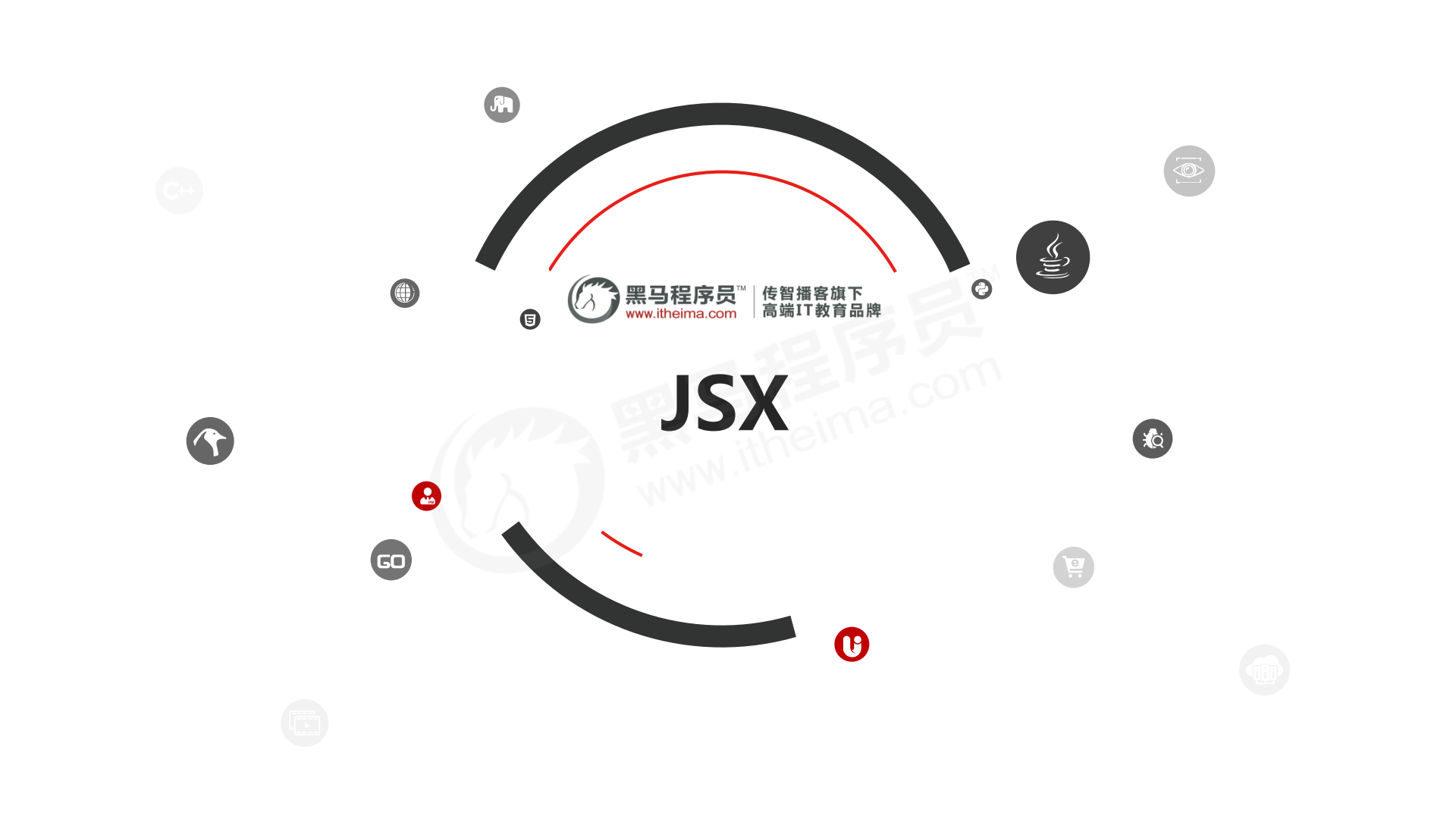

黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
JSX

目录 Contents

1. JSX 的基本使用

1.1 createElement() 的问题

1. 繁琐不简洁。
2. 不直观，无法一眼看出所描述的结构。
3. 不优雅，用户体验不爽。

```
React.createElement(
  'div',
  {className: 'shopping-list'},
  React.createElement('h1', null, 'Shopping List'),
  React.createElement(
    'ul',
    null,
    React.createElement('li', null, 'Instagram'),
    React.createElement('li', null, 'WhatsApp')
  )
)
```

JSX

```
<div className="shopping-list">
  <h1>Shopping List</h1>
  <ul>
    <li>Instagram</li>
    <li>WhatsApp</li>
  </ul>
</div>
```

1. JSX 的基本使用

1.2 JSX 简介

JSX 是 JavaScript XML 的简写，表示在 JavaScript 代码中写 XML（HTML）格式的代码。

优势：声明式语法更加直观、与HTML结构相同，降低了学习成本、提升开发效率

JSX 是 React 的核心内容。

JSX

```
<div className="shopping-list">
  <h1>Shopping List</h1>
  <ul>
    <li>Instagram</li>
    <li>WhatsApp</li>
  </ul>
</div>
```

```
React.createElement(
  'div',
  {className: 'shopping-list'},
  React.createElement('h1', null, 'Shopping List'),
  React.createElement(
    'ul',
    null,
    React.createElement('li', null, 'Instagram'),
    React.createElement('li', null, 'WhatsApp')
  )
)
```

