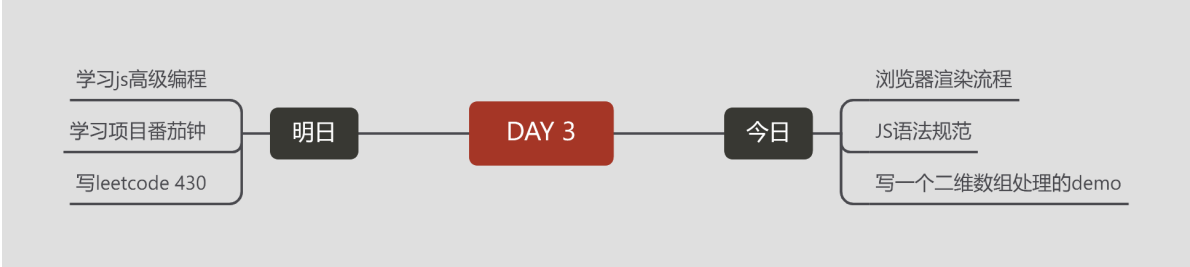
# 大纲




# 学习心得

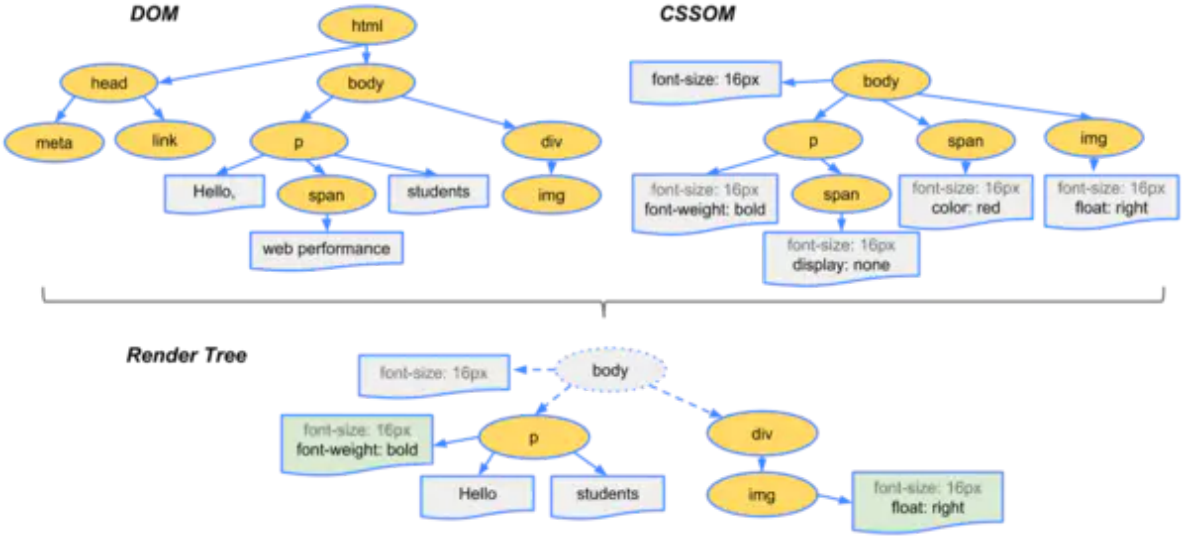
## 浏览器渲染机制

### 作用

- 了解浏览器如何进行加载，我们可以在引用外部样式文件，外部S时，将它们放到合适的位置，是浏览器以最快的速度，将文件加载完毕。
- 了解浏览器如何进行解析，我们可以在构建DOM结构，组织CSS选择器的时候，选择最优的写法，提高浏览器的解析速率。
- 了解浏览器如何进行渲染，明白渲染的过程，我们在设置元素属性，编写JS文件时，可以减少“重绘”，“重新布局”的消耗。

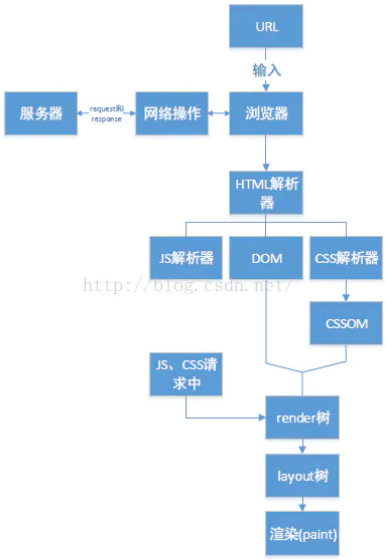
### 概念

1. DOM：浏览器将HTML解析成树形的数据结构
2. CSSOM：浏览器将CSS解析成树形的数据结构
3. Render Tree：DOM和CSSOM合并后生成Render Tree
4. Layout: 计算出Render Tree每个节点的具体位置
5. Painting：通过显卡，将Layout后的节点内容分别呈现到屏幕上。



