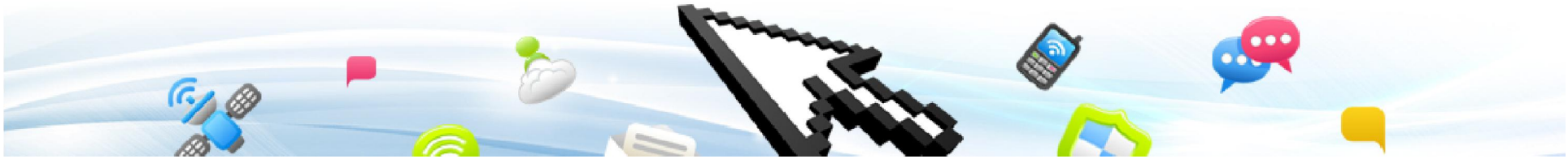
FIS 2.0 全新的百度前端解决方案

张云龙

Baidu 百度

Baidu 百度

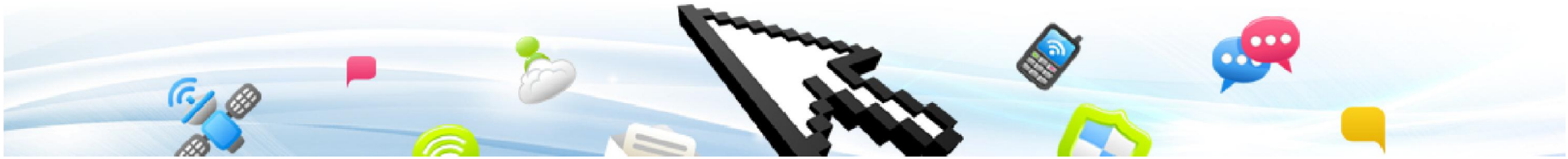
目录



fis是什么

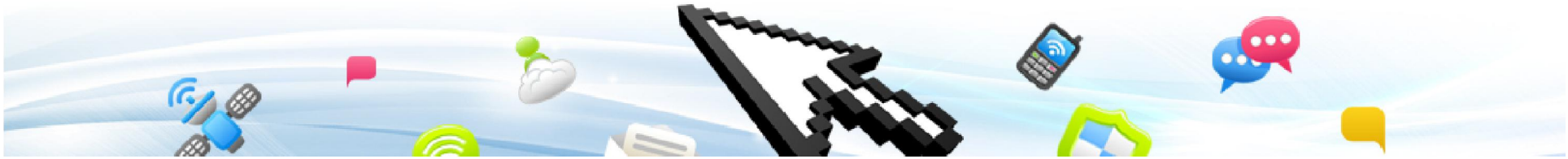
✓ F.I.S = Front-end Integrated Solution（**前端集成解决方案**）

- 项目起源
  - 2011年底，百度web前端研发部成立F.I.S团队，汇集了来自各产品线的精英工程师，该团队的任务是寻找提升前端工业生产力水平的解决方案。经过1年多的努力，fis团队和百度众多产品线共同探索出一套前端集成解决方案，解决了前端生成中遇到的诸多问题，包括但不限于**前端静态资源加载优化、页面运行性能优化、基础编译工具、运行环境模拟、**js**与**css**组件化开发**等前端领域核心问题。如今，fis团队的解决方案已应用到百度30多个产品线，覆盖了从pc到无线终端的应用，极大的提升了前端团队的生产力，降低了开发成本。
  - 2013年初，fis团队总结了之前在前端集成解决方案领域所做的探索，将整套方案整理开源，希望能为前端工业化提供新的思路。
- 官网地址
  - http://fis.baidu.com
- 项目地址
  - https://github.com/fis-dev/fis
  - https://github.com/fis-dev/fis-kernel
- 项目wiki
  - https://github.com/fis-dev/fis/wiki



似曾相识——某产品线成长日记

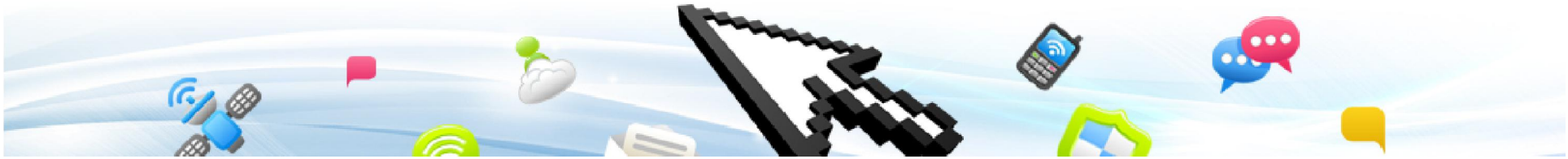
- ✓ 某团队要做一个全新的互联网产品
  - 第一天（规范定制）：
    - 定制必要的开发规范



