前端界面的三层构成

答： 结构层 表示层 行为层

HTML CSS JS

css渲染顺序

显示属性 > 自身属性 > 文本属性

display width font-size

position height color

clear:both padding text-align

list-style:none

img中的alt和元素的title属性作用

img的alt属性

如果无法显示图像，浏览器将显示alt指定的内容

元素title属性

在鼠标移到元素上时显示title的内容

href和src区别

href

href标识超文本引用，用在link和a等元素上，href是引用和页面关联，是在当前元素和引用资源之间建立联系

若在文档中添加href ，浏览器会识别该文档为 CSS 文件，就会并行下载资源并且不会停止对当前文档的处理 。这也是为什么建议使用 link 方式加载 CSS，而不是使用 @import 方式。

src

src表示引用资源，替换当前元素，用在img，script，iframe上，src是页面内容不可缺少的一部分。

当浏览器解析到src ，会暂停其他资源的下载和处理（图片不会暂停其他资源下载），直到将该资源加载、编 译、执行完毕，类似于将所指向资源应用到当前内容。这也是为什么建议把 js 脚本放在底部而不是头部的原 因。

浏览器的渲染过程

解析HTML生成DOM树。

解析CSS生成CSSOM规则树。

将DOM树与CSSOM规则树合并在一起生成渲染树。

遍历渲染树开始布局，计算每个节点的位置大小信息。

将渲染树每个节点绘制到屏幕。

什么是BFC

1.块级格式化上下文

2.可以通过inline-blocks和table-cells和table-captions和overflow来触发BFC

3.用来解决浮动后造成的文档塌陷

为何会出现浏览器兼容问题

同一产品，版本越老 bug 越多

同一产品，版本越新，功能越多

不同产品，不同标准，不同实现方

处理兼容问题的思路

要不要做

产品的角度（产品的受众、受众的浏览器比例、效果优先还是基本功能优先）

成本的角度 (有无必要做某件事)

2.做到什么程度 \* 让哪些浏览器支持哪些效果

3..如何做 根据兼容需求选择技术框架/库(jquery) 根据兼容需求选择兼容工具(html5shiv.js、respond.js、css reset、normalize.css、Modernizr) \* 条件注释、CSS Hack、js 能力检测做一些修补

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

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。

DOM和BOM有什么区别

DOM

Document Object Model，文档对象模型

DOM 是为了操作文档出现的 API，document 是其的一个对象

DOM和文档有关，这里的文档指的是网页，也就是html文档。DOM和浏览器无关，他关注的是网页本身的 内容。

BOM

Browser Object Model，浏览器对象模型

BOM 是为了操作浏览器出现的 API，window 是其的一个对象

window 对象既为 javascript 访问浏览器提供API，同时在 ECMAScript 中充当 Global 对象

如何解决不同浏览器样式兼容性问题

在确定问题原因和有问题的浏览器后，使用单独的样式表，仅供出现问题的浏览器加载。这种方法需要使用 服务器端渲染。

使用已经处理好此类问题的库，比如 Bootstrap。

使用 autoprefixer 自动生成 CSS 属性前缀。

使用 Reset CSS 或 Normalize.css。

使用 CSS 预处理的优缺点分别是什么？

优点：

提高 CSS 可维护性。

易于编写嵌套选择器。

引入变量，增添主题功能。可以在不同的项目中共享主题文件。

通过混合（Mixins）生成重复的 CSS。

Splitting your code into multiple files. CSS files can be split up too but doing so will require a HTTP rquest to download each CSS file.