## 流程

1. 用户输入URL，浏览器向服务器请求URL对应的资源。
2. 接受到服务器的响应内容后，浏览器的HTML解析器，将HTML文件解析成DOM树。
3. 将CSS解析成CSSOM树。
4. 根据DOM树和CSSOM树，来构建Render Tree（渲染树）。
5. Layout，即计算出每个节点在屏幕中的位置。
6. painting,按照算出来的规则，通过显卡，把内容画到屏幕上。

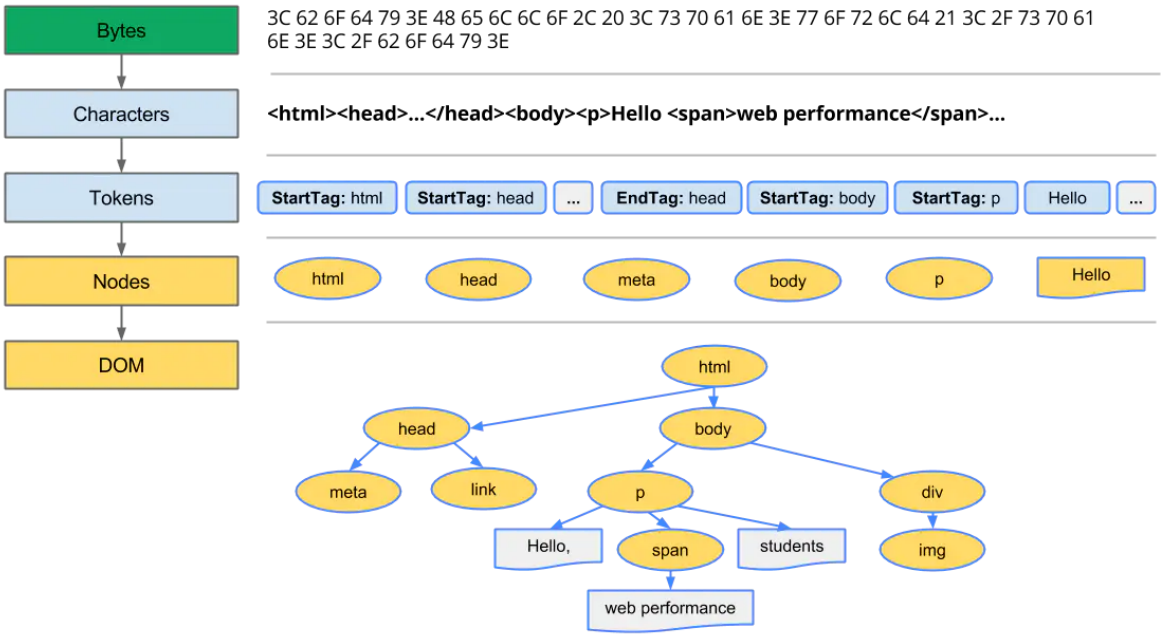



## 注意

1. 浏览器是自上而下的加载，并在加载过程中进行解析和渲染。。
2. 加载说的就是获取资源文件的过程，如果在加载过程中遇到外部CSS文件和图片，浏览器会另外发送一个请求，去获取CSS文件和相应的图片，这个请求是异步的，并不会影响HTML文件的加载。
3. 但是如果遇到Javascript文件，HTML文件会挂起渲染的进程，等待JavaScript文件加载完毕后，再继续进行渲染。因为JavaScript可能会修改DOM，导致后续HTML资源白白加载，所以HTML必须等待JavaScript文件加载完毕后，再继续渲染，所以JavaScript文件在写在底部body标签前的原因。

## DOM和CSSOM的具体构建流程

DOM 和 CSSOM 都是以" Bytes → characters → tokens → nodes → object model. " 这样的方式生成最终的数据。




# JS代码规范

## 变量命名

1. 变量名不应以短巧为荣

   把变量的名字起得真实有意义，不要搞一些很短很通用的名字。

2. 变量名不要使用计算机术语

   变量名应直指问题领域，来源于现实世界，而不是计算机世界

3. 变量名的对仗要明确

4. 警惕临时变量

   尽量少用temp类的变量

5. bool变量

   布尔变量不用以is/do之类的开头。另外变量名不要使用否定的名词，如notOk，notReady，因为否定的词取反的时候就会比较奇怪，要使用肯定的布尔变量名。

6. 变量名使用正确的语法

   不要使用中文拼音，如果是复数的话加s，或者加上List，而过去式的加上ed，如果正在进行的加上ing。

7. 声明变量时要赋值

## 声明变量时要赋值

有利于JS解释器提前做一些优化处理，不用等到使用的时候才知道这些变量是什么类型的。

## 函数的返回值类型要确定

## 不要给变量赋值undefined

undefined表示一个变量未定义，你定义了一个变量又说它未定义本身就很奇怪。这可能会造成的问题是使用上的歧义，因为我们经常使用undefined来判断变量有没有定义。