似曾相识——某产品线成长日记

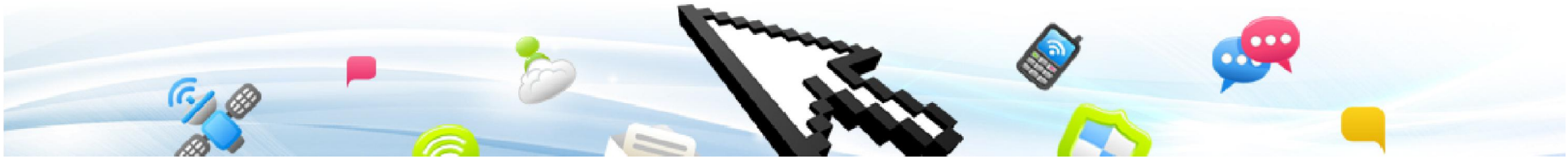
✓ 某团队要做一个全新的互联网产品

- 第一天（规范定制）：
  - 定制必要的开发规范
- 第二天（技术选型）：
  - 选择前端组件化框架（requirejs, seajs, ... ）
  - 选择前端基础库（jquery, tangram, ... ）
  - 选择前端组件库（jqueryui, magic, ... ）
  - 选择模板语言（smarty, php, ... ）
  - 选择模板插件（xss-repair ）



似曾相识——某产品线成长日记

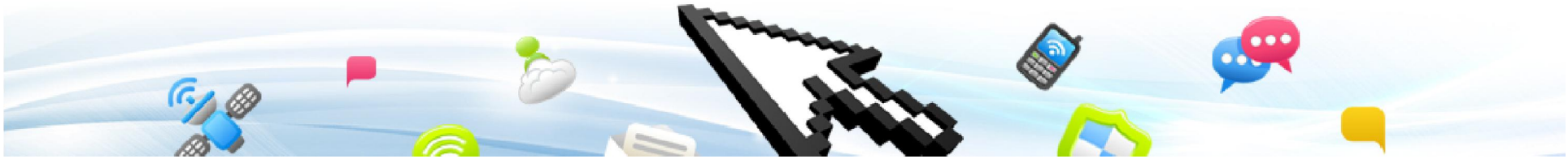
- ✓ 某团队要做一个全新的互联网产品
  - 第一天（规范定制）：
    - 定制必要的开发规范
  - 第二天（技术选型）：
    - 选择前端组件化框架（requirejs, seajs, ... ）
    - 选择前端基础库（jquery, tangram, ... ）
    - 选择前端组件库（jqueryui, magic, ... ）
    - 选择模板语言（smarty, php, ... ）
    - 选择模板插件（xss-repair ）
  - 一个月后（自动化与拆分）：
    - 选择或开发自动化工具（编译、打包、压缩、校验）
    - 组件化、模块化拆分系统，便于复用与维护



似曾相识——某产品线成长日记

✓ 某团队要做一个全新的互联网产品

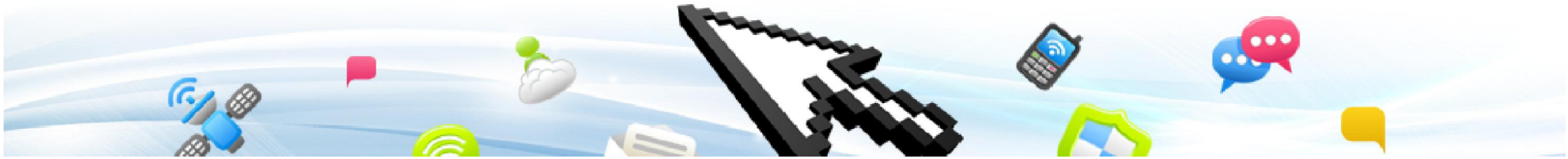
- 第一天（规范定制）：
  - 定制必要的开发规范
- 第二天（技术选型）：
  - 选择前端组件化框架（requirejs, seajs, ... ）
  - 选择前端基础库（jquery, tangram, ... ）
  - 选择前端组件库（jqueryui, magic, ... ）
  - 选择模板语言（smarty, php, ... ）
  - 选择模板插件（xss-repair ）
- 一个月后（自动化与拆分）：
  - 选择或开发自动化工具（编译、打包、压缩、校验）
  - 组件化、模块化拆分系统，便于复用与维护
- 半年以后（性能优化）：
  - 小心剔除已下线功能的资源
  - 优化http请求
  - 适当调整工具以适应性能优化
  - 使用架构级优化方案：BigPipe、PageCache、Quickling、BigRender等



F.I.S(前端集成解决方案)

- ✓ 如果你的团队已经拥有以上四项技术基础，那么恭喜，你已拥有了一套fis。
- ✓ 这样的过程每天都在前端业界上演，有没有办法：
  - 简化这个过程，让前端团队可以快速进入开发阶段
  - 不用担心未来的自动化、性能优化等问题
  - 兄弟团队可以分享彼此的技术成果
  - 不用苦等后端服务ready就能先行开始前端项目
  - 妈妈再也不用担心我的开发
- ✓ Yes，we can！

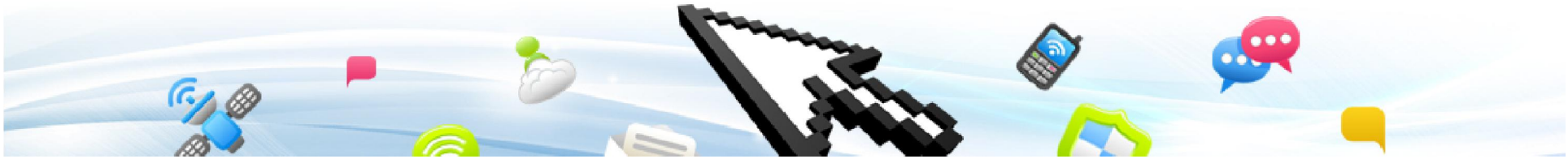
https://github.com/fis-dev/fis/wiki/%E4%BB%80%E4%B9%88%E6%98%AFF.I.S



