ouvens / frontend-system-map

Watch 109 Unstar 866 Fork 220

Code Issues 2 Pull requests 0 Projects 0 Wiki Pulse Graphs

init

23 commits 1 branch 0 releases 1 contributor

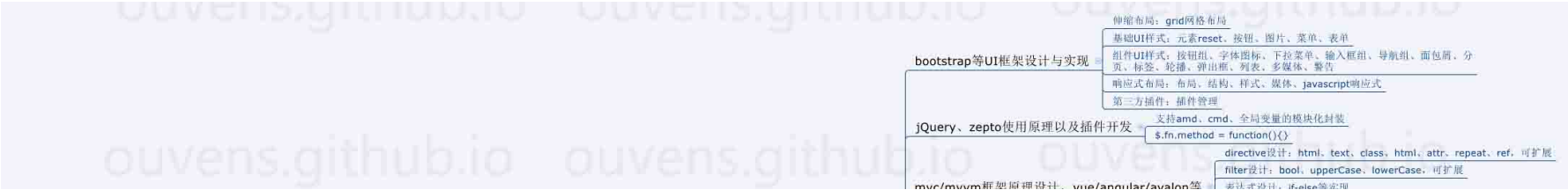
Branch: master New pull request Create new file Upload files Find file Clone or download

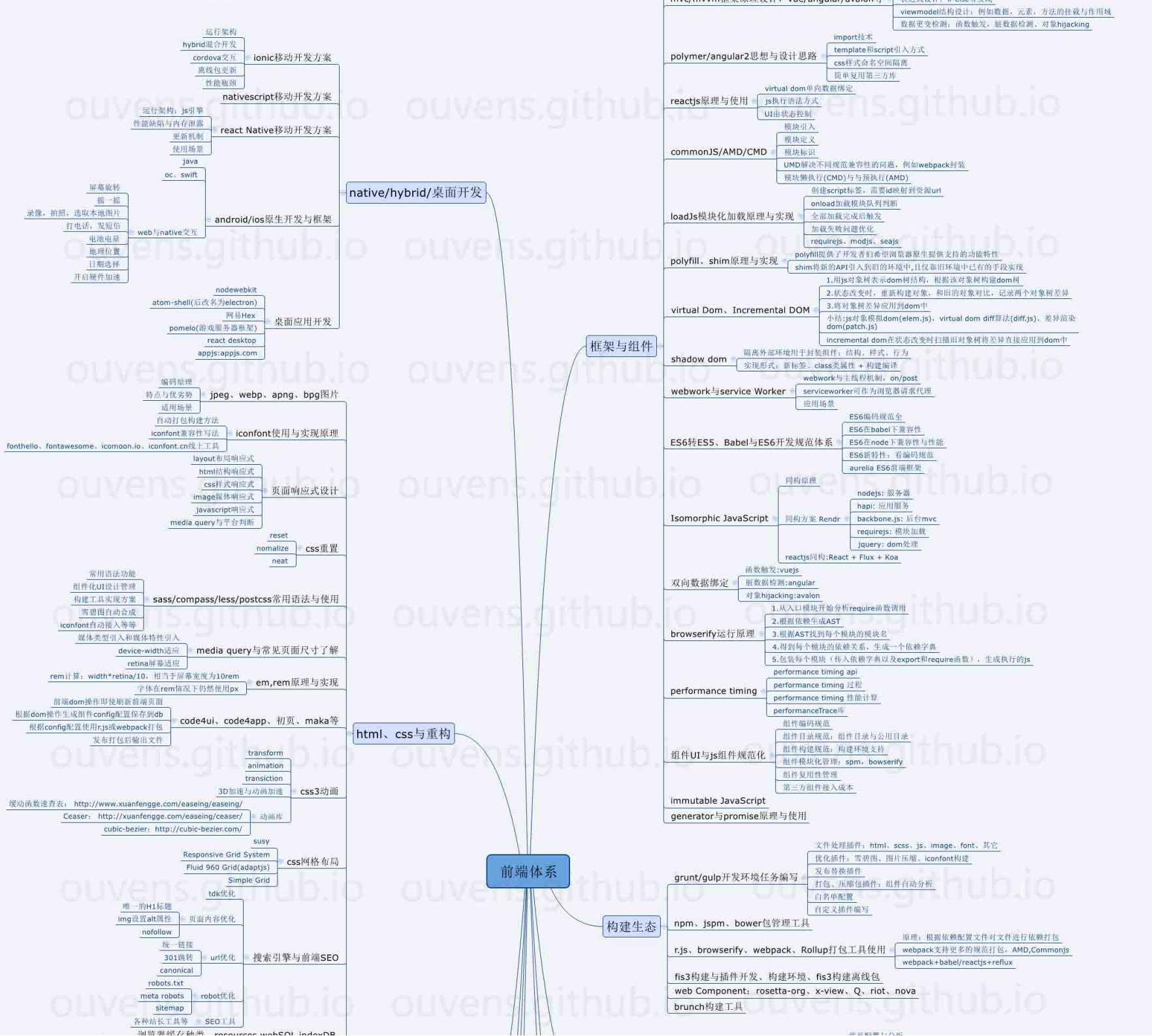
ouvens add Latest commit db0872a on 12 Sep