如果要赋值应该要赋空值，如果你的数字都是非负数，那么可以把初始值置为-1，实在不行就置成NaN。

## 排版规范

逗号后面带个空格

) { 中间带个空格

双目运算符左右两边都带个空格

一行太长要换行，如V8的源码里面一行最长是70个字符，超过就换行

## 使用===代替==

使用==会有一些比较奇怪的结果

## 减少魔数

对一些比较重要的常量起一个名字

## 不要让代码暴露在全局作用域下运行

一个原因是在全局作用域下，变量的查找时间会更长，第二个原因是污染全局作用域，有时候会造成一些意想不到的结果

## let/var/const的使用

使用let有一些好处

1. 避免变量重复定义
2. for循环的变量作用域是独立的
3. 而const适合于给常量起个名字

## 简洁代码

1. 使用三目运算代替简单的if-else
2. 使用箭头函数取代简单的函数
3. 注意避免执行过长时间的JS代码,大批量的DOM操作，只要一次有几百上千的级别就容易造成页面卡顿。使用setTimeout的方式分段处理数据，甚至使用多线程。

## 多写注释

1. 文件顶部的注释，包括描述、作者、更新、

```
/*
 * @file listing-detail.js
 * @description 房源详情页的JS主文件，处理轮播、房贷计算器、约看房等逻辑
 * @author yincheng.li
 * @update (yincheng.li 2017/8/19)
 */。
```

2. 函数的注释
3. @author @return都是注释标签,,常用的注释标签有

```
/*
@class 表示一个类
@constructor 构造函数
@deprecated 被弃用
@global 全局的变量
@namespace 具有命名空间作用的object，如$.fn.remove，$.fn.append，$和fn就是一个
namespace，而fn是$的子命名空间
@this 这里的this指向哪里
@throws 在这个函数里面可能会抛出什么异常
@version 当前版本
*/
```

4. 变量定义和代码的注释

   对一些比较重要的变量加注释，标明它是什么用途，以及对一些核心代码逻辑加上注释，或者比较复杂的业务逻辑

5. 代码不要嵌套太深

6. 对于常用的属性进行缓存

7. 尽量不要在JS里面写CSS

   如果以后颜色改了，那么你需要改两个地方，一个是CSS里设置，另一个是JS里面设置，而JS写的样式特别容易被忽略，查起来也不好定位。好的做法应该是通过添加删除类的方法

```
//变成选中态
$menuItem.addClass("selected");
//变成非选中态
$menuItem.removeClass("selected")
```

8. 在必要的地方添加非空判断

9. 不要用for in循环数组

   如果有人给数组原型添加了一个函数，循环里的i将会有4个值：0, 1, 2, "add"，这样就导致你的遍历出现问题，所以数组遍历应该使用length属性或者数组的forEach/map方法。

10. 分号规范

    提倡要每个句子后面都要加上分号，这样不容易出错。

11. 使用location跳转需要先转义

    如果不确定内容的东西需要先encode一下

```
let searchContent = encodeURIComponent(form.search.value.trim());
```

12. 点击跳转尽量不要使用onclick跳转

    如果是一个跳转应该用a标签,同时把a标签变成块级。就算你不用做SEO，也应当尽量使用这种方式，因为用这种方式比较自然，还可以控制是否要新开页，如果在移动端也不用考虑click事件是否有延迟的问题。

13. 不要直接使用localStorage

    Safari的隐身模式下本地存储会被禁用，如果你尝试往localStorage写数据的话，会报超出使用限制的错误，要做个兼容

14. 使用简便的转换

    1. 把字符串转整型可以使用+号
    2. 把小数去掉尾数转成整型，可以使用 >> 0

    3. 转成boolean值用!!
15. 注意返回false的变量

    判断一个数组有没有元素可以这么写：

    ```
    if (array.length) {}
    ```

    而不用写成：

    ```
    if (array.length !== 0) {}
    ```

    判断一个字符串是不是空可以写成：

    ```
    if (str) {}
    ```

    但是判断一个变量有没有定义还是要写成：

    ```
    if (typeof foo !== “undefined”) {}
    ```

16. 使用Object.assgin简化数据赋值
17. 调试完去掉无关的console
18. 注意this的指向
19. 使用正则表达式做字符串处理
20. 保持复用模块的观念

    当你一个函数要写得很长的时候，例如两、三百行，这个时候你考虑把这个大函数拆成几个小函数，让主函数得清晰简洁，每个小函数的功能单一独立.

21. 注意label事件会触发两次

    如果label里面有input，监听label的事件会触发两次

# JS的调试方法

运行项目，在Chrome中点击检查sourse -> 添加断点 -> 刷新页面 ->调试

或者可以在代码中要调试的行下加debugger。

# 明日计划

1. 写完贪心算法第680题
2. 看js高级编程