使用fis请别担心

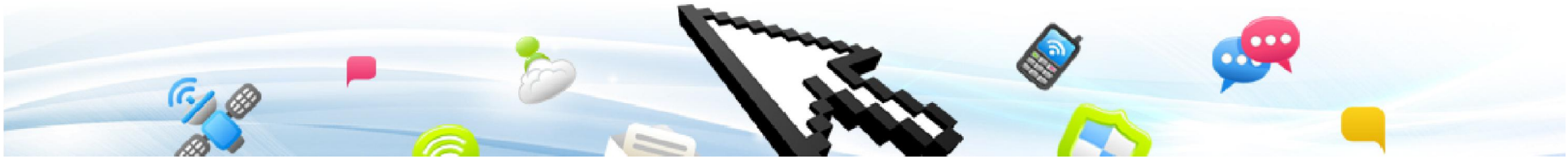
- ✓ 有些人会担心：
  - 我们后端架构是java/c++/php/ASP.NET，fis会不会不适用啊？
  - 我们的项目是纯前端的，没有模板支持，fis会不会不适用啊？
  - 我们只是个小团队，用fis会不会太重啊？
  - 我们的后端部署跟别家不太一样，fis会不会对部署有要求啊？
  - 我们不用前端组件化框架，fis会不会强制我们使用这类框架啊？
  - fis会不会捆绑tangram、magic啊？
  - fis会不会很难学很难用啊？
  - fis会不会对目录结构有规定啊？
  - fis会不会帮我找到女朋友啊？

  ……
- ✓ 以上所有问题的答案都是：不会！！！
- ✓ 那么，fis是如何做到这一切的呢？



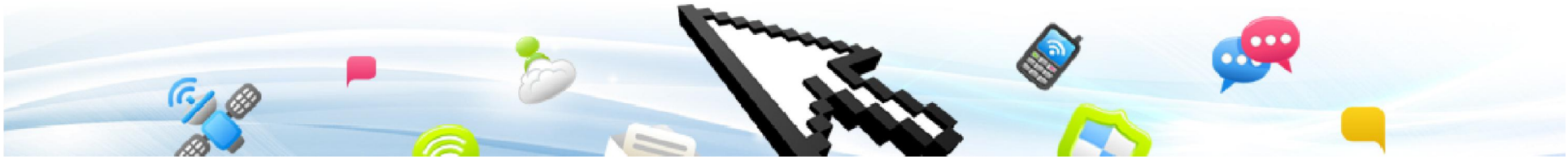
Baidu 百度

目录



关于规则的启发

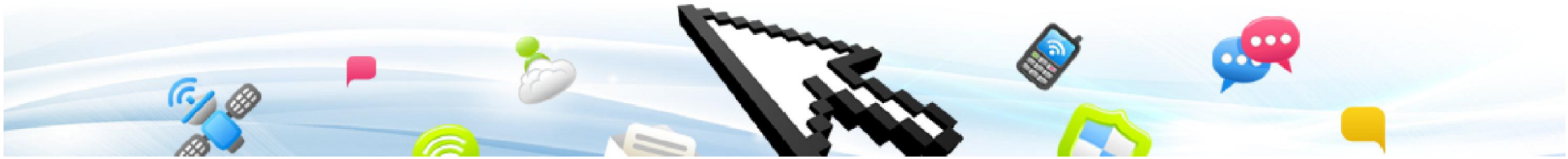
- ✓ 关于作图的问题：
  - 如何平分一条线段：刻度尺测量一下
  - 如何画出一个正三角型：刻度尺+量角器
  - 如何平分一个角：量角器+直尺
  - 如何画一个圆：圆规
  - 如何做已知直线的垂线：直角三角板
- ✓ 完成以上作图我们需要这么多工具么？
  - 刻度尺+量角器+直尺+圆规+三角板
- ✓ 答案连小学生都知道：我们只需要尺规！



前端开发的“尺”与“规”

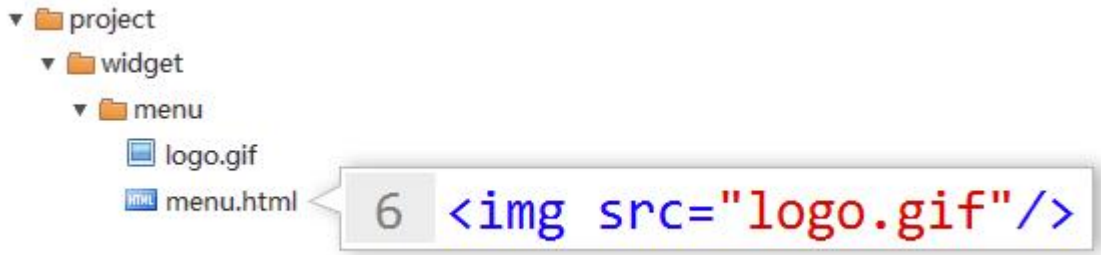
- ✓ 满足前端开发需求的最小规则集合（三种语言能力）：
  - 资源定位：定位任何开发中所使用资源的线上路径。
  - 资源嵌入：把一个文件的内容(文本)或者base64编码(图片)嵌入到另一个文件中。
  - 依赖声明：在一个文本文件内标记对其他资源的依赖关系。

https://github.com/fis-dev/fis/wiki/%E4%B8%89%E7%A7%8D%E8%AF%AD%E8%A8%80%E8%83%BD%E5%8A%9B