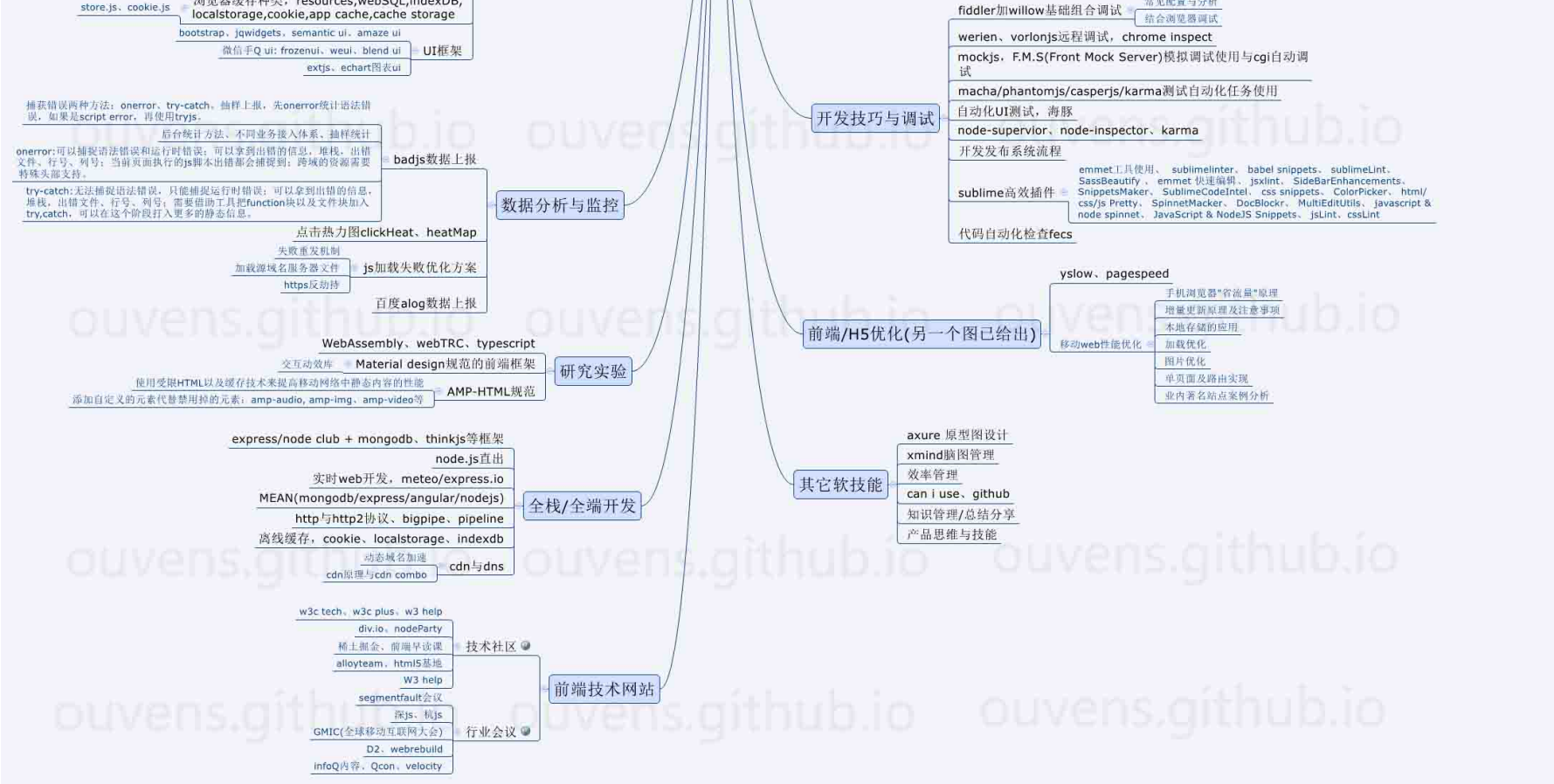
v1	add	2 months ago
.gitattributes	Added .gitattributes & .gitignore files	10 months ago
.gitignore	Added .gitattributes & .gitignore files	10 months ago
README.md	add	3 months ago
前端体系-清晰.jpg	add	10 months ago

