前端基础笔记

曹思远

2021 年 1 月 1 日

目录

摘要

前端基础笔记

1 软件安装

1.1 必备软件列表

1. vscode
2. cmder
3. chrome
4. clash
5. NodeJS, Yarn

1.1.1 必备软件配置

任何软件都需要配置

vscode

1. 必备插件
   - Chinese (Simplified) Language Pack for Visual Studio Code
   - Code Spell Checker
   - Git Easy
   - Latex Workshop
   - Markdown All in One
   - Prettier - Code formatter
2. 环境和快捷键配置
   - 具体看我知乎保存的 setting.json 和 keybinding.json
3. 快捷键
   - ctrl + p - 找文件
   - ctrl + shift + p 或 F1 - 输命令
   - alt + 单击 - 多位置输入

cmder

1. 配置内容较多，直接看我保存的配置文件。
2. 将 cmder 加入右键菜单，加入环境变量。

chrome

1. 可选插件
   - Proxy SwitchOmega
   - uBlock
2. 高级用户配置
   - 在开发者工具里面按 ESC 可以新建控制台
   - Sources 面板可以保存代码片段 (Snippets)
   - Network 关掉 show overview，filter 可以搜素，右击勾选 Method
   - Network 可模拟慢网速/断网
   - Preserve log 不会清空当前请求数据
   - Disable cache 清除缓存
3. chrome 常用快捷键
   - 鼠标中键单击 - 打开或关闭
   - ctrl + T - 新开标签
   - ctrl + shift + T - 撤销关闭
   - ctrl + 点击 - 在新标签打开
   - ctrl + W - 关闭当前标签
   - ctrl + Reload 或者 F5 - 刷新
   - ctrl + Location - 输入网址
   - ctrl + shift + Inspector 或 F12 - 打开开发者工具
   - alt + 左右 - 前进后退
   - alt + 回车 - 在新标签打开
   - shift + 回车 - 在新窗口打开
   - ctrl + shift + delete - 删除历史浏览数据

windows

1. 关掉任务栏无用标签，卸载无用软件。(如搜索框，任务视图，微软小娜 Cortana，开始菜单里的各种贴图)
2. 可安装 TranslucentTB 使任务栏透明
3. 理解用户目录，即 C:\Users\Jony，分别右击用户目录里的下载和文件，属性–> 位置–> 移动到 E 盘。
4. 显示文件后缀，打开查看–> 选项–> 查看，去掉隐藏已知文件扩展名，勾选显示已知文件，文件夹和驱动器。
5. 记住组合键
   - Win 组合键
     (a) win + Desktop - 展示桌面
     (b) win + 方向键 - 移动窗口
     (c) alt + tab - 切换窗口
     (d) win + tab - 不怎么常用的切换窗口

(e) win + ctrl + 方向键 - 切换桌面

- Ctrl 组合键

(a) ctrl + All/ctrl + Copy/ctrl + V/ctrl + Z/ctrl + Y

(b) ctrl + Reload/F5

(c) ctrl + P - 打印

NodeJS, Yarn

1. 开发必装的东西

- nrm
- tldr

2 Git 入门

2.1 命令行入门

2.2 本地仓库

2.3 远程仓库

3 HTML

3.1 概览

3.2 标签

3.3 重难点

3.4 实践和手机调试

4 CSS

4.1 基础

4.1.1 语法

4.1.2 Border 调试法

4.1.3 文档流

1. 文档流的基本概念

- 流动方向
  - inline 元素从左到右，到达最右边才会换行
  - block 元素从上到下，每一个都另起一行
  - inline-block 也是从左到右
- 宽度
  - inline 宽度为内部 inline 元素的和，不能用 width 指定
  - block 默认自动计算宽度，可用 width 指定
  - inline-block 结合前两者特点，可用 width

- 高度
  - inline 高度由 line-height 间接确定，跟 height 无关
  - block 高度由文档流元素决定，可以设 height
  - inline-block 跟 block 类似，可以设置 height