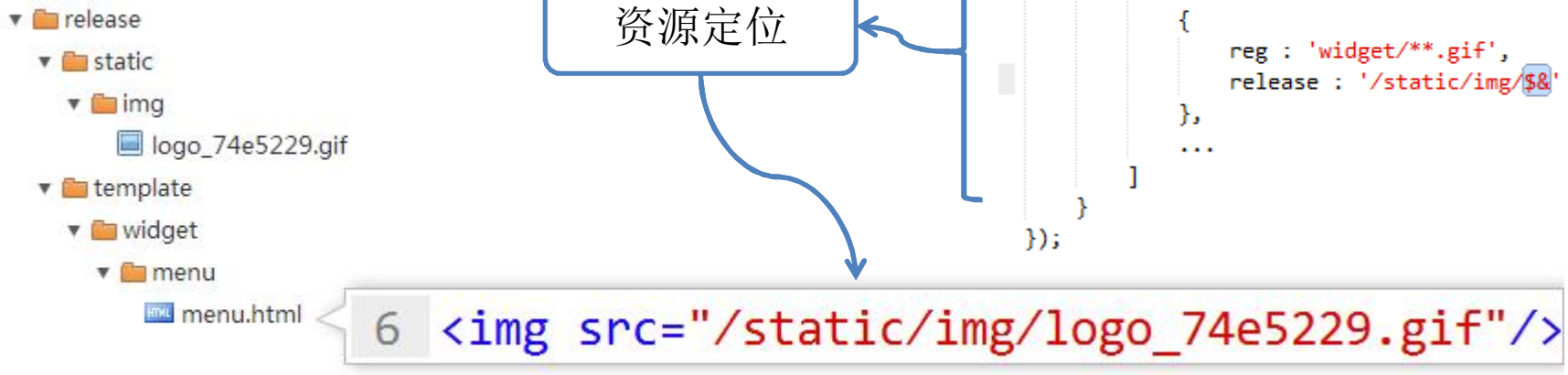


资源定位

✓ 开发中：

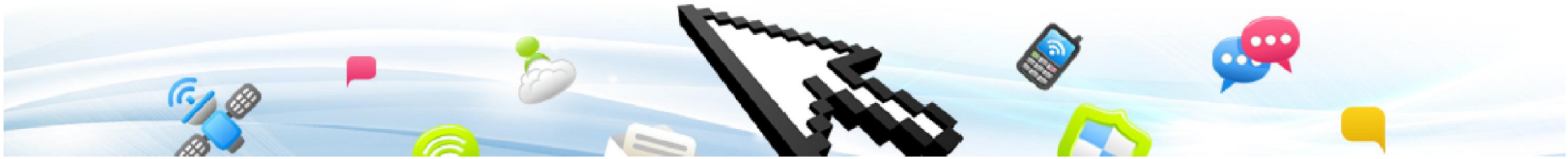


✓ 上线后：



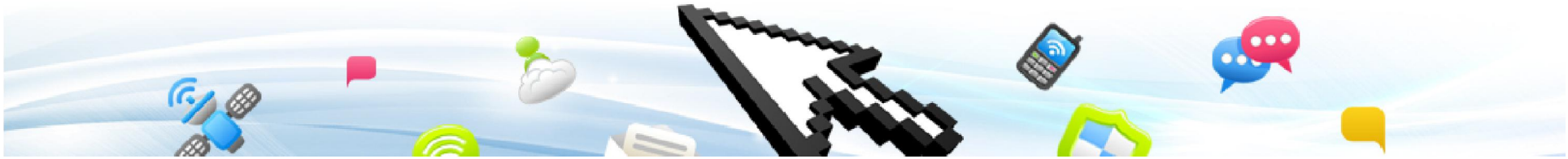
资源定位

```
//配置文件
fis.config.merge({
    roadmap {
        path : [
            ...
            {
                reg : 'widget/**.gif',
                release : '/static/img/$&'
            },
            ...
        ]
    }
});
```



资源定位

- ✓ 有效的分离开发路径与部署路径之间的关系
  - 工程师不再关心资源部署到线上之后去了哪里，变成了什么名字，这些都可以通过配置来指定
- ✓ 代码具有很强的可移植性
  - 由于开发路径与部署路径对工程师透明，因此组件的资源依赖全部都是相对路径定位的，这样，对于两个同样使用fis作为开发平台的团队，即便他们的部署方式完全不同也可以有效移植代码。
- ✓ 轻松实现md5、域名添加等功能
  - 工程师完全不用关心上线后资源的静态服务器域名是什么
  - 资源会全部自动添加md5作为版本戳，服务器可以开启强缓存、回滚时不需要回滚静态资源，只须回滚html或模板即可

