JavaScript（补充）：

value type vs reference type

value type: Boolean, String, Number, null, undefined. Do clone first, so change value will only change the copy.

reference type: Array, Function. Object. 指针 pointer change the value under the pointer will change the real value.

Shallow clone vs deep clone

Shallow clone: Object.assign, spread operator

Const clone ={…data} 只复制初始的一层

Deep clone: use lodash, or JSON.parse(JSON.stringify(obj)) (limited except including function date…) Lodash library

Var: variable hoisting, function scope (function keyword also has hoisting).

let/const: no variable hoisting, block scope only worked inside the loop.

Closure

Simple: a function returns another function. The returned function can access the inner scope of the first function.

Full: A closure is an inner function that has access to the outer function's variables. The closure has three scope chains: it has access to its own scope, it has access to the outer function's variables, and it has access to global variables.

async works

Event loop. We want non-blocking and avoid DOM render conflicting

V8 engine Call main stac run and web api run while main stac running

then when apis asy operation run finished it will went to callback

queue then event loop call the things in the callback queue to run in

the main stac.

Promise:

Promise is a cleaner way for running asynchronous tasks. Avoid callback hell.

Promise use methods like ".then" and ".catch" to deal with all kinds of asynchronous operation like “reslove” and “reject”.

Async/await is the same thing as Promise. It is just syntax sugar that allows us to write async code like sync code.

“this”:

“this" means the environment object where the function is running.

With “Function” keyword:

“this” in constructor function. -> the function called with “new” -> this refers the instance

“this” in a object method function -> refers to who call the function (the thing before the “.”)

“this” in a plain function -> window

“this” in event handler -> the element fires the event

Call: call() 方法使用一个指定的 this 值和单独给出的一个或多个参数来调用一个函数。

apply: apply() 方法调用一个具有给定 this 值的函数，以及以一个数组（或一个类数组对象）的形式提供的参数。

Bind: 调用 f.bind(someObject)会创建一个与 f 具有相同函数体和作用域的函数，但是在这个新函数中，this 将永久地被绑定到了 bind 的第一个参数，无论这个函数是如何被调用的。

Prototype: 所有的 JavaScript 对象都会从一个 prototype（原型对象）中继承属性和方法。

所有 JavaScript 中的对象都是位于原型链顶端的 Object 的实例。

只要创建了一个新函数，就会根据一组特定的规则为该函数创建一个 prototype 属性，默认情况下 prototype 属性会默认获得一个 constructor(构造函数)属性。同时自动生成一个__proto__属性，该属性指向 Person 的 prototype，可以访问到 prototype 内定义的方法。