1. **Html**
2. iframe有哪些优缺点？

**优点：**

* iframe 能够原封不动的把嵌入的网页展现出来。
* 如果有多个网页引用 iframe，那么你只需要修改 iframe 的内容，就可以实现调用的每一个页面内容的更改，方便快捷。
* 网页如果为了统一风格，头部和版本都是一样的，就可以写成一个页面，用 iframe 来嵌套，可以增加代码的可重用。
* 如果遇到加载缓慢的第三方内容如图标和广告，这些问题可以由 iframe 来解决。

#### **缺点：**

* 框架结构中出现各种滚动条
* iframe 会阻塞主页面的 Onload 事件
* 搜索引擎的检索程序无法解读这种页面，不利于 SEO
* iframe 和主页面共享连接池，而浏览器对相同域的连接有限制，所以会影响页面的并行加载。

1. 简述一下 src 与 href 的区别。

答案：src 用于引用资源，替换当前元素；href 用于在当前文档和引用资源之间确立联系。

1. 前端页面有哪三层构成，分别是什么？作用是什么？

答案：分成：结构层（对网页内容的语义含义做出了描述，但这些标签不包含任何关于如何显示有关内容的信息）、表示层（由CSS负责创建，解决了页面“如何显示内容”的问题）、行为层（由脚本负责。解决了页面上“内容应该如何对事件作出反应”的问题）。

1. 你能描述一下渐进增强和优雅降级之间的不同吗?

答案：

渐进增强 progressive enhancement：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

（一开始保证最基本的功能，再改进和追加功能）

优雅降级 graceful degradation：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

（一开始就构建完整的功能，再针对低版本浏览器进行兼容。）

区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，而渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

1. html 常见兼容性问题？

答案：

1.双边距 BUG float 引起的，解决办法: 使用 display解决

2.3 像素问题 使用 float 引起的，解决办法: 使用 dislpay:inline -3px

3.超链接 hover 点击后失效，解决办法: 使用正确的书写顺序 link visited hover active

4.Ie z-index 问题，解决办法: 给父级添加 position:relative

5.Png 透明 ，解决办法: 使用 js 代码

6.Min-height 最小高度 ，解决办法: ！Important 解决

7.select 在 ie6 下遮盖，解决办法: 使用 iframe 嵌套

8.为什么没有办法定义 1px 左右的宽度容器，解决办法: （IE6 默认的行高造成的，使用 over:hidden,zoom:0.08 line-height:1px）

1. 什么是渐进式渲染（progressive rendering）？

答案：渐进式渲染是用于提高网页性能（尤其是提高用户感知的加载速度），以尽快呈现页面的技术。

在以前互联网带宽较小的时期，这种技术更为普遍。如今，移动终端的盛行，而移动网络往往不稳定，渐进式渲染在现代前端开发中仍然有用武之地。

一些举例：

图片懒加载——页面上的图片不会一次性全部加载。当用户滚动页面到图片部分时，JavaScript 将加载并显示图像。

确定显示内容的优先级（分层次渲染）——为了尽快将页面呈现给用户，页面只包含基本的最少量的 CSS、脚本和内容，然后可以使用延迟加载脚本或监听DOMContentLoaded/load事件加载其他资源和内容。

异步加载 HTML 片段——当页面通过后台渲染时，把 HTML 拆分，通过异步请求，分块发送给浏览器。

1. **Css**
2. 为什么要初始化CSS样式

答案：因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。

1. display:none与visibility：hidden的区别？

答案：display：none 不显示对应的元素，在文档布局中不再分配空间（回流+重绘）  
visibility：hidden 隐藏对应元素，在文档布局中仍保留原来的空间（重绘）

1. position跟display、overflow、float这些特性相互叠加后会怎么样？

答案： display属性规定元素应该生成的框的类型；position属性规定元素的定位类型；float属性是一种布局方式，定义元素在哪个方向浮动。

类似于优先级机制：position：absolute/fixed优先级最高，有他们在时，float不起作用，display值需要调整。float 或者absolute定位的元素，只能是块元素或表格。

1. **JS**
2. js中let和const有什么用？

答案：在现代js中，let＆const是创建变量的不同方式。 在早期的js中，咱们使用var关键字来创建变量。 let＆const关键字是在ES6版本中引入的，其目的是在js中创建两种不同类型的变量，一种是不可变的，另一种是可变的。

const:它用于创建一个不可变变量。不可变变量是指其值在程序的整个生命周期中永不改变的变量。

let: let用于创建一个可变变量，可变变量是像var这样的普通变量，可以任意次数地更改。

1. JS中如何将页面重定向到另一个页面？

答案：使用 location.href：window.location.href=“https://www.test.com/”

