**Vue源码解读笔记**

[视频](https://coding.imooc.com/learn/list/228.html) （18513179925|a1111111）

# 【需知】

* Vue 使用了[Flow](https://flow.org/en/)（libdef的概念）做静态语法类型检查。
* Vue源码基于[Rollup](https://www.rollupjs.com/)构建，相关配置在scripts目录下。（webpack vs rollup vs gulp）
* Vue最终也是发布在了npm上。

【Vue.js源码目录】

|  |
| --- |
| --flow类型检查，有vue很多自定义类型  --src 源码目录  |\_\_compiler 编译器  |\_\_core 核心  |\_\_platforms支持的平台web、weex  |\_\_server 与服务器相关的  |\_\_sfc  |\_\_shared  --scripts脚本目录，对应所有的构建任务  --dist构建目录  --examples示例目录 |

【版本对比】

* Runtime Only

在使用此版本Vue.js时，通常需要借助如webpack的vue-loader工具把.vue文件编译成JavaScript，因为是在编译阶段做的，所以它只包含运行时的Vue.js代码，因此代码体积也会更轻量。

* Runtime + Compiler

在Vue.js2.0中，最终渲染都是通过render函数来实现的。若使用Vue的template属性，需要在客户端编译template为render函数。

显然，编译过程对性能有一定损耗，所以通常推荐使用Runtime-only版本的Vue.js.

[【入口】

platforms/web/entry-runtime-with-compiler.js

src/core/index.js

# 【Flow学习笔记】

* 安装
* 使用

1. 根目录使用flow init创建.flowconfig文件进行配置，改配置文件告诉flow后台程序进行flow语法检查的根（入口），自此项目级flow已启用。
2. 使用flow status指令启动后台程序对所有flow files进行错误检查。该程序会持续运行以检查新增代码中的错误。

使用flow指令默认执行flow status；

无论执行flow status多少次，任何时间flow status只会有一个后台程序在执行；

使用flow stop停止后台程序。

1. 使用// @flow或/\* @flow \*/将一个js文件标记为flow files。一般情况下，flow后台程序只会检查带有此标识的文件。除非调用flow check --all指令执行全局检查。
2. 编写flow代码，如参数类型校验、返回值类型校验，flow后台程序会自动进行类型检查
3. 检查编码。首次执行flow即启动flow后台程序对文件进行增量监督，再次执行flow指令最新的flow检查结果立即呈现。

* 类型注释

flow具有卓越的类型推断能力，因此，使用类型注释是flow辅助编码的一大亮点。

1. Primitive Types

MDN官网定义JS的基本类型有：Booleans、Strings、Numbers、null、undefined、Symbols。

对于字面量值（ture, “hello”, 3.14, null, undefined）用小写（boolean, string, number）;

对于包装对象（new Boolean(false)...）用大写（Boolean, String, Number）

* **boolean**：function acceptsBoolean(value: boolean) 接受字面量值（true、false、if语句及a===b等条件表达式隐式转换的结果），其他类型只能通过Boolean()转换；
* **number**：42、3.14、NaN、Infinity；
* **string**：接受“”、+拼接的string和number类型，其他类型必须转换（如：String({})、[].toString()、JSON.stringfy({})）；
* **null**（JS中的null）、**void**（JS中的undefined）；
* ?type：用于定义可选参数可能的类型，如：?string、?number，除了为type既定类型，还可以为null、void；
* 对象可选属性：{ propertyName?: string }，除type既定类型还可为void或忽略，不能为null；
* 函数可选参数：同上；
* 函数默认参数：function method(value: string = "default")，同上。
* symbol：Symbol()、Symbol.isConcat；

