1. **闭包**

读取其他函数内部变量的函数

优点：防止污染全局变量，具有缓存

缺点：参数和变量不会被垃圾回收机制回收（内存泄露，消耗内存）

1. **原型链。**
2. **继承方式。**
3. **事件代理：利用冒泡实现，减少事件注册，节约内存。**
4. **DOM事件流：捕获，目标，冒泡。**
5. **This指向。**

指向函数的直接调用者

New 的对象，this指向这个对象。

事件中，指向触发事件的对象。

1. **阻止冒泡**：e.stopPropagation() IE:window.event.cancelBuble=false
2. **阻止默认行为：**e.preventDefault() IE window.event.returnValue = false

return false

1. **new 过程。**
2. 创建一个空对象，this指向空对象，继承函数原型
3. 通过this添加属性和方法。
4. 返回this指向的新对象
5. **手写New。**
6. **跨域**

协议，域名，端口有任何一个的不同

JsonP，nginx代理，后端设置CORS跨域资源共享。Access-Control-Allow-Origin

Access-Control-Allow-method

**function** **jsonp**(req){

**var** script = document.createElement('script');

**var** url = req.url + '?callback=' + req.callback;

script.src = url;

document.getElementsByTagName('head')[0].appendChild(script);

}

1. **模块化(AMD,CommonJs)**
2. **XSS和CSRF**

XSS（跨域脚本攻击。）：通过修改 HTML 节点或者执⾏ JS 代码来攻击⽹站。策略：对于引号，尖括号，斜杠进⾏转义。

CSRF（跨站请求伪造）: 就是利⽤⽤户的登录态发起恶意请求。

策略：Token 验证

1. **Promise实现**
2. defer 并⾏加载 js ⽂件，会按照⻚⾯上 script 标签的顺序执⾏，文档解析完成后执行

async 并⾏加载 js ⽂件，下载完成⽴即执⾏，不会按照⻚⾯上 script 标签的顺序执⾏.

1. 深拷贝和浅拷贝

浅拷贝 {…obj} ,object.assign({},obj,obj1,…)

深拷贝 JSON.parse(JSON.stringify()) 会丢失function,undefined，

递归手写

1. 手写防抖和节流
2. Apply，bind，call手写
3. 为什么单线程：多个线程操作dom造成矛盾，浏览器无法正确运行
4. 浏览器事件循环
5. **Node事件循环：**

**主线成处理请求，I/O操作进行异步处理。主线程中维护了一个事件队列，接到请求，将事件放入队列，然后继续接受其他请求。当主线程空闲是（没有请求接入），就开始循环事件队列。非IO任务亲自处理，并通过回调函数回到上层调用，IO任务线程池中拿出线程处理这个事件。当线程池IO任务完成后，就执行指定的回调函数，并放到事件队列尾部，等待事件循环，循环到这个事件时，就直接处理并返回上层调用。**

**微任务中会先执行process.next tick 然后再执行其他微任务，**

**宏任务**

**1.timers 阶段：这个阶段执行timer（setTimeout、setInterval）的回调**

**2.pendingg callbacks 待定回调阶段：执行一些系统调用，**

**3.idle, prepare 阶段：仅node内部使用**

**4.poll 阶段：检索获取新的I/O事件,执行IO回调**

**5.check 阶段：执行 setImmediate() 的回调**

**6.close callbacks 阶段：执行 socket 的 close 事件回调**

1. 列表和树转换
2. ==和===
3. **变量提升：JS 解释器会找出需要提升的变量和函数，并且给他们 提前在内存中开辟好空间，函数的话会将整个函数存⼊内存中，变量只声明并 且赋值为 undefined ，所以在第⼆个阶段，也就是代码执⾏阶段，我们可以直接提前使⽤**
4. **Instance of判断对象的原型链中是不是能找到类型的 prototype**
5. **Async await原理（generator）**[**https://zhuanlan.zhihu.com/p/66593213**](https://zhuanlan.zhihu.com/p/66593213)
6. **Generator生成器：用\*声明。调用时会被yield挂起，通过调用next继续执行，调用next返回一个对象{ value: '开始', done: false }**
7. **暂时性死区：声明变量之前，该变量都是不可用的（let不会变量提升）**
8. **纯函数（一个函数的返回结果只依赖于它的参数，并且在执行过程里面没有副作用）**
9. 两个对象比较
10. **Object和Map：Object无法使用非字符串值作为键名，但Map的键名可以是任意类型。**
11. **Map和weakmap区别**

**WeakMap只接受对象作为key，没有遍历方法和清空方法。**

**WeakMap的key所引用的对象都是弱引用，只要对象的其他引用被删除，垃圾回收机制就会释放该对象占用的内存，从而避免内存泄漏。**

1. **Set和weakset区别**

**Set可以遍历**

**weakSet成员都是弱引用，可以被垃圾回收机制回收,不可遍历。**

1. 变量**在当前作用域中没有查到值，就会向上级作用域去查，直到查到全局作用域，**
2. **垃圾回收机制**

**变量清除：当变量进入执行环境（函数中声明变量,执行时）的时候，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”，在离开环境之后还有的变量则是需要被删除的变量**

1. 输入网址到浏览器渲染过程

解析url获取协议，域名，端口号，通过DNS域名解析系统解析域名得到对应的ip，连接TCP，向服务器发起http请求，服务器交给后台处理完成后返回数据，浏览器接受文件，将HTML解析成DOM树，根据CSS生成CSSOM，树。将DOM树和CSSOM树结合生成渲染树,加载解析JS，渲染页面。

1. 常见状态码

400 服务器无法理解请求的格式

401 未授权

403 禁止访问

404 url找不到资源

500 服务端报错

200 请求成功

301 重定向，永久移动

1. **TCP三次握手，四次挥手**

三次握手：防止失效的连接请求报文段被服务端接收，从而产生错误（若客户端向服务端发送的连接请求丢失，客户端等待应答超时后就会再次发送连接请求）

1. HTTP和HTTPS的区别：端口号80和443 https更加安全
2. Get和post GET参数通过URL传递，POST放在Request body中。

GET产生一个TCP数据包；POST产生两个TCP数据包。POST 比 GET 更安全（get会缓存）

1. 前端性能优化汇总

精灵图

压缩js和css

图片懒加载

在js中尽量减少闭包的使用

1. 定时器不准的原因：单线程，异步 二个参数（延时时间）只是告诉 JavaScript 再过多长时间把当前任务添加到队列中。如果队列是空的，那么添加的代码会立即执行；如果队列不是空的，那么它就要等前面的代码执行完了以后再执行。
2. Symbol
3. 设计模式
4. ES6 let 箭头函数 参数默认值 解构赋值 class promise
5. Fetch请求两次

第一次发送了一个Options请求，询问服务器是否支持修改的请求头，如果服务器支持，则在第二次中发送真正的请求。