前端笔记js篇

笔记本:	我的第一个笔记本		
创建时间:	2020/8/7 15:44	**更新时间:**	2020/9/16 14:05
作者:	1510363735@qq.com		
URL:	https://juejin.im/post/6844903776512393224#heading-25		

1、原型/构造函数/实例
1.1、变量提升，在执行环境的创建阶段，解析器会找到需要提升的函数和变量提前在内存开辟空间，函数直接放进内存，变量会声明并且赋值为undefined,函数声明后边会覆盖前边，并且声明优先于变量
2、原型链
3、执行上下文
4、作用域与作用域链
5、闭包
6、代码复用（继承、apply/call、函数封装、复制extend、混入mixin）
7、script引入方式
8、对象拷贝
9、类型转换与判断
[]==[]
[]==![]
10、this用法
11、es6/es7（单独笔记）
12、函数式编程 纯函数 柯里化，偏函数，组合，管道
纯函数 输入相同的函数，返回值一样，不引用外部的变量，只使用自己参数
柯里化 将多参函数分解成n个一个参数的函数
偏函数 将多参函数分解成n-x个一个参数的函数 ，比如某些参数是固定的可以先传过去
组合 将两个函数组合成一个函数，左边的先执行，结果传给右边
管道 和组合相反
13、数组函数及操作
14、防抖节流
15、原理题instanceof new object.create等
16、引用类型和基本类型的存储
17、浮点数运算，为什么，怎么解决

前端笔记webpack篇

笔记本： 我的第一个笔记本
创建时间： 2020/8/9 17:51 **更新时间：** 2020/9/15 16:14
作者： 1510363735@qq.com
URL： https://juejin.im/post/6844903682455109640#heading-52

1、webpack性能优化

- 对于 Webpack4，打包项目使用 production 模式，这样会自动开启代码压缩
- 使用 ES6 模块来开启 tree shaking，这个技术可以移除没有使用的代码
- 优化图片，对于小图可以使用 base64 的方式写入文件中
- 按照路由或组件拆分代码（chunk），实现按需加载
- 给打包出来的文件名添加哈希，实现浏览器缓存文件
- 用splitchunkplugin进行公共模块代码抽取，缓存不常用文件

2、webpack打包优化
（1）减少文件搜索范围
（2）Happypack 并发调用
（3）externals进行外部扩展
（4）用dllplugin将一些基本不会改变的代码打包成静态资源，避免重复编译
（5）利用wepack.cache, babel缓存

3、babel原理
（1）解析： babelon将代码字符串解析成抽象语法树（AST）
（2）转换： babel-traverse对抽象语法树（AST）遍历转换成新的AST
（3）生成：根据转换后的抽象语法树（AST）在生成代码字符串

4、webpack几种hash的实现原理
（1）hash是跟整个项目的构建相关，只要项目里有文件更改，整个项目构建的hash值都会更改，并且全部文件都共用相同的hash值。(粒度整个项目)
（2）chunkhash是根据不同的入口进行依赖文件解析，构建对应的chunk(模块)，生成对应的hash值。只有被修改的chunk(模块)在重新构建之后才会生成新的hash值，不会影响其它的chunk。(粒度entry的每个入口文件)
（3）contenthash是跟每个生成的文件有关，每个文件都有一个唯一的hash值。当要构建的文件内容发生改变时，就会生成新的hash值，且该文件的改变并不会影响和它同一个模块下的其它文件。(粒度每个文件的内容)

5、常用的loader

- 样式：style-loader、css-loader、less-loader、sass-loader等
- 文件：raw-loader、file-loader 、url-loader等
- 编译：babel-loader、coffee- 、ts-loader等
- 校验测试：mocha-loader、jshint-loader 、eslint-loader等

6、常用的plugin
（1）首先webpack内置UglifyJsPlugin，压缩和混淆代码。
（3）ProvidePlugin：自动加载模块，代替require和import
（4）html-webpack-plugin可以根据模板自动生成html代码，并自动引用css和js文件
(5) clean-webpack-plugin 每次打包会把dist文件夹先清除
(6) happypack：通过多进程模型，来加速代码构建
(7) compression-webpack-plugin 生产环境可采用gzip压缩JS和CSS
(8) HotModuleReplacementPlugin 热更新
(9) definePlugin 定义一些全局变量，比如接口地址前缀