1. Literal Types字面量类型
2. Mixed Types
3. Any Types
4. Maybe Types

# 【Vue学习笔记】

参考链接：

<https://blog.csdn.net/quanxin0222/article/details/121310600>

<https://zhuanlan.zhihu.com/p/419896443>

## 1. compiler

编译即将template编译成render函数的过程。Vue.js提供了两个版本：

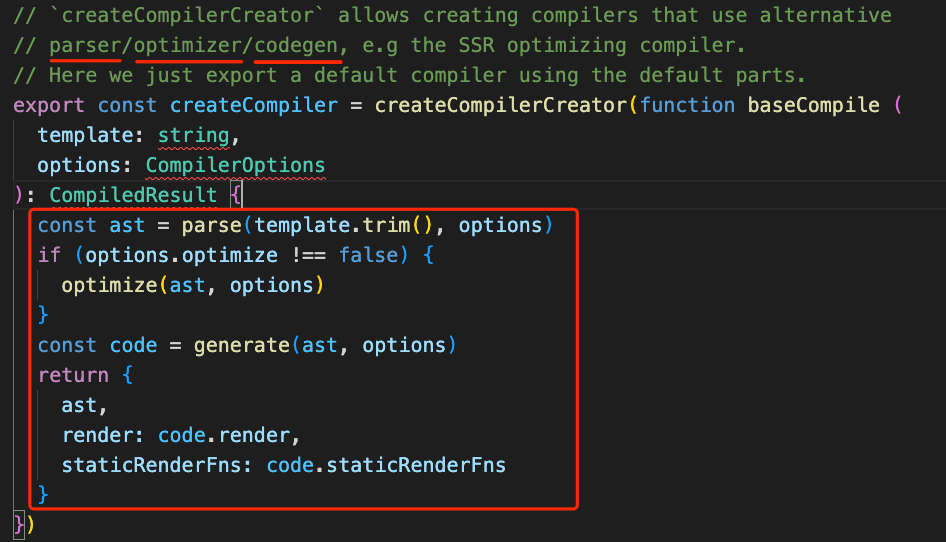
Runtime + Compiler版本，包含编译代码，将编译过程放在运行时做；

Runtime only版本，不包含编译代码，需借助webpack的vue-loader事先将模板编译成render函数。

|  |
| --- |
| compiler  |—codegen  |—directive  |—parser  |\_\_codeframe.js  |\_\_create-compiler.js  |\_\_error-detector.js  |\_\_helper.js  |\_\_index.js  |\_\_optimizer.js  |\_\_to-functions.js |

### 编译入口

由文件中，挂载在Vue原型上的$mount函数可知，当传入的options中没有render函数并且存在template模板时，则引入中的函数，由此追到src/compiler/index.js中的函数。



最后追溯到to-function.js中的createCompileToFunctionFn，即为入口。

【总结】

Vue.js在不同平台都会有编译过程，依赖的配置baseOptions会有所不同；

Vue.js利用了函数柯里化技巧，很好地实现了baseOptions参数的保留。同样，vue.js也是利用函数柯里化技巧把baseCompile函数抽出来，从而将真正的编译过程和其他逻辑（如：对编译配置的处理、缓存处理等）剥离开。

### 为何如此设计编译过程？

- 运行时和编译时

compiler编译步骤：

（1）如果有render函数，那么已经是编译完成的，返回；

（2）判断template：string | dom，离线编译（判断编译环境，执行compileToFunctions函数）；

（3）然后将template编译成函数；

parser：对模板进行AST解析（先分词，再做词法分析）

- html解析

- filter解析

- 分析v-for的key

- 优化：判断静态节点（纯dom、文本没有vue指令的是纯静态节点；若有子节点，子节点也应为静态节点）

## 2. core：核心

core目录

|——component：模板编译代码

|——global-api：文件接口

|——instance：实例，处理初始化、状态、数据、生命周期、事件

|——observer：数据订阅

|——utils：工具函数

|——vdom：虚拟dom，使用虚拟dom是因为原生dom有很多无用的属性，占用太多内存

- core是vue代码的核心，而 observer 是 core 的核心。它利用 Object.defineProperty 实现对数据的操作拦截，然后将数据绑定到一个 由 观察者模式 为单元（watcher）组成的数据维护中心（Dep）。

- 对于数组，新增或修改元素可能造成整个数组的元素重新排列，所以对数组进行了重写。所有新增的元素都调用 observer 方法，使其变为一个可观察的对象。

- 对于任何数据，都进行递归操作，使其任意属性变得可以追踪。

2.1 observer

（1）defineReactive

（2）observer

（3）watcher

（4）Dep

（5）scheduler

（6）nextTick

2.2 components

keep-alive，保存的是 vnode 节点，而不是数据。

由于 vnode 节点比描述状态的数据大一些，所以 keep-alive 能够保存的数据大小有限，所以它存在取舍问题，一般舍弃最老的组件。

对于任意组件，无论是否被添加到 keep-alive 缓存列表中，重新访问时，都会把它设置为列表的结尾。

2.3 use

用于为vue设置插件，它维护一个插件队列，判断是否已存在，如果未存在，执行插件，并且添加到插件队列中。

## 3. platform：web、weex平台

## 4. server：服务端渲染

## 5. sfc

单文件组件处理，将.vue文件的template、script、style拆分

## 6. shared：工具、常量