README.md

2015-2016前端知识体系




前端体系
native/hybrid/桌面开发
ionic移动开发方案
运行架构
hybrid混合开发
cordova交互
离线包更新
性能瓶颈
nativescript移动开发方案
react Native移动开发方案
运行架构：js引擎
性能缺陷与内存泄露
更新机制
使用场景
android/ios原生开发与框架
java
oc、swift
web与native交互
屏幕旋转
摇一摇
录像，拍照，选取本地图片
打电话，发短信
电池电量
地理位置
日期选择
开启硬件加速
桌面应用开发
nodewebkit
atom-shell(后改名为electron)
网易Hex
pomelo(游戏服务器框架)
react desktop
appjs:appjs.com
html、css与重构
jpeg、webp、apng、bpg图片
编码原理
特点与优劣势
适用场景
iconfont使用与实现原理
自动打包构建方法
iconfont兼容性写法
fonthello、fontawesome、icomoon.io、iconfont.cn线上工具
页面响应式设计
layout布局响应式
html结构响应式
css样式响应式
image媒体响应式
javascript响应式
media query与平台判断
css重置
reset
nomalize
neat
sass/compass/less/postcss常用语法与使用
常用语法功能
组件化UI设计管理
构建工具实现方案
雪碧图自动合成
iconfont自动接入等等
media query与常见页面尺寸了解
媒体类型引入和媒体特性引入
device-width适应
retina屏幕适应
em,rem原理与实现
rem计算：width*retina/10，相当于屏幕宽度为10rem
字体在rem情况下仍然使用px
code4ui、code4app、初页、maka等
前端dom操作即使刷新前端页面
根据dom操作生成组件config配置保存到db
根据config配置使用r.js或webpack打包
发布打包后输出文件
css3动画
transform
animation
transiction
3D加速与动画加速
动画库
缓动函数速查表：http://www.xuanfengge.com/easeing/easeing/
Ceaser：http://xuanfengge.com/easeing/ceaser/
cubic-bezier：http://cubic-bezier.com/
css网格布局
susy
Responsive Grid System
Fluid 960 Grid(adaptjs)
Simple Grid
搜索引擎与前端SEO
tdk优化
页面内容优化
唯一的H1标题
img设置alt属性
nofollow
url优化
统一链接
301跳转
canonical
robot优化
robots.txt
meta robots
sitemap
SEO工具
各种站长工具等
浏览器缓存种类：resources,webSQL,indexDB
框架与组件
mvc,mvvm框架原理设计：vue,angular,avalon对比
viewmodel结构设计：例如数据，元素，方法的挂载与作用域
数据更变检测：函数触发，脏数据检测、对象hijacking
polymer/angular2思想与设计思路
import技术
template和script引入方式
css样式命名空间隔离
简单复用第三方库
reactjs原理与使用
virtual dom单向数据绑定
js执行语法方式
UI由状态控制
commonJS/AMD/CMD
模块引入
模块定义
模块标识
UMD解决不同规范兼容性的问题，例如webpack封装
模块懒执行(CMD)与与预执行(AMD)
loadJs模块化加载原理与实现
创建script标签，需要id映射到资源url
onload加载模块队列判断
全部加载完成后触发
加载失败问题优化
requirejs、modjs、seajs
polyfill、shim原理与实现
polyfill提供了开发者们希望浏览器原生提供支持的功能特性
shim将新的API引入到旧的环境中,且仅靠旧环境中已有的手段实现
virtual Dom、Incremental DOM
1.用js对象树表示dom树结构，根据该对象树构建dom树
2.状态改变时，重新构建对象，和旧的对象对比，记录两个对象树差异
3.将对象树差异应用到dom中
小结:js对象模拟dom(elem.js)，virtual dom diff算法(diff.js)、差异渲染dom(patch.js)
incremental dom在状态改变时扫描旧对象树将差异直接应用到dom中
shadow dom
隔离外部环境用于封装组件：结构，样式，行为
实现形式：新标签、class类属性 + 构建编译
webwork与service Worker
webwork与主线程机制，on/post
serviceworker可作为浏览器请求代理
应用场景
ES6转ES5、Babel与ES6开发规范体系
ES6编码规范全
ES6在babel下兼容性
ES6在node下兼容性与性能
ES6新特性：看编码规范
aurelia ES6前端框架
Isomorphic JavaScript
同构原理
同构方案 Rendr
nodejs：服务器
hapi：应用服务
backbone.js：后台mvc
requirejs：模块加载
jquery：dom处理
reactjs同构:React + Flux + Koa
双向数据绑定
函数触发:vuejs
脏数据检测:angular
对象hijacking:avalon
browserify运行原理
1.从入口模块开始分析require函数调用
2.根据依赖生成AST
3.根据AST找到每个模块的模块名
4.得到每个模块的依赖关系，生成一个依赖字典
5.包装每个模块（传入依赖字典以及export和require函数），生成执行的js
performance timing
performance timing api
performance timing 过程
performance timing 性能计算
performanceTrace库
组件UI与js组件规范化
组件编码规范
组件目录规范：组件目录与公用目录
组件构建规范：构建环境支持
组件模块化管理：spm，bowserify
组件复用性管理
第三方组件接入成本
immutable JavaScript
generator与promise原理与使用
构建生态
grunt/gulp开发环境任务编写
文件处理插件：html、scss、js、image、font、其它
优化插件：雪碧图、图片压缩、iconfont构建
发布替换插件
打包、压缩包插件：组件自动分析
白名单配置
自定义插件编写
npm、jspm、bower包管理工具
r.js、browserify、webpack、Rollup打包工具使用
原理：根据依赖配置文件对文件进行依赖打包
webpack支持更多的规范打包，AMD,Commonjs
webpack+babel/reactjs+reflux
fis3构建与插件开发、构建环境、fis3构建离线包
web Component：rosetta-org、x-view、Q、riot、nova
brunch构建工具