将代码分割成多个文件。不进行预处理的 CSS，虽然也可以分割成多个文件，但需要建立多个 HTTP 请求加 载这些文件。

缺点：

需要预处理工具。

重新编译的时间可能会很慢。

CSS伪元素

CSS中常见的伪元素有:first-line, :first-letter, :before, :after;

:first-line伪元素

为文本首行设置样式：

:first-letter伪元素

为文本首字母设置样式

:before伪元素

在元素的内容前面添加新内容

:after伪元素

在元素的内容后面添加新内容

介绍一下盒子模型

1.IE盒模型：content包含margin、padding、border

2.w3c盒模型：content不包含margin、padding、border

3.弹性盒模型：display：flex；

display的属性值都有哪些？

none, block, inline, inline-block, table, table-row, table-cell, list-item.

inline和inline-block有什么区别？

display:block

block元素会独占一行，多个block元素会各自新起一行。默认情况下，block元素宽度自动填满其父元素宽 度 。

block元素可以设置width,height属性。块级元素即使设置了宽度,仍然是独占一行。

block元素可以设置margin和padding属性。

display:inline-block

简单来说就是将对象呈现为inline对象，但是对象的内容作为block对象呈现。之后的内联对象会被排列在同 一行内。 比如我们可以给一个link（a元素）inline-block属性值，使其既具有block的宽度高度特性又具有 inline的同行特性。

Flex

flex容器中存在两条轴， 横轴和纵轴， 容器中的每个单元称为flex item。

在容器上可以设置6个属性： flex-direction flex-wrap flex-flow justify-content align-items align- content

Flex 项目属性

有六种属性可运用在 item 项目上: 1. order 2. flex-basis 3. flex-grow 4. flex-shrink 5. flex 6. align-self

注意：当设置 flex 布局之后，子元素的 float、clear、vertical-align 的属性将会失效。

不定宽高水平垂直居中

1.通过display：flex；给子元素设置 justify-content:center; align-items:center;

2.通过css3，给父元素设置相对定位，给子元素设置绝对定位，left：50% top：50%

通过transform：translate（-50%，-50%）拉回自身的宽和高

3.通过给父元素设置display：table和绝对定位，给子元素设置display：table-cell

vertical-align：middle text-align：center

CSS中link和@import的区别

link属于HTML标签，而@import是CSS提供的

页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载

import只在IE5以上才能识别，而link是HTML标签，无兼容问题

link方式的样式的权重 高于@import的权重

文本超出部分显示省略号

单行

overflow: hidden;

text-overflow:ellipsis;

white-space: nowrap;

多行

display: -webkit-box;

-webkit-box-orient: vertical;

-webkit-line-clamp: 3; // 最多显示几行

overflow: hidden;

利用伪元素画三角

.info-tab {

position: relative;

}

.info-tab::after {

content: '';

border: 4px solid transparent;

border-top-color: #2c8ac2;

position: absolute;

top: 0;

}

已知父级盒子的宽高，子级img宽高未知，想让img铺满父级盒子且图片不能变形

需要用到css的object-fit属性

div {

width: 200px;

height: 200px;

}

img {

object-fit: cover;

width: 100%;

height: 100%;

}

css hack是什么

由于不同的浏览器，比如Internet Explorer 6,Internet Explorer 7,Mozilla Firefox等，对CSS的解析认识不 一样，因此会导致生成的页面效果不一样，得不到我们所需要的页面效果。

这个时候我们就需要针对不同的浏览器去写不同的CSS，让它能够同时兼容不同的浏览器，能在不同的浏览 器中也能得到我们想要的页面效果。

这个针对不同的浏览器写不同的CSS code的过程，就叫CSS hack,也叫写CSS hack。

去除inline-block元素间间距的方法

移除空格

使用margin负值

使用font-size:0

letter-spacing

word-spacing

同源策略

同源策略可防止 JavaScript 发起跨域请求。源被定义为 URI、主机名和端口号的组合。此策略可防止页面上 的恶意脚本通过该页面的文档对象模型，访问另一个网页上的敏感数据。