2. overflow 溢出
   - 当内容大于容器
     - 等内容的宽度或高度大于容器，会溢出
     - 可用 overflow 来设置是否显示滚动条
     - auto 是灵活设置
     - scroll 是永远显示
     - hidden 是直接隐藏溢出部分
     - visible 是直接显示溢出部分
     - overflow 可以分为 overflow-x 和 overflow-y
3. 脱离文档流
   - 有些元素可不在文档流
     - 原因是 block 高度由内部文档流元素决定，可以设 height
   - 以下元素脱离文档流
     - float
     - position:absolute/fixed
   - 不用以上属性就不脱离文档流

4.1.4 盒模型

1. 两种
   - content-box 内容盒 - 内容就是盒子的边界
   - border-box 边框盒 - 边框才是盒子的边界
2. 公式
   - content-box width = 内容宽度
   - border-box width = 内容宽度 + padding + border
3. 哪个好用？
   - border-box 好用
   - 因为可以同时指定 padding, width, border

4.1.5 margin 合并

1. 哪些情况会合并
   - 父子 margin 合并
   - 兄弟 margin 合并
2. 如何阻止合并

- 父子合并用 padding/border 挡住
- 父子合并用 overflow:hidden 挡住
- 父子合并用 display:flex
- 兄弟合并是符合预期的
- 兄弟合并可以用 inline-block 消除
- css 属性逐年增多，每年都有新的，死记就完事了

4.1.6 基本单位

1. 长度单位
    - px 像素
    - em 相对于自身 font-size 的倍数
    - 百分数
    - 整数
    - rem
    - vw 和 vh
2. 颜色
    - 十六进制 #FF6600 或者 #F60
    - RGBA 颜色 rgb(0,0,0) 或者 rgba(0,0,0,1)
    - hsl 颜色 hsl(360,100%, 100%)

4.1.7 练手项目

彩虹 demo

4.2 布局

4.2.1 布局分类

1. 两种
    - 固定宽度布局，一般宽度为 960/1000/1024px
    - 不固定宽度布局，主要靠文档流的原来布局
2. 回顾
    - 文档流本来就是自适应的，不需要加额外的样式
3. 响应式布局
    - PC 上固定宽度，手机上不固定宽度
    - 也就是一种混合布局

4.2.2 两种布局思路

1. 从大到小
    - 先定下大局
    - 然后完善每个部分的小布局
2. 从小到大
    - 先完成小布局
    - 然后组合成大布局
3. 两种均可
    - 新人推荐第二种，因为小的简单
    - 老手一般用第一种，因为熟练有大局观

4.2.3 float 布局

一图流 (图片以后贴出)

1. float 布局
    - 子元素加上 float:left 和 width
    - 在父元素上加.clearfix
2. float 布局经验
    - 留一些空间或者最后一个不设 width
    - 不需要做响应式，因为手机上没有 IE，而这个布局是专门为 IE 准备的
    - 解决 IE6/7 存在的双倍 margin bug 如下
    - 一是将错就错，针对 IE6/7 把 margin 减半
    - 二是神来一笔，再加一个 display:inline-block

float 布局实践

1. 不同布局
    - 用 float 做两栏布局 (如顶部条)
    - 用 float 做三栏布局 (如内容区)
    - 用 float 做四栏布局 (如导航)
    - 用 float 做平均布局 (如产品展示区)
    -
2. 实践经验
    - 加上头尾，即可满足所有 PC 页面需求
    - 手机页面傻子采用 float
    - float 要程序员自己计算宽度，不灵活
    - float 用来应付 IE 足矣

4.2.4 Flex 布局

1. 重点
   - display：flex
   - flex-direction：row/column
   - flex-wrap:wrap
   - just-content:center/space-between
   - align-item:center
2. 颜色
   - 十六进制 #FF6600 或者 #F60
   - RGBA 颜色 rgb(0,0,0) 或者 rgba(0,0,0,1)
   - hsl 颜色 hsl(360,100%, 100%)
3. 实践
   - 用 flex 做两栏布局
   - 用 flex 做三栏布局
   - 用 flex 做四栏布局
   - 用 flex 做平均布局
   - 用 flex 组合使用，做更复杂的布局
   -
4. 经验
   - 永远不要把 width 和 height 写死，除非特殊说明
   - 用 min-width/max-width/min-height/max-height
   - flex 可以基本满足所有需求
   - flex 和 margin-xxx：auto 配合有意外的效果
5. 什么是写死
   - width:100px
6. 不写死
   - width:50%
   - max-width:100px
   - width:30vw
   - min-width:80%
   - 特点：不使用 px，或者加 min max 前缀