使用 location.replace： window.location.replace(" [https://www.test.com/;");](https://www.test.com/;\");)

Vue:

this.$router.push

this.$router.replace

#### 解释一下什么是 promise ？

答案：promise是js中的一个对象，用于生成可能在将来产生结果的值。 值可以是已解析的值，也可以是说明为什么未解析该值的原因。

promise 可以有三种状态:

pending：初始状态，既不是成功也不是失败

fulfilled：意味着操作完全成功

rejected：意味着操作失败

一个等待状态的promise对象能够成功后返回一个值，也能失败后带回一个错误  
当这两种情况发生的时候，处理函数会排队执行通过then方法会被调用。

#### 数组去重复的方法有哪些

答案：1：使用 set；2、使用 filter；3、使用 for 循环

#### 列出JS基本和非基本数据类型之间的一些区别？

答案：1.目前JS中有6种基本数据类型: Undefined、Null、Boolean、Number、Symbol 和 String。还有1种复杂的数据类型————Object，Object本质上是由一组无序的名值对组成的。Object、Array和Function则属于引用类型。

2.基本数据类型是不可变的，而非基本数据类型是可变的。

3.基本数据类型是不可变的，因为它们一旦创建就无法更改，但非基本数据类型刚可更改，意味着一旦创建了对象，就可以更改它。

4.将基本数据类型与其值进行比较，这意味着如果两个值具有相同的数据类型并具有相同的值，那么它们是严格相等的。

5.非基本数据类型不与值进行比较。例如，如果两个对象具有相同的属性和值，则它们严格不相等。

#### JS中的宿主对象与原生对象有何不同？

答案：宿主对象:这些是运行环境提供的对象。这意味着它们在不同的环境下是不同的。例如，浏览器包含像windows这样的对象，但是Node.js环境提供像Node List这样的对象。

原生对象:这些是JS中的内置对象。它们也被称为全局对象，因为如果使用JS，内置对象不受是运行环境影响。

#### JS 中 == 和 === 区别是什么？

答案：

1、对于string,number等基础类型，==和===有区别

1）不同类型间比较，==之比较“转化成同一类型后的值”看“值”是否相等，=== 如果类型不同，其结果就是不等。  
 2）同类型比较，直接进行“值”比较，两者结果一样。

2、对于Array,Object等高级类型，==和===没有区别

进行“指针地址”比较。

3、基础类型与高级类型，==和===有区别

1）对于==，将高级转化为基础类型，进行“值”比较。  
 2）因为类型不同，===结果为false。

#### BOM 和 DOM 的关系

答案：BOM全称Browser Object Model，即浏览器对象模型，主要处理浏览器窗口和框架。

DOM全称Document Object Model，即文档对象模型，是 HTML 和XML 的应用程序接口（API），遵循W3C 的标准，所有浏览器公共遵守的标准。

JS是通过访问BOM（Browser Object Model）对象来访问、控制、修改客户端(浏览器)，由于BOM的window包含了document，window对象的属性和方法是直接可以使用而且被感知的，因此可以直接使用window对象的document属性，通过document属性就可以访问、检索、修改XHTML文档内容与结构。因为document对象又是DOM的根节点。

可以说，BOM包含了DOM(对象)，浏览器提供出来给予访问的是BOM对象，从BOM对象再访问到DOM对象，从而js可以操作浏览器以及浏览器读取到的文档。

#### 解释JS中的事件冒泡和事件捕获

答案：事件捕获和冒泡: 在html DOM API中，有两种事件传播方法，它们决定了接收事件的顺序。两种方法是事件冒泡和事件捕获。第一个方法事件冒泡将事件指向其预期的目标，第二个方法称为事件捕获，其中事件向下到达元素。

事件捕获：

捕获过程很少被使用，但是当它被使用时，它被证明是非常有用的。这个过程也称为滴流模式。在这个过程中，事件首先由最外层的元素捕获，然后传播到最内部的元素。

事件冒泡：

冒泡的工作原理与冒泡类似，事件由最内部的元素处理，然后传播到外部元素。

1. **Vue**

#### 说说你对 SPA 单页面的理解，以及它的优缺点

答案：SPA（ single-page application ）仅在 Web 页面初始化时加载相应的 HTML、JavaScript 和 CSS。一旦页面加载完成，SPA 不会因为用户的操作而进行页面的重新加载或跳转；取而代之的是利用路由机制实现 HTML 内容的变换，UI 与用户的交互，避免页面的重新加载。