同源策略就是浏览器都会使用的一个策略，并且域名，协议，端口相同。

2.同源策略的限制：存储在浏览器中的数据，如localStroage和cookie不能

通过脚本跨域访问，不能通过脚本操作不同域下的DOM，不能通过ajax请求

不同域的数据

3.为什么要使用同源策略：主要是为了安全，如果没有同源限制，存在浏览器中的

cookie等其他数据可以任意读取，不同域下的DOM任意操作，ajax任意请求的话

就会泄漏这些隐私数据

跨域

原因

浏览器的同源策略导致了跨域

作用

用于隔离潜在恶意文件的重要安全机制

解决

1.jsonp js脚本标签script 上有一个src的属性 这个属性无视浏览器的限制 可以直接去访问外部资源

该协议的一个要点就是允许用户传递一个callback参数给服务端，然后服务端返回数据时会将 这个callback参数作为函数名来包裹住JSON数据，

但ajax和jsonp其实本质上是不同的东西。ajax的核心是通过XmlHttpRequest获取非本页内容，而jsonp 的核心则是动态添加

2.cors 后台添加一个白名单直接可以进行访问外部资源

3.iframe代理 让iframe载入一个与你想要通过ajax获取数据的目标页面处在相同域的页面

4.niginx反向代理 由你自己写的接口在后端调用要访问的接口地址并拿回返回值，然后返回给index.html

响应性开发技术

flex vw vh 微信小程序

手淘 flexible 只支持手机

rem+js 全平台

get和post区别

1.最直接的区别，GET请求的参数是放在URL里的，POST请求参数是放在请求body里的

2.GET请求的URL传参有长度限制，而POST请求没有长度限制

3.GET请求的参数只能是SCII码，所以中文需要URL编码，而POST请求传参没有这个限制

什么是事件委托：

事件委托——给父元素绑定事件，用来监听子元素的冒泡事件，并找到是哪个子元素的事件。

事件委托的好处：

事件委托技术可以避免对每个字元素添加事件监听器，减少操作DOM节点的次数，从而减少浏览器的重绘 和重排，提高代码的性能。

使用事件委托，只有父元素与DOM存在交互，其他的操作都是在JS虚拟内存中完成的，这样就大大提高了 性能。

什么时候用事件委托：

当子元素有很多，需要对子元素的时间进行监听的时候

js原生实现事件委托

给父元素 ul 绑定事件，通过addEventListener点击事件click触发

第二步: 监听子元素的冒泡事件

第三步 直接获取

document.getElementsByTagName("ul")[0].addEventListener( 'click',function(e){ alert("点击的内容是:" + e.target.innerHTML )} )

事件循环

事件循环是一个单线程循环，用于监视调用堆栈并检查是否有工作即将在任务队列中完成。如果调用堆栈为 空并且任务队列中有回调函数，则将回调函数出队并推送到调用堆栈中执行。

js执行时，会进入一个执行栈，执行栈里面放的是同步任务，执行代码的时候会遇到异步

代码，像promise.then方法，setTimeout这两个异步任务就会放到任务队列中，等执行栈

里面的同步代码执行完毕，去执行任务队列里面的异步任务，异步任务又分为宏任务和

微任务，.then方法是微任务，setTimeout是宏任务，先执行微任务，在执行宏任务，执行

setTimeout的时候，又回进入执行栈，里面又是同步代码，一直这样循环

栈和队列的区别

栈是一种数据结构，具有后进先出的特点

队列也是一种数据结构，一边进，一边出，先进先出

原型原型链

（1）什么是原型链

由\_\_proto\_\_串起来到object.prototype.\_\_proto\_\_为空的链

当js引擎查找对象的属性时，如果当前对象没有 就会去原型链上去找

浏览器缓存机制

就是把一个已经请求过得资源拷贝存储起来，当下次需要该资源时，浏览器会根据缓存机制决 定直接 使用缓存资源还是再次向服务器发送请求

优点

减少了冗余的数据传输，节省了网费

减少了服务器的负担，大大提升了网站的性能

加快了客户端加载网页的速度

js如何创建，添加， 插入，查找，删除，替换和创建文档碎片

创建：

var DIV=document.createElement( " div " )

添加:

① appendChild( ) 将节点插入到父节点的末尾

② document.body.appendChild( DIV ) 把 DIV 插入body中，并且位于末尾

插入：

insertBefore ( )

document.body.insertBefore( "要插入的节点" , " 原本节点 " )

删除：

document.body.removeChild ( 要删除的节点 )

查找：

getElementById ( )

替换：

创建一个节点

document.body.replaceChild( 创建的节点，要替换的节点 )

文档碎片：

var oFragmeng = document.createDocumentEragment( )

es5和es6的继承

1.es6通过extends继承父类的方法，super方法调用父类构造器并且继承父类的属性

2.通过构造器函数假装类，通过原型方法prototype，继承父类的方法

可以通过父构造器call，apply方法更改this指向，让它指向子类并获得父类属性

闭包

（1）什么是闭包

一个函数嵌套另一个函数，外部函数可以访问内部函数的变量

（2）闭包的适用场景

1.选项卡和焦点轮播图

2.模块化开发 每一个组件就是一个闭包函数

（3）闭包的缺点

造成内存泄漏

（4）为什么会造成内存泄漏（写的不全）

闭包跟作用域和垃圾回收机制有关，而且私有变量占用内存又无法清除就会造成内存泄漏

（5）如何解决内存泄漏

手动清除不需要的变量，并且设置为null

（6）内存泄漏及解决方法

1.内存泄漏指因为疏忽或错误造成程序未能释放已经不在使用的内存的情况。

内存泄漏并不是指内存在物理上的消失，而是应用程序分配某段内存后，

因为设计错误，失去了对该段内存的控制，而造成内存的浪费。

2.

（1）良好的编码习惯，尽量在设计内存的程序段，检测出内存泄漏。

（2）使用了内存分配的函数或变量，使用完毕后，及时清除

（7）垃圾回收机制

1.垃圾回收器定期扫描对象，并计算引用了每个对象的其他的对象的数量，如果

一个对象的引用数量为0（没有其他对象去引用这个对象），或对该对象的唯一

引用是循环的，那么该对象的内存即可回收.

ajax原理

1.创建ajax对象

2.设置请求的方法和接口地址

3.设置请求的编码

4.通过onreadystatechange事件去监听ajax对象请求过程

5.发送请求

脚本的异步加载或异步加载

1.defer：js脚本下载完毕并且DOM创建完毕，立即执行，一般需要操作DOM的时候使用

2.async：js脚本下载完毕就立即执行，不用等待DOM的创建，一般不需要操作DOM的时候使用

简述下proxy代理（拦截器）？

1.proxy代理一般用于数据过滤，是一个拦截器。

2.当我的变量显示或改变的时候，会调用proxy的两个方法，get和set

3.get方法默认就会调用，set方法当变量改变的时候去调用

应用场景：比如用户名，手机号，银号卡号等等，这些隐私信息，需要我们拦截之后，进行处理。把一些信息转化为\*\*\*

let pro=new Proxy({

name:"tom",

age:15,

sex:"男"

},{

get(target,key){ //对象属性显示之前触发get方法 target当前监控对象

console.log("你触发了get方法");

return "机密"+target[key];

},

set(target,key,value){ //对象属性改变之前触发set方法

console.log("你触发了set方法");

target[key]=value+"\*\*\*";

}

})

变量提升

var会使变量提升

一开始使用var声明或者初始化的变量，会将语句提升到当前作用域的顶部，只有声明才会触发提升

let const var

var会使变量提升，这意味着变量可以在声明之前使用，可以重复声明

let和const的区别在于：let允许多次赋值，而const只允许一次。

对象浅拷贝和深拷贝有什么区别

浅拷贝就等于赋值

深拷贝等同于两个相同但又没有任何关系

JSON.parse（ JSON.stringify( 要拷贝的对象) ）

利用Object.prototype.toString.call( )

Object.assign( { }, 要拷贝的)