4.2.5 Grid 布局

二维布局用 Grid，一维布局用 Flex

语法

例子和语法

4.3 定位

布局与定位的区别是：布局是屏幕平面上的，定位是垂直于屏幕的

4.3.1 一个 div 的分层

4.3.2 positon 的五个取值

4.3.3 层叠上下文

4.4 动画

4.4.1 动画的原理

4.4.2 浏览器渲染的原理

浏览器渲染过程

1. 根据 HTML 构建 HTML 树 (DOM)
2. 根据 CSS 构建 CSS 树 (CSSOM)
3. 将两颗树合并成一颗渲染树 (render tree)
4. Layout 布局 (文档流，盒模型，计算大小和位置)
5. Paint 绘制 (把边框颜色，文字颜色，阴影等画出来)
6. Compose 合成 (根据层叠关系展示画面)

三棵树　图片以后放

如何更新样式　一般我们采用 JS 来更新样式

1. 比如 div.style.background=’red’
2. 比如 div.style.display=’none’
3. 比如 div.classList.add(’red’)
4. 比如 div.remove() 直接删掉节点

三种更新方式

1. JS/CSS > 样式 > 布局 > 绘制 > 合成
2. JS/CSS > 样式 > 绘制 > 合成
3. JS/CSS > 样式 > 合成

三种更新方式区别

1. 第一种，全走
    - div.remove() 会触发当前消失，其他元素 relayout
    -
2. 第二种，跳过 layout
    - 改变背景颜色，直接 repaint+composite
    -
3. 第三种，跳过 layout 和 paint
    - 改变 transform，只需 composite
    - 注意必须全屏查看效果，在 iframe 里看有问题
    -

4.4.3 CSS 动画优化

JS 优化

1. 使用 requestAnimationFrame 代替 setTimeout 或 setInterval

JS 优化

1. 使用 will-change 或 translate

参考文章

4.4.4 transition

位移 translate　缩放 scale　旋转 rotate　倾斜 skew

经验

1. 一般都不需要配合 transition 过度
2. inline 元素不支持 transform，需要先变成 block

translate

1. 常用写法
    - translateX(<length-percentage>)
    - translateY(<length-percentage>)
    - translate(<length-percentage>, <length-percentage>?)
    - translateZ(<length>) 且父容器 perspective
    - translate3d(x,y,z)
    - 演示
2. 经验
    - 看懂 MDN 语法示例
    - translate(-50%, -50%) 可做绝对定位元素的居中

scale

1. 常用写法
    - scaleX(<number>)
    - scaleX(<number>)
    - scaleX(<number>, <number>?)
    - 演示
2. 经验
    - 用的少

rotate

1. 常用写法
    - rotate([<angle>|<zero>])
    - rotateZ([<angle>|<zero>])
    - rotateX([<angle>|<zero>])
    - rotateY([<angle>|<zero>])
    - rotate3d 太复杂
    - 演示
2. 经验
    - 一般用于 360 度选择制作 loading
    - 用到的时候查 rotate MDN 文档

skew

1. 常用写法
    - skewX([<angle>|<zero>])
    - skewY([<angle>|<zero>])
    - skew([<angle>|<zero>],[<angle>|<zero>]?)
    - 演示
2. 经验
    - 用的较少
    - 用到的时候查 skew MDN 文档

transform 多重效果

1. 组合使用
    - transform:scale(0.5) translate(-100%, -100%);
    - transform:none; 取消所有