总结了下前端这两年的主流技术，大部分技术在我的博客里有较深入的研究学习，对应技术博客地址：http://ouvens.github.io ，博客持续更新中，欢迎大家关注~

一、框架与组件

bootstrap等UI框架设计与实现

- 伸缩布局：grid网格布局
- 基础UI样式：元素reset、按钮、图片、菜单、表单
- 组件UI样式：按钮组、字体图标、下拉菜单、输入框组、导航组、面包屑、分页、标签、轮播、弹出框、列表、多媒体、警告

- 响应式布局：布局、结构、样式、媒体、javascript响应式
- 第三方插件：插件管理

jQuery、zepto使用原理以及插件开发

- 支持amd、cmd、全局变量的模块化封装
- $.fn.method = function(){}

mvc/mvvm框架原理设计，vue/angular/avalon等

- directive设计：html、text、class、html、attr、repeat、ref，可扩展
- filter设计：bool、upperCase、lowerCase，可扩展
- 表达式设计：if-else等实现
- viewmodel结构设计：例如数据，元素，方法的挂载与作用域
- 数据更变检测：函数触发，脏数据检测、对象hijacking

polymer/angular2思想与设计思路

- import技术
- template和script引入方式
- css样式命名空间隔离
- 简单复用第三方库

reactjs原理与使用

- virtual dom单向数据绑定
- js执行语法方式

- UI由状态控制

commonJS/AMD/CMD

- 模块引入
- 模块定义
- 模块标识
- UMD解决不同规范兼容性的问题，例如webpack封装
- 模块懒执行(CMD)与与预执行(AMD)

loadJs模块化加载原理与实现

- 创建script标签，需要id映射到资源url
- onload加载模块队列判断
- 全部加载完成后触发
- 加载失败问题优化
- requirejs、modjs、seajs

polyfill、shim原理与实现

- polyfill提供了开发者们希望浏览器原生提供支持的功能特性
- shim将新的API引入到旧的环境中,且仅靠旧环境中已有的手段实现

virtual Dom、Incremental DOM

- 1.用js对象树表示dom树结构，根据该对象树构建dom树
- 2.状态改变时，重新构建对象，和旧的对象对比，记录两个对象树差异

- 3.将对象树差异应用到dom中
- 小结:js对象模拟dom(elem.js)，virtual dom diff算法(diff.js)、差异渲染dom(patch.js)
- incremental dom在状态改变时扫描旧对象树将差异直接应用到dom中

