@chakra-ui/core';  
  
<Box w={200} h={200} bg="tomato" p="3px"> Hello chakra-ui </Box>
```



# 4. 主题

## 4.1 颜色模式(color mode)

chakra-ui 提供的组件都支持两种颜色模式, 浅色模式(light)和暗色模式(dark).

可以通过 useColorMode 进行颜色模式的更改.

```
import { useColorMode } from '@chakra-ui/core';

const [colorMode, toggleColorMode] = useColorMode();

<Text>当前的颜色模式为 {colorMode}</Text>
<Button onClick={toggleColorMode}>切换颜色模式</Button>
```

Chakra 将颜色模式存储在 localStorage 中, 并使用类名策略来确保颜色模式是持久的.

# 4. 主题

## 4.2 根据颜色模式设置样式

chakra 允许在为元素设置样式时根据颜色模式产生不同值. 通过 useColorModeValue 钩子函数实现.

```
const bgColor = useColorModeValue(lightModeValue, darkModeValue);  
  
<Box bgColor={bgColor}></Box>
```

# 4. 主题

## 4.3 强制组件颜色模式

使组件不受颜色模式的影响, 始终保持在某个颜色模式下的样式.

```
import { LightMode, DarkMode } from '@chakra-ui/core';  
  
<LightMode>  
  <Button onClick={toggleColorMode}>按钮</Button>  
</LightMode>
```

# 4. 主题

## 4.4 颜色模式通用设置

### 1. 设置默认颜色模式

通过 `theme.config.initialColorMode` 可以设置应用使用的默认主题.

### 2. 使用操作系统所使用的颜色模式










通过 `theme.config.useSystemColorMode` 可以设置将应用的颜色模式设置为操作系统所使用的颜色模式.

# 4. 主题

## 4.5 主题对象

### 1. Colors

在设置颜色时, 可以但不限于取主题中提供的颜色值.

Red		
	Red 50 #FFF5F5	
	Red 100 #FED7D7	
	Red 300 #FC8181	
	Red 600 #C53030	
		
	Red 900 #63171B	

# 4. 主题

## 4.5 主题对象

### 1. Colors

在设置颜色时, 可以但不限于取主题中提供的颜色值.

```
<Box color="gray.500"></Box>
```

# 4. 主题

## 4.5 主题对象

### 2. Space

使用 space 可以自定义项目间距. 这些间距值可以由 padding, margin 和 top, left, right, bottom 样式引用.

```
<Box mb="5"></Box>  $\leftarrow 5 \Rightarrow 1.25rem \rightarrow$ 
```

# 4. 主题

## 4.5 主题对象

### 3. Sizes

使用 size 可以自定义元素大小, 这些值可以由 width, height 和 maxWidth, minWidth 等样式引用.

```
<Box w="lg"></Box> ←!—— lg ⇒ 32rem —→
```



# 4. 主题

## 4.5 主题对象

### 4. Breakpoints

配置响应数组值中使用的默认断点. 这些值将用于生成移动优先(即最小宽度)的媒体查询.

```
// theme.js
export default {
  breakpoints: ["30em", "48em", "62em", "80em"],
};
```

[COPY](#)

```
<Box fontSize={['12px', '14px', '16px', '18px', '20px']}></Box>
```

# 4. 主题

## 4.6 创建标准的 Chakra-UI 组件

### 1. 创建 Chakra-UI 组件

```
const LaGouButton = chakra("button", {  
  baseStyle: {},  
  sizes: {},  
  variants: {}  
});
```

```
LaGouButton.defaultProps = {};
```

```
<LaGouButton>按钮</LaGouButton>
```

# 4. 主题

## 4.6 创建标准的 Chakra-UI 组件

### 2. 全局化 Chakra-UI 组件样式

- a. 在 src 文件夹中创建 lagou 文件夹用于放置自定义 Chakra-UI 组件
- b. 在 lagou 文件夹中创建 button.js 文件并将组件样式放置于当前文件中并进行默认导出

```
const LaGouButton = {  
  baseStyle: {},  
  sizes: {},  
  variants: {},  
  defaultProps: {}  
}  
export default LaGouButton;
```

# 4. 主题

## 4.6 创建标准的 Chakra-UI 组件

### 2. 全局化 Chakra-UI 组件样式

c. 在 lagou 文件夹中创建 index.js 文件用于导入导出所有的自定义组件

```
import LaGouButton from './button';  
  
export default {  
  LaGouButton  
}
```

# 4. 主题

## 4.6 创建标准的 Chakra-UI 组件

### 2. 全局化 Chakra-UI 组件样式

d. 在 src 文件夹中的 index.js 文件中导入自定义 Chakra-UI 组件并和 components 属性进行合并

```
import LaGouComponents from './Lagou';
const myTheme = {
  ...theme,
  components: {
    ...theme.components,
    ...LaGouComponents
  }
}
```

# 4. 主题

## 4.6 创建标准的 Chakra-UI 组件

### 2. 全局化 Chakra-UI 组件样式

#### e. 在组件中使用样式化组件

```
const LaGouButton = chakra('button', {  
  themeKey: 'LaGouButton'  
});
```