参考文章

4.4.5 transition 过渡

作用是补充中间帧

4.4.6 红心实践

css 需要想象力

5 HTTP

Hyper Text Transfer Protocol

5.1 URL

Uniform Resource Locator

协议 + 域名或 IP+ 端口号 + 路径 + 查询字符串 + 锚点

5.1.1 IP

Internet Protocal

1. 约定了两件事
   - 如何定位一台设备
   - 如何封装数据报文，以跟其他设备交流
2. 外网 IP
   - 从电信租用带宽，一年一千多。
   - 买了路由器，然后用电脑和手机分别连接路由器广播出来的无线 WIFI。
   - 路由器连上电信服务器，路由器有一个外围 IP，这是你互联网中的地址。
   - 重启路由器可能会被重新分配外围 IP，也就是路由器没有固定的外网 IP。
   - 连接路由器的手机和电脑是内网 IP。
3. 内网 IP
   - 路由器会在家里创建一个内网，内网设备使用内网 IP，一般是 192.169.xxx.xxx。
   - 一般路由器会给自己分配一个好记的内网 IP，如 192.168.1.1。
   - 然后路由器会给每一个内网中的设备分配不同的内网 IP。
   - 如电脑是 192.168.1.2，手机是 192.168.1.3。
4. 路由器的功能
   - 现在路由器会有两个 IP，一个外网 IP 和一个内网 IP。
   - 内网的设备可以互相访问，但是不能直接访问外网。
   - 内网设备想要访问外围，就必须经过路由器中转。
   - 外网中的设备可以互相访问，但是无法访问你的内网。
   - 外网设备想要把内容送到内网，也必须通过路由器。
   - 也就是说内网和外网就像两个隔绝的空间，无法互通，唯一的联通点就是路由器。

- 所以路由器有时候也被叫做网关。

5. 几个特殊的 IP

   - 127.0.0.1 表示自己。
   - localhost 通过 hosts 指定为自己。
   - 0.0.0.0 不表示任何设备。

5.1.2 端口

一台机器可以提供很多服务，每个服务一个号码，这个号码就叫端口号 port

1. 一个比喻

   - 麦当劳提供两个窗口，一号快餐，二号咖啡。
   - 你去快餐窗口点咖啡会被拒绝，让你去两一个窗口。
   - 你去咖啡窗口点快餐结果一样。

2. 一台机器可以提供不同服务

   - 要提供 HTTP 服务最好使用 80 端口。
   - 要提供 HTTPS 服务最好使用 443 端口。
   - 要提供 FTP 服务最好使用 21 端口。
   - 一共有 65535 个端口 (基本够用)。

3. 端口使用规则

   - 0 到 1023(2 的 10 次方减 1) 号端口是留给系统使用的。
   - 你只有拥有了管理员权限后，才能使用这 1024 个端口。
   - 其他端口可以给普通用户使用。
   - 比如 http-server 默认使用 8080 端口。
   - 一个端口如果被占用，你就只能换一个端口。

5.1.3 域名

1. 域名就是 IP 的别称

   - baidu.com 对应的什么 IP –> ping baidu.com
   - qq.com 对应的什么 IP –> ping qq.com
   - 一个域名可以对应不同 IP。
   - 这个叫做均衡负载，防止一台机器扛不住。
   - 一个 IP 可以对应不同域名。
   - 这个叫做共享主机，穷开发者会这么做。

2. 域名和 IP 是怎么对应起来的

   - 通过 DNS

3. 当你输入 qq.com 的过程

   - 你的 Chrome 浏览器会向电信提供的 DNS 服务器询问 qq.com 对应什么 IP。

- 电信会回答一个 IP(具体过程很发杂，不研究)。
- 然后 Chrome 才会向对应 IP 的 80/443 端口发送请求。
- 请求内容是查看 qq.com 的首页。

4. 为什么是 80 或 443 端口

- 服务器默认用 80 提供 http 服务。
- 服务器默认用 443 提供 https 服务。
- 可以在开发者工具看到具体的端口。

5. 题外话

- www.caosiyuan.com 和 caosiyuan.com 不是同一域名。
- comn 是顶级域名。
- caosiyuan.com 是二级域名 (俗称一级域名)。
- www.caosiyuan.com 是三级域名 (俗称二级)。
- 他们是父子关系
- 比如 github.io 把子域名 xx.github.io 免费给你使用
- 但 www.caosiyuan.com 和 caosiyuan.com 可以不是同一家公司，也可以是。
- www 非常多余

6. 如何请求不同的页面

- 路径可以做到
- `https://developer.mozilla.org/zh-CN/docs/Web/HTML`
- `https://developer.mozilla.org/zh-CN/docs/Web/CSS`
- 使用 chrome 开发者工具 Network 面板看区别。
- 有点类似爬虫找规律。