1. JSX 的基本使用

1.3 使用步骤

1. 使用 JSX 语法创建 react 元素

```
// 使用 JSX 语法，创建 react 元素：
const title = <h1>Hello JSX</h1>
```

2. 使用 ReactDOM.render() 方法渲染 react 元素到页面中

```
// 渲染创建好的React元素
ReactDOM.render(title, root)
```

1. JSX 的基本使用

小结

1. 推荐使用JSX语法创建React元素
2. 写JSX就跟写HTML一样，更加直观、友好
3. JSX语法更能体现React的声明式特点（描述UI长什么样子）
4. 使用步骤：

```
// 1 使用JSX创建react元素
const title = <h1>Hello JSX</h1>
// 2 渲染创react元素
ReactDOM.render(title, root)
```

1. JSX 的基本使用

为什么脚手架中可以使用 JSX 语法？

1. JSX 不是标准的 ECMAScript 语法，它是 ECMAScript 的语法扩展。
2. 需要使用 babel 编译处理后，才能在浏览器环境中使用。
3. create-react-app 脚手架中已经默认有该配置，无需手动配置。
4. 编译 JSX 语法的包为：@babel/preset-react 。

1. JSX 的基本使用

1.4 注意点

1. React元素的属性名使用驼峰命名法
2. 特殊属性名：class -> className、for -> htmlFor、tabindex -> tabIndex 。
3. 没有子节点的React元素可以用 /> 结束 。
4. 推荐：使用小括号包裹 JSX ，从而避免 JS 中的自动插入分号陷阱。

```
// 使用小括号包裹 JSX
const dv = (
  <div>Helo JSX</div>
)
```

目录 Contents

2. JSX 中使用 JavaScript 表达式

嵌入 JS 表达式

- 数据存储在JS中
- 语法：**{ JavaScript表达式 }**
- 注意：语法中是**单大括号**，不是双大括号！

```
const name = 'Jack'
const dv = (
  <div>你好，我叫：？？？</div>
)
```

```
const name = 'Jack'
const dv = (
  <div>你好，我叫：{name}</div>
)
```

2. JSX 中使用 JavaScript 表达式

注意点

- **单大括号**中可以使用任意的 JavaScript 表达式
- JSX 自身也是 JS 表达式
- 注意：JS 中的对象是一个例外，一般只会出现在 style 属性中
- 注意：不能在{}中出现语句（比如：if/for 等）

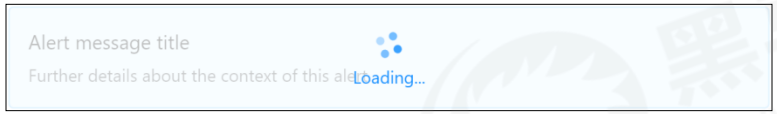
```
const h1 = <h1>我是JSX</h1>

const dv = (
  <div>嵌入表达式：{h1}</div>
)
```

目录 Contents

3. JSX 的条件渲染

- 场景：loading效果
- 条件渲染：根据条件渲染特定的 JSX 结构
- 可以使用if/else或三元运算符或逻辑与运算符来实现




```
const loadData = () => {
  if (isLoading) {
    return <div>数据加载中，请稍后...</div>
  }
  return (
    <div>数据加载完成，此处显示加载后的数据</div>
  )
}
const dv = (
  <div>
    {loadData()}
  </div>
)
```

目录 Contents

4. JSX 的列表渲染

- 如果要渲染一组数据，应该使用数组的 **map()** 方法

```
const songs = [
  {id: 1, name: '痴心绝对'},
  {id: 2, name: '像我这样的人'},
  {id: 3, name: '南山南'},
]
```

```
const list = (
  <ul>
    {songs.map(item => <li>{item.name}</li>)}
  </ul>
)
```

4. JSX 的列表渲染

- 如果要渲染一组数据，应该使用数组的 **map()** 方法
- 注意：渲染列表时应该添加 key 属性，key 属性的值要保证唯一
- 原则：map() 遍历谁，就给谁添加 key 属性
- 注意：尽量避免使用索引号作为 key

```
const songs = [
  {id: 1, name: '痴心绝对'},
  {id: 2, name: '像我这样的人'},
  {id: 3, name: '南山南'},
]
```

```
const list = (
  <ul>
    {songs.map(item => <li key={item.id}>{item.name}</li>)}
  </ul>
)
```

目录 Contents

5. JSX 的样式处理

1. 行内样式 —— style

```
<h1 style={{ color: 'red', backgroundColor: 'skyblue' }}>
  JSX的样式处理
</h1>
```

2. 类名 —— className（推荐）

```
<h1 className="title">
 JSX的样式处理
</h1>
```

JSX

1. JSX 是React 的核心内容。
2. JSX 表示在JS代码中写HTML结构，是React声明式的体现。
3. 使用 JSX 配合嵌入的 JS 表达式、条件渲染、列表渲染，可以描述任意 UI 结构。
4. 推荐使用 className 的方式给JSX添加样式。
5. React 完全利用 JS 语言自身的能力来编写UI，而不是造轮子增强 HTML 功能。

传智播客旗下高端IT教育品牌