优点：

用户体验好、快，内容的改变不需要重新加载整个页面，避免了不必要的跳转和重复渲染；

基于上面一点，SPA 相对对服务器压力小；

前后端职责分离，架构清晰，前端进行交互逻辑，后端负责数据处理；

缺点：

初次加载耗时多：为实现单页 Web 应用功能及显示效果，需要在加载页面的时候将 JavaScript、CSS 统一加载，部分页面按需加载；

前进后退路由管理：由于单页应用在一个页面中显示所有的内容，所以不能使用浏览器的前进后退功能，所有的页面切换需要自己建立堆栈管理；

SEO 难度较大：由于所有的内容都在一个页面中动态替换显示，所以在 SEO 上其有着天然的弱势。

1. Vue生命周期



1. Vue 的父组件和子组件生命周期钩子函数执行顺序？

答案：Vue 的父组件和子组件生命周期钩子函数执行顺序可以归类为以下 4 部分：

加载渲染过程：

父 beforeCreate -> 父 created -> 父 beforeMount -> 子 beforeCreate -> 子 created -> 子 beforeMount -> 子 mounted -> 父 mounted

子组件更新过程：

父 beforeUpdate -> 子 beforeUpdate -> 子 updated -> 父 updated

父组件更新过程：

父 beforeUpdate -> 父 updated

销毁过程：

父 beforeDestroy -> 子 beforeDestroy -> 子 destroyed -> 父 destroyed

1. 在哪个生命周期内调用异步请求？

答案：可以在钩子函数 created、beforeMount、mounted 中进行调用，因为在这三个钩子函数中，data 已经创建，可以将服务端端返回的数据进行赋值。但是本人推荐在 created 钩子函数中调用异步请求，因为在 created 钩子函数中调用异步请求有以下优点：

能更快获取到服务端数据，减少页面 loading 时间；

ssr 不支持 beforeMount 、mounted 钩子函数，所以放在 created 中有助于一致性；

1. 父组件可以监听到子组件的生命周期吗？

答案：比如有父组件 Parent 和子组件 Child，如果父组件监听到子组件挂载 mounted 就做一些逻辑处理，可以通过以下写法实现：

// Parent.vue

以上需要手动通过 $emit 触发父组件的事件，更简单的方式可以在父组件引用子组件时通过 @hook 来监听即可，如下所示：

// Parent.vue

当然 @hook 方法不仅仅是可以监听 mounted，其它的生命周期事件，例如：created，updated 等都可以监听。

## 谈谈你对 keep-alive 的了解？

答案：keep-alive 是 Vue 内置的一个组件，可以使被包含的组件保留状态，避免重新渲染 ，其有以下特性：

一般结合路由和动态组件一起使用，用于缓存组件；

提供 include 和 exclude 属性，两者都支持字符串或正则表达式， include 表示只有名称匹配的组件会被缓存，exclude 表示任何名称匹配的组件都不会被缓存 ，其中 exclude 的优先级比 include 高；

对应两个钩子函数 activated 和 deactivated ，当组件被激活时，触发钩子函数 activated，当组件被移除时，触发钩子函数 deactivated。

## vue-router 路由模式有几种？

答案：vue-router 有 3 种路由模式：hash、history、abstract，

其中，3 种路由模式的说明如下：

hash: 使用 URL hash 值来作路由。支持所有浏览器，包括不支持 HTML5 History Api 的浏览器；

history : 依赖 HTML5 History API 和服务器配置。具体可以查看 HTML5 History 模式；

abstract : 支持所有 JavaScript 运行环境，如 Node.js 服务器端。如果发现没有浏览器的 API，路由会自动强制进入这个模式.

## vue-router 中常用的 hash 和 history 路由模式实现原理吗？

**（1）hash 模式的实现原理**

早期的前端路由的实现就是基于 location.hash 来实现的。其实现原理很简单，location.hash 的值就是 URL 中 # 后面的内容。比如下面这个网站，它的 location.hash 的值为 '#search'：

https://www.word.com#search

hash 路由模式的实现主要是基于下面几个特性：

URL 中 hash 值只是客户端的一种状态，也就是说当向服务器端发出请求时，hash 部分不会被发送；

hash 值的改变，都会在浏览器的访问历史中增加一个记录。因此我们能通过浏览器的回退、前进按钮控制hash 的切换；

可以通过 a 标签，并设置 href 属性，当用户点击这个标签后，URL 的 hash 值会发生改变；或者使用 JavaScript 来对 loaction.hash 进行赋值，改变 URL 的 hash 值；