7. 同一个页面，不同内容

- 查询参数可以做到
- `http://www.baidu.com/s?wd=hi`
- `http://www.baidu.com/s?wd=hello`

8. 同一个页面，不同位置

- 锚点可以做到
- `https://developer.mozilla.org/zh-CN/docs/Web/CSS#`
- `https://developer.mozilla.org/zh-CN/docs/Web/CSS#`
- 注意，锚点看起来有中文，但实际不支持中文。
- # 参考书会变成 #%E5%8F...。
- 锚点是无法在 Network 面板看到。
- 锚点不会传给服务器。

5.1.4 HTTP 协议

基于 TCP 和 IP 两个协议，规定了请求的格式是什么，响应的格式是什么

1. 用 curl 可以发 HTTP 请求
   - curl -v http://baidu.com
   - curl -s -v – https://www.baidu.com
2. 理解一下概念
   - url 会被 curl 工具重写，先请求 DNS 获得 IP
   - 先进行 TCP 连接，TCP 连接成功后，开始发送 HTTP 请求
   - 请求内容看一眼
   - 响应内容看一眼
   - 响应结束后，关闭 TCP 连接 (看不出来)
   - 真正结束

5.2 请求响应和 NodeJS Sever

6 JS

6.1 概览

JS 需要一点逻辑能力，数学学的好不用担心，因为比数学简单太多了。

6.1.1 硬要求

1. 足够的代码量
   - 达到 1000 行 –> 新手
   - 达到 10000 行 –> 熟手
   - 达到 50000 行 –> 专业选手
   - 只能靠时间积累，人生就是奋斗，最快一年就可达到。
2. 如何统计自己的代码行数
   - 安装 yarn global add cloc
   - 在项目文件下使用 cloc –vcs=git.
   - 注意把仓库里 node_modules 等不想关内容写入.gitignore
3. 了解最够多的概念，不仅会写，还要会说
   - 常用考点：闭包，原型，类，继承，MVC，Flux，高阶函数，前端工程化
   - 博客总结，代码实践，多多积累。
4. 有足够的踩坑经验
   - 把该领域内所有的错误都犯完的人，就是专家。
   - 多做个人项目，全方位踩坑。

6.1.2 JS 的历史

1. JavaScript 的诞生

   - 布兰登生平自行了解，我的总结是成为了领导后千万不能犯错。
   - 牛逼的程序员不怕辞退，很容易创业，可以干到 50 岁以上。
   - 公司要求 JS 的命名蹭 Java 的流量，现在各行各业也存在者这种营销。
   - 由于版权问题，JS 又叫 ECMAScript。
   - 布兰登十天设计了 JS 最初版本 (不是实现)，所以 JS 有很多 bug。
   - 网景被微软收购，IE6 如日中天。
   - 2004 谷歌雇佣了一些 Firefox 和 IE 的开发者。
   - 2016 年 Chrome 全球份额 62%, 横空出世。
   - 移动市场智能手机的崛起。

2. JavaScript 的兴起

   - 2004 年愚人节，谷歌发布 Gmail 在线网页，当时人们认为网页只能看新闻和图片。
   - 2005 年，Jesse 将谷歌用到的技术命名为 AJAX，从此，前端技术正式出现。
   - 用历史唯物主义的观念看，正如现在很多前端概念就是过去技术的打包。
   - 在此之前的网页开发都是由后端和设计师完成。
   - 2006 年，jQuery 发布，是目前最长寿的 JS 库。
   - 后来的十年，jQuery 大放异彩，直到 IE 不行了，才稍微没有那么火。

3. 中国的前端

   - 正式出现时间是 2010 年左右，中国才有专门的前端岗位。
   - 可以用百度指数关键词搜索趋势。
   - 早期的前端是一些自学前端的后端程序员，他们把 Java 思想带入 JS
   - 因此面向对象成了 JS 的主流思想。
   - 行业还是很缺前端。