数组遍历的方法

1.for循环

2.map即可遍历对象又可以遍历数组，并且是变异方法，会更改原数组

并且有返回值

3.forEach遍历 非变异方法，不会更改原数组，没有返回值 遍历对象的键值对

不能遍历对象，只能遍历数组，数组元素是一个对象

4.for...in 遍历 遍历对象返回的是键名

5.for...of 遍历 遍历对象返回的是键值

数组去重

遍历去重 indexOf

函数内声明一个temp

形参. forEach 遍历

if判断temp.indexOf == -1

temp.push

一定得return

ES6

function unique (arr) {

return Array.from(new Set(arr))

}

数据类型

Undefined

Null

Boolean

Number

String

Object

symbol(ES6新增)

如何判断数组与对象

Array.isArray([]) // true

Array.isArray({}) // false

typeof [] // "object"

typeof {} // "object"

Object.prototype.toString.call( )

ES5对象转换成数组的方法

1.Object.value（）把对象的键值转换成数组

2.Object.keys（）把对象的键名转换成数组

3.Object.entries（）把对象的键值对转换成二维数组，每个键值对是一个小数组，外层

一个大数组

JS数组的方法说说

sort排序

push 添加

pop删除

sort排序

reverse反序

toString 数组转换字符串 非变异

join 分隔 非变异

concat 连接 非变异

cookies和sessionStorage 和 localStorage的区别

共同点：都保存在浏览器端，并且同源的

cookie:存储在用户本地终端上的数据。有时也用cookies，指某些网站为了辨别用户身份，进行session跟踪而存储在本地终端上的数据，通常经过加密。一般应用最典型的案列就是判断注册用户是否已经登过该网站。

localStorage - 没有时间限制的数据存储,第二天、第二周或下一年之后，数据依然可用

sessionStorage - 针对一个 session 的数据存储,当用户关闭浏览器窗口后，数据会被删除。

箭头函数与普通函数的区别

1.箭头函数是匿名函数，不能作为构造函数，不能使用new

2.箭头函数不绑定arguments，取而代之用rest参数...解决

3.箭头函数不绑定this，会捕获其所在的上下文的this值，作为自己的this值

4.箭头函数通过 call() 或 apply() 方法调用一个函数时，只传入了一个参数，对 this 并没有影响。

5.箭头函数没有原型属性

6.箭头函数不能当做Generator函数,不能使用yield关键字

总结

箭头函数的 this 永远指向其上下文的 this ，任何方法都改变不了其指向，如 call() , bind() , apply()