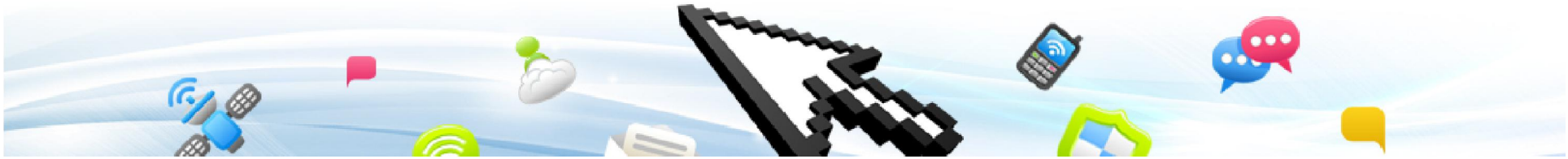


在html中定位资源

fis支持对html中的script、link、style、video、audio、embed等标签的src、data-src或href属性进行分析，一旦这些标签的资源定位属性可以命中已存在文件，则把命中文件的url路径替换到属性中，同时可保留原来url中的query查询信息。

```
<!--源码：
<img title="百度logo" src="images/logo.gif"/>
编译后-->
<img title="百度logo" src="/static/pic/logo_74e5229.gif"/>

<!--源码：
<link rel="stylesheet" type="text/css" href="demo.css">
编译后-->
<link rel="stylesheet" type="text/css" href="/static/css/demo_7defa41.css">

<!--源码：
<script type="text/javascript" src="demo.js"></script>
编译后-->
<script type="text/javascript" src="/static/js/demo_33c5143.js"></script>
```

https://github.com/fis-dev/fis/wiki/%E5%9C%A8html%E4%B8%AD%E5%AE%9A%E4%BD%8D%E8%B5%84%E6%BA%90

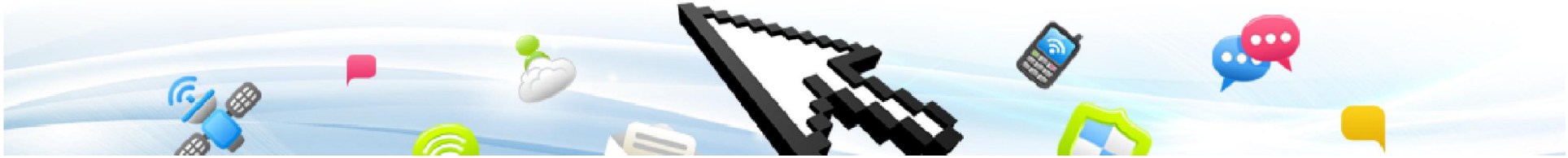


在javascript中定位资源

js语言中，可以使用编译函数 __uri(path) 来定位资源，fis分析js文件或 html中的script标签内内容 时会替换该函数所指向文件的线上url路径。

```
/*源码:
var img = __uri('images/logo.gif');
编译后*/
var img = '/static/pic/logo_74e5229.gif';

/*源码:
var css = __uri('demo.css');
编译后*/
var css = '/static/css/demo_7defa41.css';

/*源码:
var js = __uri("demo.js");
编译后*/
var js = "/static/js/demo_33c5143.js";
```

https://github.com/fis-dev/fis/wiki/%E5%9C%A8javascript%E4%B8%AD%E5%AE%9A%E4%BD%8D%E8%B5%84%E6%BA%90

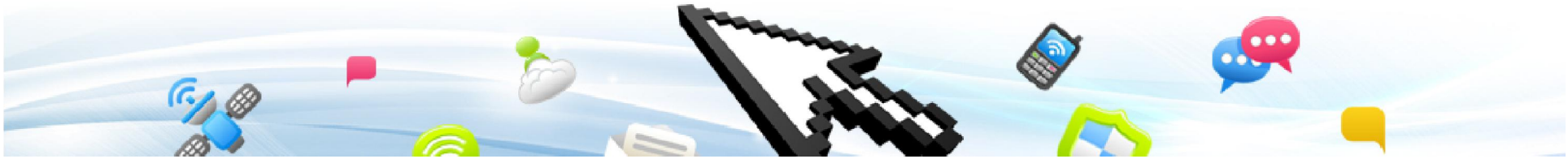


在css中定位资源

fis编译工具会识别css文件或 html的style标签内容 中 url(path) 以及 src=path 字段，并将其替换成对应资源的编译后url路径

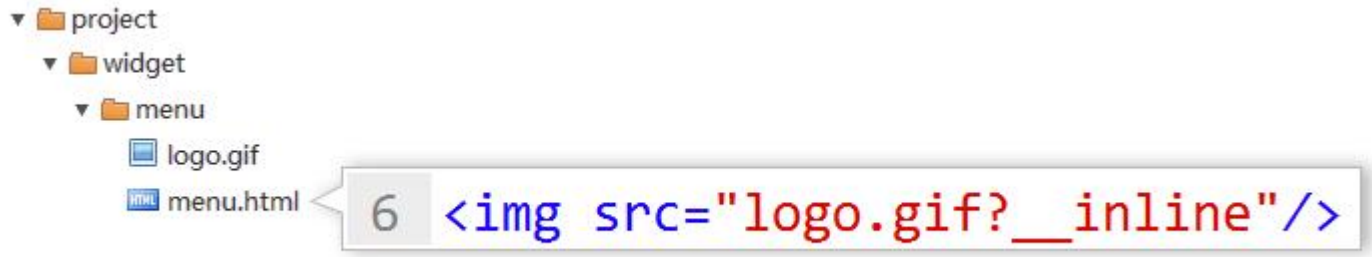
```
/*源码：
@import url('demo.css');
编译后*/
@import url('/static/css/demo_7defa41.css');

.style {
    /*源码：
    background: url('images/body-bg.png');
    编译后*/
    background: url('/static/pic/body-bg_1b8c3e0.png');

    /*源码：
    _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='images/body-bg.png');
    编译后*/
    _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='/static/pic/body-bg_1b8c3e0.png');
}
```

https://github.com/fis-dev/fis/wiki/%E5%9C%A8css%E4%B8%AD%E5%AE%9A%E4%BD%8D%E8%B5%84%E6%BA%90

