CTA SDK 技术文档

目录

- 简介
- 安装
- 快速开始
- API 参考
- 使用示例
- 常见问题

简介

CTA (Call To Action) SDK 是一个用于在网页中嵌入引导组件的 JavaScript SDK。

安装

1. 在 HTML 文件中引入 SDK：

```
//ctaSDK使用依赖官网sdk，请先完成官网sdk引入
<script src="https://www.fxiaoke.com/ec/h5-landing/release/ctaSDK-1.0.0.js"></script>
```

2. 确保全局变量 `FsYxt` 可用，SDK 会挂载在该对象下。

快速开始

基本使用示例：

```
const cta = FsYxt.CtaSDK({
    ctaId: "your-cta-id"
});

// 监听准备就绪事件
cta.onReady(() => {
    console.log('CTA 组件已准备就绪');
});

// 监听错误事件
cta.onError((error) => {
    console.log('发生错误: ', error);
});
```

API 参考

构造函数选项

`FsYxt.CtaSDK(options)`

参数	类型	必填	默认值	说明
ctaId	string	是	-	CTA 组件的唯一标识
hideEntryBtn	boolean	否	false	是否隐藏默认引导按钮
autoExecuteOnReady	boolean	否	true	是否在准备就绪后自动执行
marketingEventId	string	否	-	市场活动 ID
buttonId	string	否	-	自定义触发按钮的 ID
actionStyle	object	否	-	样式配置对象

样式配置选项

`actionStyle` 对象包含以下配置项：

参数	类型	必填	默认值	说明
position	string	否	'center'	组件弹出位置， 可选值：'center'（居中）、'bottom-

参数	类型	必填	默认值	说明
				right'（右下角）
maskOpacity	number	否	0.6	遮罩层透明度，取值范围 0-1
width	string	否	'280px'	表单弹窗宽度，支持 px、%、vw 等单位
borderRadius	string	否	'6px'	弹层圆角，支持 px、% 等单位

实例方法

onReady(callback)

监听组件准备就绪事件

```
cta.onReady(() => {
    console.log('组件已就绪');
});
```

onError(callback)

监听错误事件

```
cta.onError((error) => {
    console.log('发生错误：', error);
});
```

start()

手动触发 CTA 引导组件

```
cta.start();
```

onActionComplete(callback)

监听引导完成事件

```
cta.onActionComplete(() => {
    console.log('引导完成');
});
```

onActionStep(callback)

监听引导步骤事件

```
cta.onActionStep((step) => {
    console.log('当前步骤：', step);
});
```

destroy()

销毁 CTA 实例

```
cta.destroy();
```

使用示例

1. 基础用法

```
const cta = FsYxt.CtaSDK({
    ctaId: "your-cta-id"
});
```

2. 自定义按钮触发和样式

```
const cta = FsYxt.CtaSDK({
    ctaId: "your-cta-id",
    autoExecuteOnReady: false,
    hideEntryBtn: true,
    actionStyle: {
        position: 'bottom-right',
        maskOpacity: 0.8,
        width: '500px',
        borderRadius: '16px'
    }
});

cta.onReady(() => {
    const button = document.getElementById('custom-button');
    button.addEventListener('click', () => {
        cta.start();
    });
});
```

3. 完整的生命周期管理

```
const cta = FsYxt.CtaSDK({
    ctaId: "your-cta-id",
    marketingEventId: "your-marketing-event-id"
});

cta.onReady(() => {
    console.log('CTA 已就绪');
});

cta.onActionStep((step) => {
    console.log('当前步骤: ', step);
});

cta.onActionComplete(() => {
    console.log('引导完成');
    // 执行后续操作
});

cta.onError((error) => {
    console.log('发生错误: ', error);
});

// 不再需要时销毁实例
// cta.destroy();
```

4. 多实例使用

```
// 第一个实例
const cta1 = FsYxt.CtaSDK({
    ctaId: "first-cta-id"
});

// 第二个实例
const cta2 = FsYxt.CtaSDK({
    ctaId: "second-cta-id",
    autoExecuteOnReady: false
});
```

常见问题

Q1: 如何在页面加载完成后初始化多个 CTA?

使用 `window.addEventListener('load', ...)` 而不是 `window.onload`：

```
window.addEventListener('load', () => {
    FsYxt.CtaSDK({
        ctaId: "first-cta-id"
    });
});

window.addEventListener('load', () => {
    FsYxt.CtaSDK({
        ctaId: "second-cta-id"
    });
});
```

Q2: 如何确保 CTA 组件在特定条件下才执行?

使用 `autoExecuteOnReady: false` 并手动控制启动时机：

```
const cta = FsYxt.CtaSDK({
    ctaId: "your-cta-id",
    autoExecuteOnReady: false
});

cta.onReady(() => {
    // 检查条件
    if (yourCondition) {
        cta.start();
    }
});
```

Q3: 如何处理错误情况?

始终添加错误处理回调：

```
cta.onError((error) => {
    console.error('CTA 错误: ', error);
    // 实现错误恢复逻辑
});
```

Q4: 如何在单页面应用（SPA）中使用?

在路由变化时记得销毁不需要的实例：

```
// 组件卸载时
componentWillUnmount() {
    if (this.cta) {
        this.cta.destroy();
    }
}
```