Vue模板编译

模板编译器的由来

模板编译的主要目标：将用户定义视图模板（template）转换为渲染函数（render function）

何时执行模板编译

1. 运行时编译：runtime-with-compiler

```
new Vue({
  template: '<div>string template</div>',
})
```

2. 预编译：webpack + vue-loader

```
// App.vue
//...

new Vue({
  render: h => h(App)
}).$mount()
```

如何编译模板

三个阶段：

解析：parse，模板转换为抽象语法树AST

优化：optimize，标记静态节点

生成：generate，抽象语法树AST转换为渲染函数

体验模板编译

```
// 输出render函数
console.log(app.$options.render)
```

```
(function anonymous(
) {
with(this){
    return _c('div',{attrs:{"id":"demo"}},[
        _c('h1',[_v("编译器")]),
        _v(" "),
        _c('p',[_v(_s(foo))]),_v(" "),
        _c('comp')],1)}
})
```

> _c：生成VNode

解析 - parse

将用户编写的template解析为AST

三个解析器：parseHTML，parseText，parseFilter

parse src\compiler\index.js

解析为ast

parse src\compiler\parser\index.js

具体解析函数

优化 - optimize

从AST中找出静态标记并打上记号。

两步：

1. 找出静态节点并标记
2. 找出静态根节点并标记

optimize src\compiler\optimizer.js

优化处理

代码生成 - generate

将AST转换成渲染函数中的内容，即代码字符串。

generate方法生成渲染函数代码，src/compiler/codegen/index.js

常见指令解析

v-if、v-else-if、v-else

解析时

代码生成时

render

```
render(h) {
  if(condition)
    return h(...)
  return h(...)
}
```