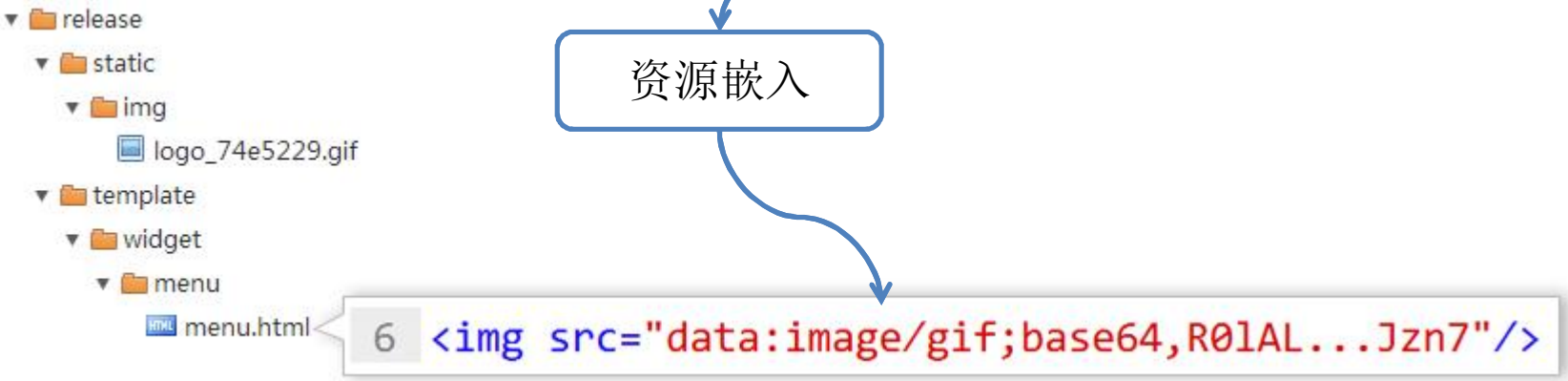


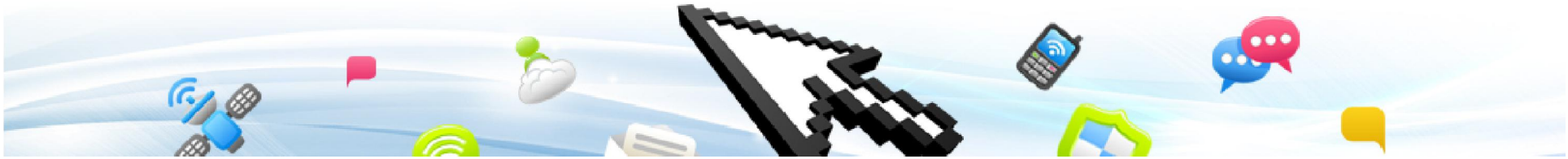
资源嵌入

✓ 开发中：


project
widget
menu
logo.gif
menu.html
6

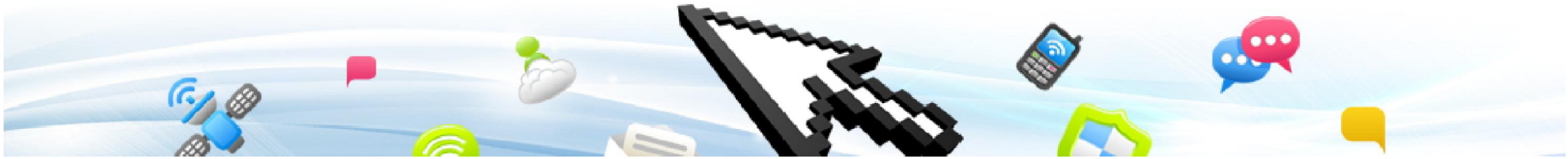
✓ 上线后：


release
static
img
logo_74e5229.gif
template
widget
menu
menu.html
资源嵌入
6



资源嵌入

- ✓ 资源嵌入可以为工程师提供诸如将图片base64嵌入到css、js里，将前端模板编译到js文件中，或将js、css、html拆分成几个文件最后合并到一起的能力。有了这项能力，可以有效的减少http请求数，提升工程的可维护性。
  - *fis不建议用户使用资源嵌入能力作为组件化拆分的手段。*



