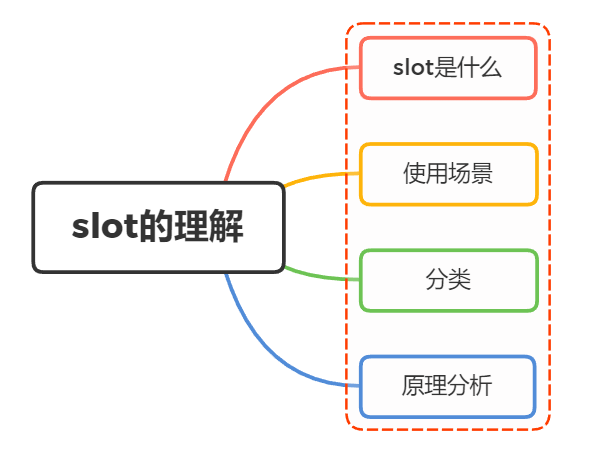
# 面试官：说说你对slot的理解？slot使用场景有哪些？



## 一、slot是什么

在HTML中 slot 元素 ，作为 Web Components 技术套件的一部分，是Web组件内的一个占位符

该占位符可以在后期使用自己的标记语言填充

举个栗子

<template id="element-details-template">  
 <slot name="element-name">Slot template</slot>  
</template>  
<element-details>  
 <span slot="element-name">1</span>  
</element-details>  
<element-details>  
 <span slot="element-name">2</span>  
</element-details>

template不会展示到页面中，需要用先获取它的引用，然后添加到DOM中，

customElements.define('element-details',  
 class extends HTMLElement {  
 constructor() {  
 super();  
 const template = document  
 .getElementById('element-details-template')  
 .content;  
 const shadowRoot = this.attachShadow({mode: 'open'})  
 .appendChild(template.cloneNode(true));  
 }  
})

在Vue中的概念也是如此

Slot 艺名插槽，花名“占坑”，我们可以理解为solt在组件模板中占好了位置，当使用该组件标签时候，组件标签里面的内容就会自动填坑（替换组件模板中slot位置），作为承载分发内容的出口

可以将其类比为插卡式的FC游戏机，游戏机暴露卡槽（插槽）让用户插入不同的游戏磁条（自定义内容）

放张图感受一下 

## 二、使用场景

通过插槽可以让用户可以拓展组件，去更好地复用组件和对其做定制化处理

如果父组件在使用到一个复用组件的时候，获取这个组件在不同的地方有少量的更改，如果去重写组件是一件不明智的事情

通过slot插槽向组件内部指定位置传递内容，完成这个复用组件在不同场景的应用

比如布局组件、表格列、下拉选、弹框显示内容等

## 三、分类

slot可以分来以下三种：

* 默认插槽
* 具名插槽
* 作用域插槽

### 默认插槽

子组件用<slot>标签来确定渲染的位置，标签里面可以放DOM结构，当父组件使用的时候没有往插槽传入内容，标签内DOM结构就会显示在页面

父组件在使用的时候，直接在子组件的标签内写入内容即可

子组件Child.vue

<template>  
 <slot>  
 <p>插槽后备的内容</p>  
 </slot>  
</template>

父组件

<Child>  
 <div>默认插槽</div>   
</Child>

### 具名插槽

子组件用name属性来表示插槽的名字，不传为默认插槽

父组件中在使用时在默认插槽的基础上加上slot属性，值为子组件插槽name属性值

子组件Child.vue

<template>  
 <slot>插槽后备的内容</slot>  
 <slot name="content">插槽后备的内容</slot>  
</template>

父组件

<child>  
 <template v-slot:default>具名插槽</template>  
 <!-- 具名插槽⽤插槽名做参数 -->  
 <template v-slot:content>内容...</template>  
</child>

### 作用域插槽

子组件在作用域上绑定属性来将子组件的信息传给父组件使用，这些属性会被挂在父组件v-slot接受的对象上

父组件中在使用时通过v-slot:（简写：#）获取子组件的信息，在内容中使用

子组件Child.vue

<template>   
 <slot name="footer" testProps="子组件的值">  
 <h3>没传footer插槽</h3>  
 </slot>  
</template>

