**Function Comonents 与 Classes Components？**

**function components的优点有：**

**1、function component更易于编写阅读和测试**

**2、因为没有状态，可以更好的实现容器和表现的分离，可以只负责表现层的逻辑，不用考虑因为复杂的逻辑去改变状态从而带来的麻烦，有利于代码复用**

**然后是class components的优点：**

**1、当需要实现一些容器组件的时候，需要改变内部状态来实现自组件的改变的时候。**

**2、当需要用到生命周期钩子函数实现一些功能的时候**

**3、当我们需要提升性能时，性能是一个很重要的问题，有些时候我们需要减少组件的渲染次数，我们就需要在组件内部用shouldComponentUpdate 方法来去判断，或者继承React.PureComponent 类（自动调用shouldComponentUpdate）来实现state和props的浅比较进行判断组件是否重新渲染。**

**当然这是以前，Function Components也叫做无状态组件，只是一个普通的函数。而class组件提供了更多特性(如State、生命周期); 因此即使function component有很多好处，但是有些时候class component也是不可替代的。**

**然而react16.8新增了Hook特性，Function Components不再单单是一个普通的函数, Hook使我们在非class组件的情况下同样可以使用更多的React特性**

**当然，虽然增强了函数组件的功能，然而class并不会从react中被移除。react官方也推荐我们在新的代码中同时使用Hooks和class。**

**Hook 是一些可以让你在函数组件里“钩入” React state 及生命周期等特性的函数。**

**如果你在编写函数组件并意识到需要向其添加一些 state，以前的做法是必须将其它转化为 class。现在你可以在现有的函数组件中使用 Hook。**

**import React, { useState } from 'react';**

**function Example() {**

**// 声明一个叫 "count" 的 state 变量**

**const [count, setCount] = useState(0);**

**return (**

**<div>**

**<p>You clicked {count} times</p>**

**<button onClick={() => setCount(count + 1)}>**

**Click me**

**</button>**

**</div>**

**);**

**}**

**等价于**

**class Example extends React.Component {**

**constructor(props) {**

**super(props);**

**this.state = {**

**count: 0**

**};**

**}**

**render() {**

**return (**

**<div>**

**<p>You clicked {this.state.count} times</p>**

**<button onClick={()=>this.setState({count: this.state.count + 1 })}>**

**Click me**

**</button>**

**</div>**

**);**

**}**

**}**

**useEffect, 通过使用这个 Hook，你可以告诉 React 组件需要在渲染后执行某些操作; 不用像在class中可能需要在componentDidMount和componentDidUpdate中同时使用；与 componentDidMount或componentDidUpdate不同,使用useEffect调度的 effect 不会阻塞浏览器更新屏幕，这让你的应用看起来响应更快。**

**effect中返回一个函数， 则是可选的清除机制。**

**参考: react官方文档**<https://zh-hans.reactjs.org/docs>

<https://overreacted.io/how-are-function-components-different-from-classes/>

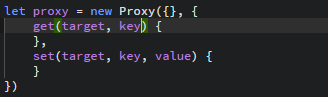
ES6 Proxy

**Proxy 可以理解成，在目标对象之前架设一层“拦截”，外界对该对象的访问，都必须先通过这层拦截，这样提供了方法可以对外界的访问进行过滤和改写。Proxy 这个词的原意是代理，用在这里表示由它来“代理”某些操作，可以译为“代理器”。**

**ES6原生提供 Proxy 构造函数，用来生成 Proxy 实例**

****

**Proxy 对象的所有用法，都是上面这种形式，不同的只是handler参数的写法。其中，target参数表示所要拦截的目标对象，handler参数也是一个对象，用来定制拦截行为，若handler是空对象{}，访问proxy等同于直接访问target对象。**

****

**Proxy 支持的拦截操作一览，一共 13 种:**

**1、get(target, propKey, receiver)**

**拦截对象属性的读取，比如proxy.foo和proxy['foo']**

**2、set(target, propKey, value, receiver)**

**拦截对象属性的设置，比如proxy.foo = v或proxy['foo'] = v**

**3、has(target, propKey)**

**拦截propKey in proxy的操作，返回一个布尔值**

**4、deleteProperty(target, propKey)**

**拦截delete proxy[propKey]/ Reflect.deleteProperty(obj, 's')的操作，返回一个布尔值**

**5、ownKeys(target)**

**拦截Object.getOwnPropertyNames(proxy)、Object.keys(proxy)、for in循环，返回一个数组。该方法返回目标对象所有自身的属性的属性名，而Object.keys()的返回结果仅包括目标对象自身的可遍历属性**