普通函数的this指向调用它的那个对象

常用的HTTP方法有哪些

GET：用于请求访问已经被URL（统一资源标识符）识别的资源，可以通过URL传参给服务器。

POST：用于传输信息给服务器，主要功能与Get方法类似，但一般推荐POST方式。

PUT：传输文件，报文主体包含文件内容，保存到对应URL位置。

HEAD：获取报文首部，与GET方法类似，只是不返回报文主体，一般用于验证URL是否有效。

DELET：删除文件，与PUT方法相反，删除对应URL位置的文件。OPTIONS：查询相应URL支持的HTTP方法。

三次握手

http请求过程 三次握手

第一次握手：客户端向服务器发送SYN报文，并发送客户端初始序列号Seq=X；等待服务器确认，

　　第二次握手：服务器接收客户端的SYN报文，然后向客户端返回一个包SYN+ACK响应报文，并发送初始序列号Seq=Y

　　第三次握手：客户端接受SYN+ACK报文，并向服务器发送一个ACK确认报文，至此连接建立

　　【建立连接的最重要目是让连接的双方交换初始序号(ISN, Initial Sequence Number)，所以再响应的ACK报文中会包含序列号递增序列】

2.四次挥手

（1）第一次挥手

主机1（可以使客户端，也可以是服务器端），设置Sequence Number，向主机2

发送一个FIN报文段，主机1进入FIN\_WAIT\_1状态，这表示主机1没有数据要发送

给主机2了

（2）第二次挥手

主机2收到了主机1发送的FIN报文段，向主机1回一个ACK报文段，Acknowledgment

Number为Sequence Number加1；主机1进入FIN\_WAIT\_2状态，主机2告诉主机1，

我同意你的关闭请求

（3）第三次分手

主机2向2主机1发送FIN报文段，请求关闭连接，同时主机2进入LAST\_ACK状态

（4）第四次分手

主机1知道主机2接收完了，断开连接

正向代理和反向代理

1.正向代理 就是介于页面和服务器端的一个方法

2.反向代理 就是服务器端的一个方法

具体看http://baijiahao.baidu.com/s?id=1600687025749463237&wfr=spider&for=pc

地狱回调怎么解决

1.通过promise解决回调地狱

2.通过generator函数解决

3.通过async和await解决

简述下promise？

1.promise是es6新增的API 是一个异步对象 用来解决异步操作

2.promise有三个状态，pending，resolved，rejected

3.进入其中某个状态，就不能再更改

4.有两个方法， .then方法 成功回调 .catch方法 错误回调

5.多条数据回来的时候，可以通过 . race方法，他有三个参数，每个参数都是一个函数，

谁最先回来，以谁为准

.all方法 有三个参数，每个参数都是函数，谁最后回来 以谁为准

简述下generator函数？

1.generator是一个状态机，每条yield语句都是一个状态

2.用同步的方法去写异步操纵

3.generator会返回一个遍历器对象，遍历器对象都有一个next方法，可以对yield语句进行

遍历，每次调用一次next方法，都会返回一个对象{value:值，done：false}

直到yield语句执行完毕或碰上 return 会返回{value：undefined，done：true}

web性能时间:从用户输入网址回车到网页呈现到用户面前所花时间。

1.域名解析时间---中国万网

2.网页下载时间

--尽可能降低页面及资源的体积

--减少http请求次数

3.页面加载时间

--优化css

--优化js

--减少DOM刷新次数---采用虚拟DOM

性能优化

1.从DOM结构和标签上来优化

（1）使用语义化的标签，代码清晰简洁；

（2）减少Dom节点，增加渲染速度；

（3）使用W3C标准书写闭合小写的标签；

（4）给图片和table指定宽高，避免缩放；

（5）防止src和href值为空，当为空时，浏览器会把当前页面当做属性值重新加载；

（6）css在头部位置，js在body底部位置；

（7）避免频繁的操作DOM节点，使用innerHTML代替

2.从CSS样式上来优化

（1）使用link加载样式而不是@import（是css2提供的一种方式，不兼容，只能加载css，而且页面所有组件被加载完后才会被加载，完成前会导致‘闪烁’，link属于XHTML标签，没有兼容问题）；

（2）避免使用css表达式；

（3）避免使用css filter滤镜；

（4）使用css 缩写 如#fff，减少代码量；

（5）删除重复的css，css简化；

（6）使用CSS Sprite把同类图片合成一张，减少图片http请求；

（7）减少css查询层级，如.header .log 要好于.header .top .log；

（8）减少css查询范围，如header>div获取直系子元素要好于heade div；

雪碧图的优缺点

答： 优点:

减少了Http请求数量，提高了加载性能

某些情况下可以减少图片的大小（字节总和）

缺点

高清屏会失真

雪碧图不方便变化，提高了网页开发和维护成本

HTTP协议 状态码

HTTPS【现在常用，较流行】 与 HTTP 的区别？

HTTPS多了SSL/TLS保密协议

特点：无状态协议【最重要，必说】减少带宽 防止人数访问过多 连接一直不断 炸了

多次HTTP请求 基于TCP协议 三次握手 四次挥手

基于TCP协议

HTTP报文

请求报文

响应报文

请求报文抛出信息：

开始行：【重要 要知道这里包含些啥】请求地址 请求方法 具体信息

首部行：

实体行：

常用请求方法：

GET【一般用来获取数据】&POST区别

常用状态码：

200 成功

204 请求成功 但服务器端没给我返回具体数据

301 想访问的地址，已经永久改成了另一个地址 永久性定向

302 与301很像 临时性的定向 比如临时做个维护，临时用一下，暂时修改

304 请求资源未修改 用当前浏览器缓存的信息 不用在加载

4-客户端问题

400 客户端请求时出现语法错误

403 服务器端权限有限制 禁止访问 【地址存在 但是拒绝访问】

404 资源找不到 【地址不存在】

5-服务器端问题

500 服务器端 出现错误

503 服务器端超负载 比如访问人数过多

504 网关超时

408&504区别

客户端向服务器请求超时

服务器向客户端请求超时

flex单词

水平的主轴 ： main axis

垂直的交叉轴 ： cross axis

主轴的开始位置 ： main start

结束位置 ： main end

交叉轴的开始位置 ： cross start

结束位置 ： cross end

单个项目占据的主轴空间 ： main size

占据的交叉轴空间 : cross size

flex - direction : row 默认值 横向排列

flex - direction : row - reverse 反向排列

flex - direction : column 垂直排列

flex - direction : column - reverse 反向垂直排列

flex - wrap : nowrap flex容器为单行

flex - wrap : wrap flex容器为多行

flex - wrap : wrap - reverse flex容器为多行 并且反向

flex - flow : flex - direction flex - wrap 合并写法

flex - direction : 定义弹性盒子元素的排列方向

flex - wrap : 控制flex容器是单行或者是多行

justify - content 子元素横向排版

justify - content : flex -start 弹性盒子元素将向起始位置对齐

justify - content : flex - end 弹性盒子元素将向行结束位置对齐

justify - content : center 弹性盒子将向行中间位置对齐

justify - content : space - between 弹性盒子元素平均分布在行内

justify - content : space - around 两端保留的距离是子元素于子元素间距大小的一半

align - self 子元素纵向排版

align - self : flex -start 元素位于容器开头

align - self : flex -end 元素位于容器的结尾

align - self : auto 默认值 元素继承了 父容器的 align - items 属性 如果没有父容器，则为 " stretch "

align - self : baseline 元素位于容器的基线上

align - self :stretch 元素被拉伸以适应容器

align - items 子元素纵向排版

align -item : auto 默认 stretch

align - items : flex - start 纵轴起始位置的边界紧靠该行侧纵轴起始边界

align - items : flex -end 纵轴起始位置的边界紧靠住该行的侧纵轴结束边界

align - items : center 元素在该行的纵轴上居中放置

align - items : baseline 基线对齐 / 与 flex - start 等效

align - items : stretch 元素的边距盒的尺寸尽可能接近所在行的尺寸，但同时回遵照 ' min/max-width/height ' 属性的限制

align - content 子元素纵向排版

align - content : stretch 默认值 元素的高度填充整个容器

align - content : flex - start 从头开始， 一个挨着一个横向排列

align - content : flex - end 从尾部开始，一个挨着一个横向排列

align - content : center 从中间开始 一个挨着一个横向排列

align - content : space - between 各行在弹性盒容器内平均分布

align - content : space - around 各行在弹性容器盒容器内平均分布，两端保留子元素与子元素之间间距大小的一半

flex - grow 默认值为 0 意思：该元素b不索取父元素的剩余空间 ，如果值大于 0 表示索取 。 值越大，索取的越厉害

flex - basis 该属性可以设置该元素的宽度 ， 如果元素上同时设置了width 和 flex - basis ，那么 flex - basis 会覆盖 width的值

flex - shrink 默认值为 1 该属性来设置 ， 当 子元素宽度和超过父元素宽度时侯 ， 子元素如何缩小自己的宽度的

当父元素宽度小于子元素宽度和时候，子元素宽度就会减小。

值越大，减小的越厉害 值为 0 表示不减小

order : 用来对 flex item 进行排序