shadow dom

- 隔离外部环境用于封装组件：结构、样式、行为
- 实现形式：新标签、class类属性 + 构建编译

webwork与service Worker

- webwork与主线程机制，on/post
- serviceworker可作为浏览器请求代理
- 应用场景

ES6转ES5、Babel与ES6开发规范体系

- ES6编码规范全
- ES6在babel下兼容性
- ES6在node下兼容性与性能
- ES6新特性：看编码规范
- aurelia ES6前端框架

Isomorphic JavaScript

- 同构原理
- 同构方案 Rendr

- nodejs: 服务器
- hapi: 应用服务
- backbone.js: 后台mvc
- requirejs: 模块加载
- jquery: dom处理
- reactjs同构:React + Flux + Koa

双向数据绑定

- 函数触发:vuejs
- 脏数据检测:angular
- 对象hijacking:avalon

browserify运行原理

- 1.从入口模块开始分析require函数调用
- 2.根据依赖生成AST
- 3.根据AST找到每个模块的模块名
- 4.得到每个模块的依赖关系，生成一个依赖字典
- 5.包装每个模块（传入依赖字典以及export和require函数），生成执行的js

performance timing

- performance timing api
- performance timing 过程

- performance timing 性能计算
- performanceTrace库

组件UI与js组件规范化

- 组件编码规范
- 组件目录规范：组件目录与公用目录
- 组件构建规范：构建环境支持
- 组件模块化管理：spm，bowserify
- 组件复用性管理
- 第三方组件接入成本

immutable JavaScript

generator与promise原理与使用

二、构建生态

grunt/gulp开发环境任务编写

- 文件处理插件：html、scss、js、image、font、其它
- 优化插件：雪碧图、图片压缩、iconfont构建
- 发布替换插件
- 打包、压缩包插件：组件自动分析
- 白名单配置
- 自定义插件编写

npm、jspm、bower包管理工具

r.js、browserify、webpack、Rollup打包工具使用

- 原理：根据依赖配置文件对文件进行依赖打包
- webpack支持更多的规范打包，AMD,Commonjs
- webpack+babel/reactjs+reflux

fis3构建与插件开发、构建环境、fis3构建离线包

web Component：rosetta-org、x-view、Q、riot、nova

brunch构建工具

三、开发技巧与调试

fiddler加willow基础组合调试

- 常见配置与分析
- 结合浏览器调试

werien、vorlonjs远程调试，chrome inspect

mockjs，F.M.S(Front Mock Server)模拟调试使用与cgi自动调试

macha/phantomjs/casperjs/karma测试自动化任务使用

自动化UI测试，海豚

node-supervior、node-inspector、karma

开发发布系统流程

sublime高效插件

- emmet工具使用、 sublimelinter、 babel snippets、 sublimeLint、 SassBeautify 、 emmet 快速编辑、 jsxlint、 SideBarEnhancements、 SnippetsMaker、 SublimeCodeIntel、 css snippets、 ColorPicker、 html/css/js Pretty、 SpinnetMacker、 DocBlockr、 MultiEditUtils、 javascript & node spinnet、 JavaScript & NodeJS Snippets、 jsLint、 cssLint

代码自动化检查fecs

四、html、css与重构

jpeg、webp、apng、bpg图片

- 编码原理
- 特点与优劣势
- 适用场景

iconfont使用与实现原理

- 自动打包构建方法
- iconfont兼容性写法
- fonthello、fontawesome、icomoon.io、iconfont.cn线上工具

页面响应式设计

- layout布局响应式
- html结构响应式
- css样式响应式
- image媒体响应式

- javascript响应式
- media query与平台判断

css重置

- reset
- nomalize
- neat

sass/compass/less/postcss常用语法与使用

- 常用语法功能
- 组件化UI设计管理
- 构建工具实现方案
- 雪碧图自动合成
- iconfont自动接入等等

media query与常见页面尺寸了解

- 媒体类型引入和媒体特性引入
- device-width适应
- retina屏幕适应

em,rem原理与实现

- rem计算：width*retina/10，相当于屏幕宽度为10rem
- 字体在rem情况下仍然使用px

code4ui、code4app、初页、maka等

- 前端dom操作即使刷新前端页面
- 根据dom操作生成组件config配置保存到db
- 根据config配置使用r.js或webpack打包
- 发布打包后输出文件