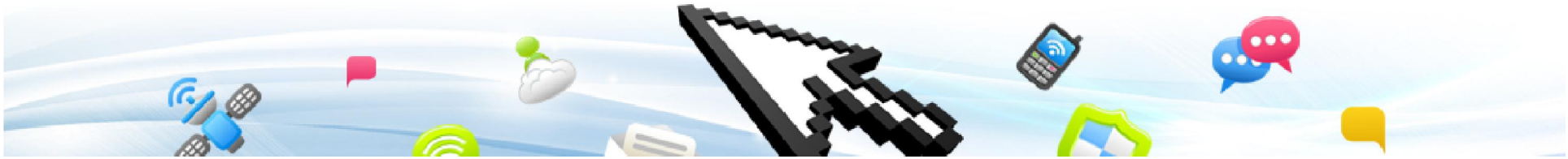
在html中嵌入资源

```
<!--源码：
<img title="百度logo" src="images/logo.gif?__inline"/>
编译后-->
<img title="百度logo" src="...Jzna6853wjKc850nPeoYgAgA7"/>

<!--源码：
<link rel="stylesheet" type="text/css" href="demo.css?__inline">
编译后-->
<style>img { border: 5px solid #ccc; }</style>

<!--源码：
<script type="text/javascript" src="demo.js?__inline"></script>
编译后-->
<script type="text/javascript">console.log('inline file');</script>
```

```
<!--inline[demo.html]-->
```

```
<!-- this is the content of demo.html -->
<h1>demo.html content</h1>
```

https://github.com/fis-dev/fis/wiki/%E5%9C%A8html%E4%B8%AD%E5%B5%8C%E5%85%A5%E8%B5%84%E6%BA%90

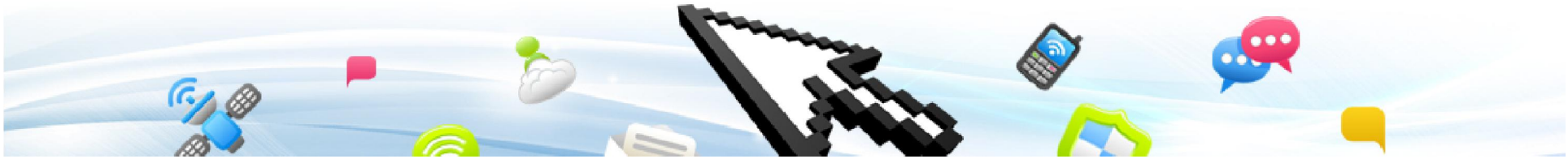


在javascript中嵌入资源

在js中，使用编译函数 __inline() 来提供内容嵌入能力。可以利用这个函数嵌入图片的base64编码、嵌入其他js或者前端模板文件的编译内容， **这些处理对html中script标签里的内容同样有效**。

```
/*源码：
__inline('demo.js');
编译后*/
console.log('demo.js content');

/*源码：
var img = __inline('images/logo.gif');
编译后*/
var img = '...Jzna6853wjKc850nPeoYgAgA7';
```

https://github.com/fis-dev/fis/wiki/%E5%9C%A8javascript%E4%B8%AD%E5%B5%8C%E5%85%A5%E8%B5%84%E6%BA%90

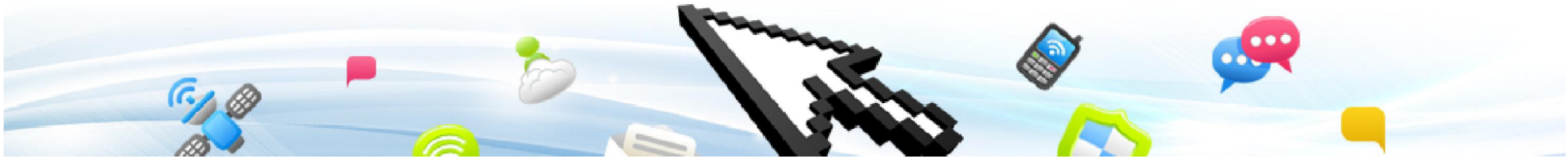


在css中嵌入资源

与html类似，凡是命中了资源定位能力的编译标记，**除了src="xxx"之外**，都可以通过添加?__inline 编译标记都可以把文件内容嵌入进来。src="xxx"被用在ie浏览器支持的filter内，该属性不支持base64字符串，因此未做处理。

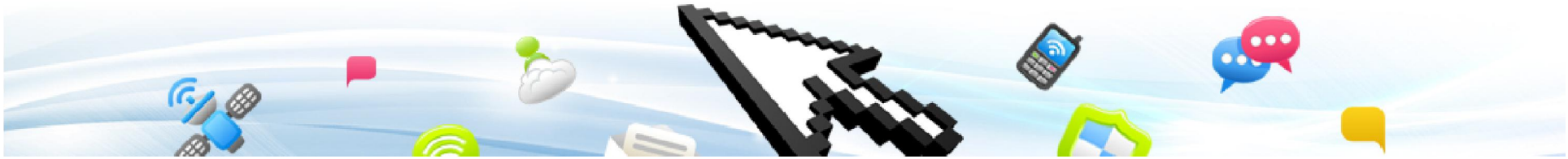
```
/*源码:
@import url('demo.css?__inline');
编译后*/
img { border: 5px solid #ccc; };

.style {
    /*源码:
    background: url('images/logo.gif?__inline');
    编译后*/
    background: url('...Jzna6853wjKc850nPeoYgAgA7);
}
```

https://github.com/fis-dev/fis/wiki/%E5%9C%A8css%E4%B8%AD%E5%B5%8C%E5%85%A5%E8%B5%84%E6%BA%90