我们可以使用 hashchange 事件来监听 hash 值的变化，从而对页面进行跳转（渲染）。

**（2）history 模式的实现原理**

HTML5 提供了 History API 来实现 URL 的变化。其中做最主要的 API 有以下两个：history.pushState() 和 history.repalceState()。这两个 API 可以在不进行刷新的情况下，操作浏览器的历史纪录。唯一不同的是，前者是新增一个历史记录，后者是直接替换当前的历史记录，如下所示：

window.history.pushState(null, null, path);

history 路由模式的实现主要基于存在下面几个特性：

pushState 和 repalceState 两个 API 来操作实现 URL 的变化 ；

我们可以使用 popstate 事件来监听 url 的变化，从而对页面进行跳转（渲染）；

history.pushState() 或 history.replaceState() 不会触发 popstate 事件，这时我们需要手动触发页面跳转（渲染）。

## Proxy 与 Object.defineProperty 优劣对比

答案：Proxy 的优势如下:

Proxy 可以直接监听对象而非属性；

Proxy 可以直接监听数组的变化；

Proxy 有多达 13 种拦截方法,不限于 apply、ownKeys、deleteProperty、has 等等是 Object.defineProperty 不具备的；

Proxy 返回的是一个新对象,我们可以只操作新的对象达到目的,而 Object.defineProperty 只能遍历对象属性直接修改；

Proxy 作为新标准将受到浏览器厂商重点持续的性能优化，也就是传说中的新标准的性能红利；

Object.defineProperty 的优势如下:

兼容性好，支持 IE9，而 Proxy 的存在浏览器兼容性问题,而且无法用 polyfill 磨平，因此 Vue 的作者才声明需要等到下个大版本( 3.0 )才能用 Proxy 重写。

## 虚拟 DOM 实现原理？

答案：虚拟 DOM 的实现原理主要包括以下 3 部分：

用 JavaScript 对象模拟真实 DOM 树，对真实 DOM 进行抽象；

diff 算法 — 比较两棵虚拟 DOM 树的差异；

pach 算法 — 将两个虚拟 DOM 对象的差异应用到真正的 DOM 树。

## Vue 中的 key 有什么作用？

答案：key 是为 Vue 中 vnode 的唯一标记，通过这个 key，我们的 diff 操作可以更准确、更快速。

Vue 的 diff 过程可以概括为：oldCh 和 newCh 各有两个头尾的变量 oldStartIndex、oldEndIndex 和 newStartIndex、newEndIndex，它们会新节点和旧节点会进行两两对比，即一共有4种比较方式：newStartIndex 和oldStartIndex 、newEndIndex 和 oldEndIndex 、newStartIndex 和 oldEndIndex 、newEndIndex 和 oldStartIndex，如果以上 4 种比较都没匹配，如果设置了key，就会用 key 再进行比较，在比较的过程中，遍历会往中间靠，一旦 StartIdx > EndIdx 表明 oldCh 和 newCh 至少有一个已经遍历完了，就会结束比较。所以 Vue 中 key 的作用是：key 是为 Vue 中 vnode 的唯一标记，通过这个 key，我们的 diff 操作可以更准确、更快速

更准确：因为带 key 就不是就地复用了，在 sameNode 函数 a.key === b.key 对比中可以避免就地复用的情况。所以会更加准确。

更快速：利用 key 的唯一性生成 map 对象来获取对应节点，比遍历方式更快

## 你有对 Vue 项目进行哪些优化？

答案：

（1）代码层面的优化

v-if 和 v-show 区分使用场景

computed 和 watch 区分使用场景

v-for 遍历必须为 item 添加 key，且避免同时使用 v-if

长列表性能优化

事件的销毁

图片资源懒加载

路由懒加载

第三方插件的按需引入

优化无限列表性能

服务端渲染 SSR or 预渲染

（2）Webpack 层面的优化

Webpack 对图片进行压缩

减少 ES6 转为 ES5 的冗余代码

提取公共代码

模板预编译

提取组件的 CSS

优化 SourceMap

构建结果输出分析

Vue 项目的编译优化

（3）基础的 Web 技术的优化

开启 gzip 压缩

浏览器缓存

CDN 的使用

使用 Chrome Performance 查找性能瓶颈

## v-for 和 v-if 谁的[优先级](https://so.csdn.net/so/search?q=%E4%BC%98%E5%85%88%E7%BA%A7&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/namechenfl/article/details/_blank)高，使用的时候有什么要注意的？

答案：显然v-for优先于v-if被解析

如果同时出现，每次渲染都会先执行循环再判断条件，无论如何循环都不可避免，浪费了性能

要避免出现这种情况，则在外层嵌套template(div)，在这一层进行v-if判断，然后在内部进行v-for循环

如果条件出现在循环内部，可通过计算属性提前过滤掉那些不需要显示的项

## 说说你使用 Vue 框架踩过最大的坑是什么？怎么解决的？

有做过项目优化吗?

跨专业?