css3动画

- transform
- animation
- transiction
- 3D加速与动画加速
- 动画库
- 缓动函数速查表： http://www.xuanfengge.com/easeing/easeing/
- Ceaser： http://xuanfengge.com/easeing/ceaser/
- cubic-bezier：http://cubic-bezier.com/

css网格布局

- susy
- Responsive Grid System
- Fluid 960 Grid(adaptjs)
- Simple Grid

搜索引擎与前端SEO

- tdk优化
- 页面内容优化
- 唯一的H1标题
- img设置alt属性
- nofollow
- url优化
- 统一链接
- 301跳转
- canonical
- robot优化
- robots.txt
- meta robots
- sitemap
- SEO工具
- 各种站长工具等

浏览器缓存种类，resources,webSQL,indexDB, localstorage,cookie,app cache,cache storage

- store.js、cookie.js

UI框架

- bootstrap、jqwidgets、semantic ui、amaze ui

- 微信手Q ui: frozenui、weui、blend ui
- extjs、echart图表ui

五、native/hybrid/桌面开发

ionic移动开发方案

- 运行架构
- hybrid混合开发
- cordova交互
- 离线包更新
- 性能瓶颈

nativescript移动开发方案

react Native移动开发方案

- 运行架构：js引擎
- 性能缺陷与内存泄露
- 更新机制
- 使用场景

android/ios原生开发与框架

- java
- oc、swift
- web与native交互

- 屏幕旋转
- 摇一摇
- 录像，拍照，选取本地图片
- 打电话，发短信
- 电池电量
- 地理位置
- 日期选择
- 开启硬件加速

桌面应用开发

- nodewebkit
- atom-shell(后改名为electron)
- 网易Hex
- pomelo(游戏服务器框架)
- react desktop
- appjs:appjs.com

六、前端/H5优化(另一个图已给出)

yslow、pagespeed

移动web性能优化

- 手机浏览器"省流量"原理

- 增量更新原理及注意事项
- 本地存储的应用
- 加载优化
- 图片优化
- 单页面及路由实现
- 业内著名站点案例分析

七、全栈/全端开发

express/node club + mongodb、thinkjs等框架

node.js直出

实时web开发，meteor/express.io

MEAN(mongodb/express/angular/nodejs)

http与http2协议、bigpipe、pipeline

离线缓存，cookie、localstorage、indexdb

cdn与dns

- 动态域名加速
- cdn原理与cdn combo

八、研究实验

WebAssembly、webTRC、typescript

Material design规范的前端框架

- 交互动效库

AMP-HTML规范

- 使用受限HTML以及缓存技术来提高移动网络中静态内容的性能
- 添加自定义的元素代替禁用掉的元素：amp-audio, amp-img、amp-video等

九、数据分析与监控

badjs数据上报

- 捕获错误两种方法：onerror、try-catch。抽样上报，先onerror统计语法错误，如果是script error，再使用tryjs。
- 后台统计方法、不同业务接入体系、抽样统计
- onerror:可以捕捉语法错误和运行时错误；可以拿到出错的信息，堆栈，出错文件、行号、列号；当前页面执行的js脚本出错都会捕捉到；跨域的资源需要特殊头部支持。
- try-catch:无法捕捉语法错误，只能捕捉运行时错误；可以拿到出错的信息，堆栈，出错文件、行号、列号；需要借助工具把function块以及文件块加入try,catch，可以在这个阶段打入更多的静态信息。

点击热力图clickHeat、heatMap

js加载失败优化方案

- 失败重发机制
- 加载源域名服务器文件
- https反劫持

百度alog数据上报

十、其它软技能

axure 原型图设计

xmind脑图管理

效率管理

can i use、github

知识管理/总结分享

产品思维与技能

十一、前端技术网站

技术社区

- w3c tech、w3c plus、w3 help
- div.io、nodeParty
- 稀土掘金、前端早读课
- alloyteam、html5基地
- W3 help

行业会议

- segmentfault会议
- 深js、杭js
- GMIC(全球移动互联网大会)

- D2、webrebuild
- infoQ内容、Qcon、velocity

  后面此知识体系图会更新，内容更全，讲解更全面，同时将和与之讲解的书籍一起放出，敬请期待。

  持续更新中，如果觉得不错，请点star支持下，甚至在您的团队里扩散下，谢谢~