4. JavaScript 的爆发

   - Chrome 的 JS 引擎叫做 V8, V8 原本是跑车引擎的叫法，快如闪电。
   - 2009 年，Ryon 基于 V8 创建了 Node.js。
   - 2010 年，Isaac 基于 Node.js 写出了 npm。
   - 前端工程师可以在浏览器之外执行 JS 了，Node.js 快速风靡。
   - 同年，TJ 受 Sinatra 启发，发布了 Express.js。
   - 至此，前端工程师可以愉快的写后端应用了。
   - 至此，爆发了很多技术了，gulp，grunt，yeoman，requireJs，webpack 等。
   - JS 是历史的选择，一开始是玩具，但 JS 走对了风口，所以活到了最后。
   - 总结：类似考研政治，历史人物可以影响事物的进程，但决定不了历史的发展的方向。

6.2 内存图与 JS 世界

6.2.1 操作系统常识

一切都运行在内存里

1. 开机
   - 操作系统在 C 盘里 (macOS 的在根目录下多个目录里)
   - 当按下开机键，主板通电，开始读取固件
   - 固件就是固定在主板上的存储设备，里面有开机程序
   - 开机程序会将文件里的操作系统加载到内存中运行
2. 操作系统 (以 Linux 为例)
   - 首先加载操作系统内核
   - 然后启动初始化进程，编号为 1，每个进程都有编号
   - 启动系统服务：文件，安全，联网
   - 等待用户登录：输入密码登录/ssh 登录
   - 登录后，运行 shell，用户就可以和操作系统对话了
   - bash 是一种 shell，图形化界面可认为是一种 shell
3. 打开浏览器 (chrome.exe)
   - 你双击 Chrome 图标，就会运行 chrome.exe 文件
   - 开启 Chrome 进程，作为主进程
   - 主进程会开启一些辅助进程，如网络服务，GPU 加速
   - 你每新建一个网页，就有可能会开启一个子进程
4. 浏览器的功能
   - 发起请求，下载 HTML，解析 HTML，下载 CSS，解析 CSS
   - 渲染界面，下载 JS，解析 JS，执行 JS 等
   - 功能模块：用户界面，渲染引擎，JS 引擎，存储等
   - 以上功能模块一般各处于不同的线程 (比进程更小)
   - 如果进程是车间，那么线程就是车间里的流水线
5. JS 引擎
   - Chrome 用的是由 C++ 编写的 V8 引擎
   - 网景用的是 SpiderMonkey，后被 Firefox 使用
   - Safari 用的是 JavaScriptCore
   - IE 用的是 Chakra(JScript9)
   - Edge 用的是 Chakra(JavaScript)
   - Node.js 用的是 V8 引擎
6. JS 引擎的功能
   - 编译：把 JS 代码翻译为机器能执行的字节码或机器码

- 优化：改写代码，使其更高效
- 执行：执行上面的字节码或者机器码
- 垃圾回收：把 JS 用完的内存回收，方便之后再次使用

7. 执行 JS 代码的准备工作

- 浏览器提供 API：window/document/setTimeout
- 没错，上面东西都不是 JS 自身具备的功能
- 我们将这些功能称为运行环境 runtime env
- 一旦把 JS 放进页面，就开始执行 JS
- JS 代码在内存里运行，看下部分内存图