**6、getOwnPropertyDescriptor(target, propKey)**

**拦截Object.getOwnPropertyDescriptor(proxy, propKey)，返回属性的描述对象**

**7、defineProperty(target, propKey, propDesc)**

**拦截Object.defineProperty(proxy, propKey, propDesc）、Object.defineProperties(proxy, propDescs)，返回一个布尔值**

**8、preventExtensions(target)**

**拦截Object.preventExtensions(proxy)，返回一个布尔值**

**9、getPrototypeOf(target)**

**拦截Object.getPrototypeOf(proxy)，返回一个对象**

**10、isExtensible(target)**

**拦截Object.isExtensible(proxy)，返回一个布尔值**

**11、setPrototypeOf(target, proto)**

**拦截Object.setPrototypeOf(proxy, proto)，返回一个布尔值。如果目标对象是函数，那么还有两种额外操作可以拦截**

**12、apply(target, object, args)**

**拦截 Proxy 实例作为函数调用的操作，比如proxy(...args)、proxy.call(object, ...args)、proxy.apply(...)**

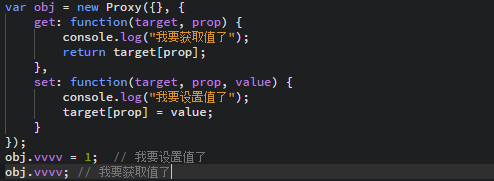
**13、construct(target, args)**

**拦截 Proxy 实例作为构造函数调用的操作，比如new proxy(...args)**

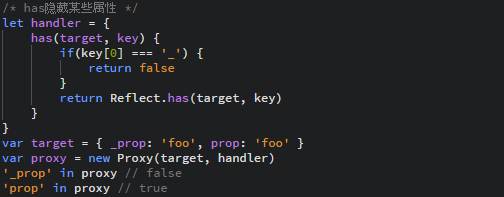
**需要强调的是，Reflect对象的方法与Proxy对象的方法一一对应，只要是Proxy对象的方法，就能在Reflect对象上找到对应的方法。这就让Proxy对象可以方便地调用对应的Reflect方法，完成默认行为，作为修改行为的基础。也就是说，不管Proxy怎么修改默认行为，你总可以在Reflect上获取默认行为。**

**基本用法:**

**get与set**

****

**has**

****

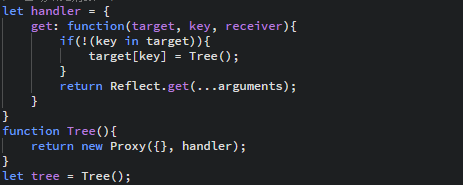
**下面是一些实战场景：**

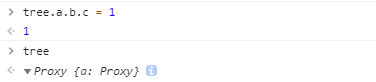
**1、自动填充对象**

**现在若有个对象obj， obj.a.b.c = 1这种赋值操作是不被原生js所支持的。这个时候我们就可以使用proxy**

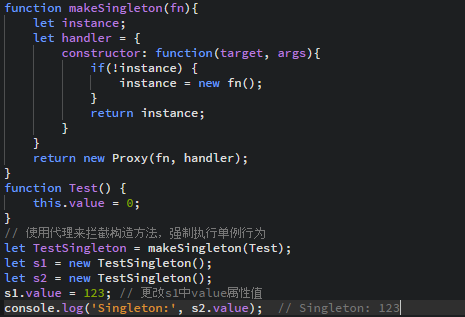
**如下图所示，由于obj.a.b时会先获取a的值，因此声明一个proxy对象，拦截其get操作；若赋值过程中其a属性值为空则在get拦截方法中给其赋值此proxy对象。之后我们即可直接进行obj.a.b.c = 1操作，obj属性值会自动填充为对象**

**其中receiver指操作行为所针对的对象，一般指proxy实例本身**

****

****

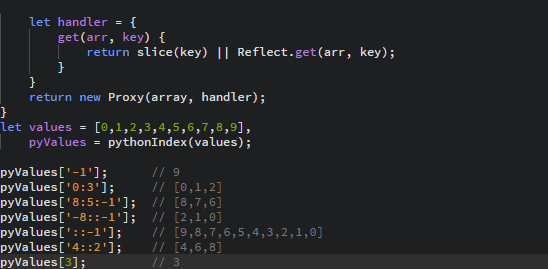
**2、利用proxy实现单例模式**

****

**3、类似python的数组切片**

x:y:z, 其中x代表开始位置，y代表结束位置，z代表间距

****

****

我们还可以利用proxy来实现默认值、校验器、缓存等功能

**参考:** <http://es6.ruanyifeng.com/#docs/proxy>