父组件

<child>   
 <!-- 把v-slot的值指定为作⽤域上下⽂对象 -->  
 <template v-slot:default="slotProps">  
 来⾃⼦组件数据：{{slotProps.testProps}}  
 </template>  
 <template #default="slotProps">  
 来⾃⼦组件数据：{{slotProps.testProps}}  
 </template>  
</child>

### 小结：

* v-slot属性只能在<template>上使用，但在只有默认插槽时可以在组件标签上使用
* 默认插槽名为default，可以省略default直接写v-slot
* 缩写为#时不能不写参数，写成#default
* 可以通过解构获取v-slot={user}，还可以重命名v-slot="{user: newName}"和定义默认值v-slot="{user = '默认值'}"

## 四、原理分析

slot本质上是返回VNode的函数，一般情况下，Vue中的组件要渲染到页面上需要经过template -> render function -> VNode -> DOM 过程，这里看看slot如何实现：

编写一个buttonCounter组件，使用匿名插槽

Vue.component('button-counter', {  
 template: '<div> <slot>我是默认内容</slot></div>'  
})

使用该组件

new Vue({  
 el: '#app',  
 template: '<button-counter><span>我是slot传入内容</span></button-counter>',  
 components:{buttonCounter}  
})

获取buttonCounter组件渲染函数

(function anonymous(  
) {  
with(this){return \_c('div',[\_t("default",[\_v("我是默认内容")])],2)}  
})

\_v表示穿件普通文本节点，\_t表示渲染插槽的函数

渲染插槽函数renderSlot（做了简化）

function renderSlot (  
 name,  
 fallback,  
 props,  
 bindObject  
) {  
 // 得到渲染插槽内容的函数   
 var scopedSlotFn = this.$scopedSlots[name];  
 var nodes;  
 // 如果存在插槽渲染函数，则执行插槽渲染函数，生成nodes节点返回  
 // 否则使用默认值  
 nodes = scopedSlotFn(props) || fallback;  
 return nodes;  
}

name属性表示定义插槽的名字，默认值为default，fallback表示子组件中的slot节点的默认值

关于this.$scopredSlots是什么，我们可以先看看vm.slot

function initRender (vm) {  
 ...  
 vm.$slots = resolveSlots(options.\_renderChildren, renderContext);  
 ...  
}

resolveSlots函数会对children节点做归类和过滤处理，返回slots

function resolveSlots (  
 children,  
 context  
 ) {  
 if (!children || !children.length) {  
 return {}  
 }  
 var slots = {};  
 for (var i = 0, l = children.length; i < l; i++) {  
 var child = children[i];  
 var data = child.data;  
 // remove slot attribute if the node is resolved as a Vue slot node  
 if (data && data.attrs && data.attrs.slot) {  
 delete data.attrs.slot;  
 }  
 // named slots should only be respected if the vnode was rendered in the  
 // same context.  
 if ((child.context === context || child.fnContext === context) &&  
 data && data.slot != null  
 ) {  
 // 如果slot存在(slot="header") 则拿对应的值作为key  
 var name = data.slot;  
 var slot = (slots[name] || (slots[name] = []));  
 // 如果是tempalte元素 则把template的children添加进数组中，这也就是为什么你写的template标签并不会渲染成另一个标签到页面  
 if (child.tag === 'template') {  
 slot.push.apply(slot, child.children || []);  
 } else {  
 slot.push(child);  
 }  
 } else {  
 // 如果没有就默认是default  
 (slots.default || (slots.default = [])).push(child);  
 }  
 }  
 // ignore slots that contains only whitespace  
 for (var name$1 in slots) {  
 if (slots[name$1].every(isWhitespace)) {  
 delete slots[name$1];  
 }  
 }  
 return slots  
}

\_render渲染函数通过normalizeScopedSlots得到vm.$scopedSlots

vm.$scopedSlots = normalizeScopedSlots(  
 \_parentVnode.data.scopedSlots,  
 vm.$slots,  
 vm.$scopedSlots  
);

作用域插槽中父组件能够得到子组件的值是因为在renderSlot的时候执行会传入props，也就是上述\_t第三个参数，父组件则能够得到子组件传递过来的值

## 参考文献

* https://juejin.cn/post/6844903817746628615#heading-4
* https://developer.mozilla.org/en-US/docs/Web/Web\_Components/Using\_templates\_and\_slots
* https://vue3js.cn/docs/zh
* https://segmentfault.com/a/1190000019492734?utm\_source=tag-newest