6.2.2 内存图

要求会画内存图

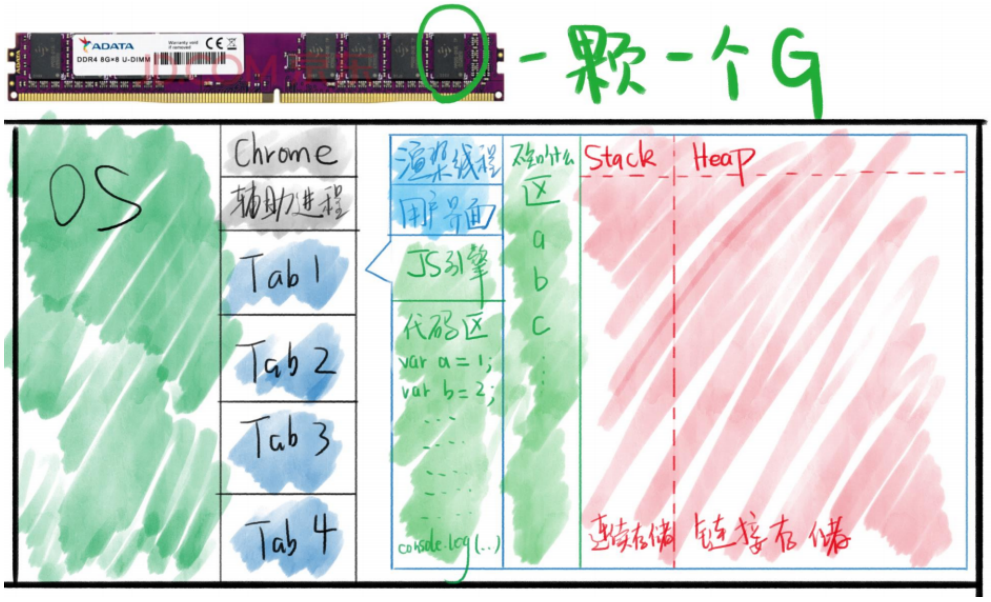

图 1: 瓜分内存图

1. 红色区域的作用

- 红色专门用来存放数据，我们目前只研究该区域
- 红色区域并不存变量名，变量名在不知什么区
- 每种浏览器的分配规则并不一样
- 还有调用栈，任务队列尚未画出

2. Stack 和 Heap

- 红色区域分为 Stack 栈和 Heap 堆
- 栈和堆需要用到数据结构知识
- Stack 区特点：每个数据顺序存放
- Heap 区特点：每个数据随机存放

3. js 代码在 Heap 和 Stack 区的执行过程

```
var a = 1
var b = a
var person = {name: 'syuancao', hobby: 'coding'}
var person2 = person
```

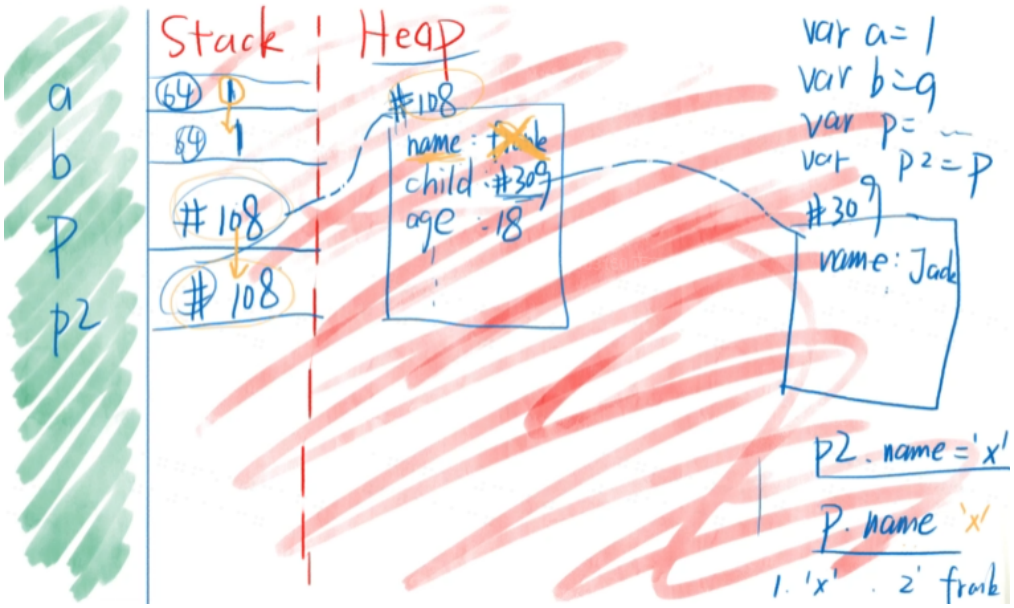

图 2: js 代码在 Heap 和 Stack 区执行的内存图

4. 规律

- 数据分两种：非对象和对象
- 非对象都存在 Stack (数字，字符串，布尔不是对象)
- 对象都存在 Heap (数组是对象，函数是对象)
- = 号总是会把右边的东西复制到左边 (不存在什么传值和传址，是直接拷贝，图中箭头指向是虚的)
- 很多书上会让你区分值和地址，只有不会画内存图的人才需要做这件事