7、实现一个loader(loader返回值必须是一段js代码)

```
// markdown-loader.js
const marked = require('marked');
module.exports=source=>{
  const html = marked(source);
  const code = `export default ${JSON.stringify(html)}`;
  return code;
}
```

8、实现一个plugin

```
// remove-comments-plugin.js
// 插件必须的几种条件
1、 首先有一个apply方法 在插件被webpack调用的时候触发，参数是compiler（webpack实例）
2、 然后指定一个 webpack 钩子事件hooks，可能的值有 beforeRun、run、beforeCompile、
compile、make、seal、emit、done,这些钩子有个tap方法，异步的话用tapAsync或
tapPromise,tap方法有两个参数,插件名称和回调
3、 这个回调有个参数compilation对象，包含了当前的模块资源、编译生成资源、变化的文件等
4、 对 webpack 实例内部做一些操作处理
5、在功能流程完成后可以调用 webpack 提供的回调函数

class RemoveCommentsPlugin {
  apply(compiler){
  compiler.hocks.emit.tap('removeCommentsPlugin',compilation=>{
     for(const name in compilation.assets){
      if(name.endsWith('.js')){
          const contents = compilation.assets[name].source();
          const noComments = contents.replce(/\/\*{2,}\/\s?/g,'');
          compilation.assets[name]={
            source:()=>noComments,
            size: ()=>noComments.length,
          }
      }
    }
})
}
}
```

9、打包原则（多入口打包,动态导入）
（1）选择合适的打包粒度，生成的单文件大小不要超过500KB
（2）充分利用浏览器的并发请求，同时保证并发数不超过6
（3）尽可能让浏览器命中304，频繁改动的业务代码不要与公共代码打包
（4）避免加载太多用不到的代码，层级较深的页面进行异步加载

10、webpack打包原理
（1）载入webpack核心模块，创建compiler对象
（2）从入口文件开始，解析模块依赖，形成依赖关系树
（3）递归依赖树，将每个模块交给对应的loader处理
（4）编译模块，生成抽象语法树AST
（5）循环遍历AST树，合并结果将打包结果输出到dist。

11、配置tree-shaking

```
module.export={
// 最优选择
  optimization:{
  // 模块只导出使用的成员
    usedExports: true,
  // 尽可能连接每个模块到一个函数中
concatenateModules: true,
  // 尽可能压缩代码
```

```
minimize：true,
// 没用到的成员分为两类，一个是导出了但是没用到由usedExports去除，一个是没导出没用到
sideEffect负责这种，
// sideEffect还可以在package.json文件配置
sideEffect：true,
// 多入口打包提取公共模块
splitChunks:{
  chunk: 'all',
}
}
}
```

12、webpack魔法注释，import(/*webpackChunkName:'post'*/'./post'),给分包chunk起名字

13、webpack有哪些配置

14、splitchunksplugin用法

按照路由对代码进行拆分，实现按需加载

（1）多入口打包，一个路口一个路由

（2）在output中设置chunkfilename，然后在路由中使用魔术注释配置chunk名称

（3）通过插件分包

15、静态文件直接打包

前端笔记Vue篇

笔记本： 我的第一个笔记本
创建时间： 2020/8/8 19:56 **更新时间：** 2020/9/10 2:15
作者： 1510363735@qq.com

1、nextTick
2、生命周期
3、双向绑定原理
4、virtual dom原理（diff算法）
5、vue3比vue2 更快的原因，使用proxy，重写vdom diff算法加入静态节点，没有this,更好的tree-shaking支持，更好的逻辑复用
6、vue-router（钩子，传参，鉴权， 转换）
7、vuex
8、mvvm的理解
m就是model层
v就是视图层
vm是连接model和视图的桥梁，负责数据和逻辑的处理，把model层的变化展示到视图，把视图变化绑定到model
9、解析器如何解析模板
10、实现简单vdom

前端笔记es6篇

笔记本： 我的第一个笔记本
创建时间： 2020/8/12 14:59 **更新时间：** 2020/9/10 2:14
作者： 1510363735@qq.com

1、let和const
（1）都存在块级作用域，仅在作用域生效
（2）三个特性不存在变量提升，暂时性死区，不可重复声明
（3）const是只读的，不可修改，本质指向的地址不可修改
2、变量解构赋值
3、字符串模板
4、函数参数默认值，箭头函数
（1）外形不同
（2）箭头函数都是匿名函数
（3）箭头函数不能用于构造函数
（4）this指向不同
（5）箭头函数没有arguments
5、数组扩展运算符，Array.from()
6、symbol
7、map和set
8、proxy
用于修改默认行为，相当于给对象加了一层拦截
9、promise
promise.all原理，执行所有promise并保存结果，while 结果长度等于arr长度resolve
promise.race原理，保存所有promise.then,如果typeo等于function就说明执行完毕，resolve结果否则resolve当前promise
10、async/await 假如想并行，可以给await 赋值，es5实现
11、class

前端移动端

笔记本： 我的第一个笔记本
创建时间： 2020/9/10 1:57 **更新时间：** 2020/9/10 2:08
作者： 1510363735@qq.com
URL： https://www.cnblogs.com/zhouwenfan-home/p/10469573.html

1、 移动端高清方案如何解决
2、 移动端300ms延时的原因？如何处理？
3、 半像素线
4、rem px em 其他布局方案
5、 移动端优化方式？离线包是如何实现的？

前端笔记网络篇

笔记本： 我的第一个笔记本
创建时间： 2020/8/7 11:17 **更新时间：** 2020/9/10 1:54
作者： 1510363735@qq.com
URL： https://baike.baidu.com/item/XSS%E6%94%BB%E5%87%BB/954065?fr=aladdin

1、http1.0 http1.1 http2.0 https 非全站的https不一定安全
2、tcp三次握手 四次挥手 tcp拥塞处理
3、网络缓存原理
4、状态码
5、跨域 4种
6、安全 3种
xss（跨站脚本攻击）
（1）设置httpOnly不能阻止注入攻击，用来阻止窃取cookie
（2）设置白名单csp，只执行特定来源的代码
（3）对输入进行严格检查
csrf（跨站请求伪造）
（1）同源检测origin referer
（2）设置token
（3）Set-Cookie 响应头新增SameSite
（4）验证码
中间人劫持
（1）服务端添加x-frame-options响应头，阻止frame渲染
（2）top.location.host===self.location.host

7、get/post
8、websoket和soket
9、ssr
10、udp和tcp的区别
（1）tcp是面向连接的，udp不需要连接
（2）tcp传输数据可靠的，udp尽最大努力交付，不保证数据可靠性
（3）tcp只能1对1，udp可以一对多
（4）tcp首部较大20字节udp只有8字节
（5）tcp面向字节流，udp面向报文
11、常见的网络劫持，dns劫持 http劫持
12、seo怎么做
13、http请求方式及主要用来干什么
14、 http报文头部有哪些字段？有什么意义 ？

前端笔记浏览器篇

笔记本： 我的第一个笔记本
创建时间： 2020/8/7 11:31 **更新时间：** 2020/8/20 20:54
作者： 1510363735@qq.com

1、浏览器架构
2、浏览器事件循环
注意await，阻塞代码，并且先执行微任务
3、浏览器存储
4、垃圾回收机制
标记清除,标记所以变量，清除环境中和环境引用变量的标记，再次标记就是准备删除，最后把所有标记的变量清除
引用计数
v8分代清除，新生代，老生代（标记清除，碎片压缩算法，注意标记清除和js的不同，这里只标记存活的对象，清除未标记的）
5、重绘与回流
6、标签页通信
7、内存泄漏
8、web worker
9、server worker
10、window对象

前端笔记css篇

笔记本： 我的第一个笔记本
创建时间： 2020/8/7 16:21 **更新时间：** 2020/9/20 14:52
作者： 1510363735@qq.com
URL： https://zhuanlan.zhihu.com/p/78230297

1、盒模型
2、BFC
3、层叠上下文
指页面上元素图层的显示顺序
zindex正>zindex0>行内元素>浮动元素>块级元素>zindex负>background
4、居中布局
5、去除浮动
6、link和@import的区别
7、选择器优先级
8、css动画
9、css3新特性
`flex` `border-radius` `border-image` `background-image` `linear-gradient()` `transition` `transform` `animation`
10、css画三角形平行四边形
11、transform和position
`transform` 没有触发 repaint， `transform` 动画由GPU控制，支持硬件加速，还包括opcity和filter

前端项目相关

笔记本： 我的第一个笔记本
创建时间： 2020/9/10 1:50 **更新时间：** 2020/9/20 23:25
作者： 1510363735@qq.com
URL： https://www.nowcoder.com/discuss/392405?type=2&order=0&pos=23&page=...

1、设计模式
2、静态文件的浏览器缓存如何实现
3、并发请求
4、axios interceptors原理