依赖声明

- ✓ 依赖声明能力为工程师提供了声明依赖关系的编译接口。
  - fis在执行编译的过程中，会扫描这些编译标记，从而建立一张 **静态资源关系表**，它在编译阶段最后会被产出为一份 map.json 文件，这份文件详细记录了项目内的静态资源id、发布后的线上路径、资源类型以及**依赖关系** 和 **资源打包**等信息。使用fis作为编译工具的项目，可以将这张表提交给后端或者前端框架去运行时根据组件使用情况来 **按需加载资源或者资源所在的包**，从而提升前端页面运行性能。



依赖声明

- ✓ 在html中声明依赖

```
<!--
    @require demo.js
    @require "demo.css"
-->
```

- ✓ 在javascript中声明依赖

```
//demo.js
/**
 * @require demo.css
 * @require list.js
 */
var $ = require('jquery');
```

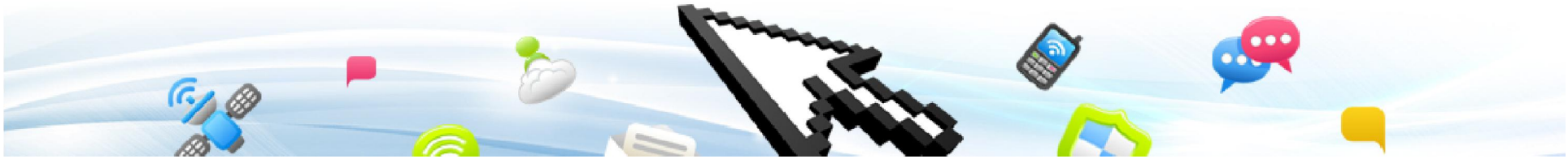
- ✓ 在css中声明依赖

```
/**
 * demo.css
 * @require reset.css
 */
```

- ✓ map.json

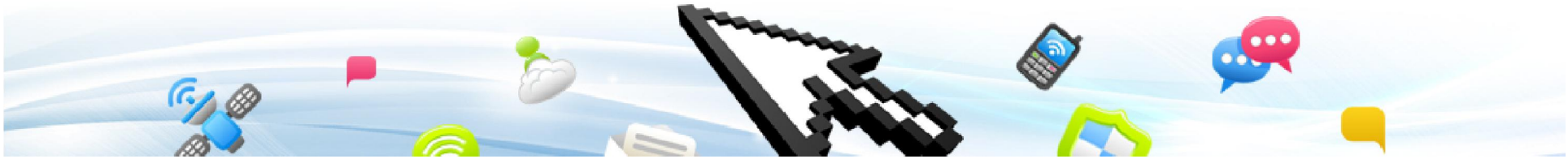
```
{
    "res" : {
        "demo.css" : {
            "uri" : "/static/css/demo_7defa41.css",
            "type" : "css",
            "deps" : [ "reset.css" ]
        },
        "demo.js" : {
            "uri" : "/static/js/demo_33c5143.js",
            "type" : "js",
            "deps" : [ "demo.css" , "list.js" , "jquery" ]
        },
        "index.html" : {
            "uri" : "/index.html",
            "type" : "html",
            "deps" : [ "demo.js", "demo.css" ]
        }
    },
    "pkg" : {}
}
```

https://github.com/fis-dev/fis/wiki#-1



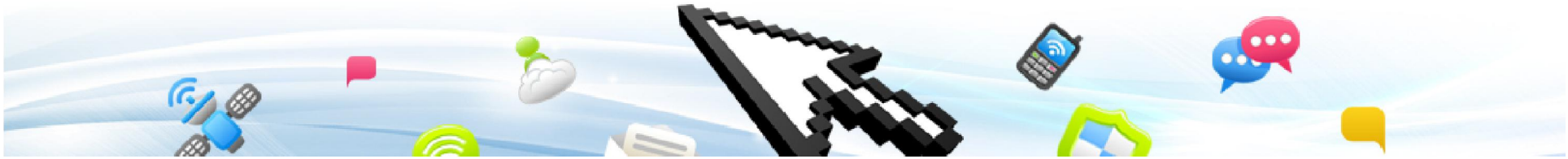
利用三种语言能力解决问题

- ✓ 资源定位解决：
  - 开发与部署目录结构映射的问题
  - 资源添加md5戳的问题
  - 使用domain分发静态资源的问题
- ✓ 内容嵌入解决：
  - 初等资源合并需求
  - 图片的base64嵌入需求
- ✓ 依赖声明解决：
  - 高级资源管理需求（辅助解决，需要前端或后端框架配合）



Baidu 百度

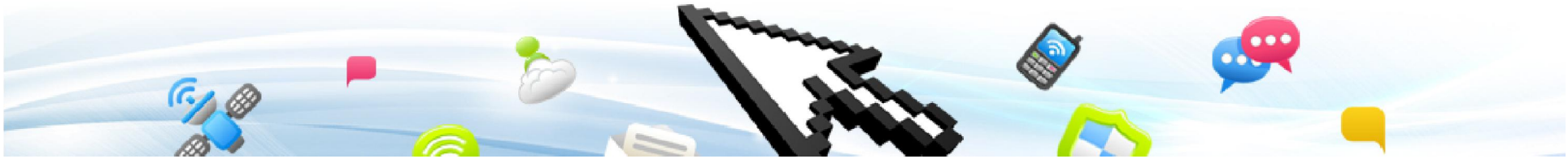
目录



举个小栗子

```
<html>
    <link href="A.css">
    <link href="B.css">
    <link href="C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

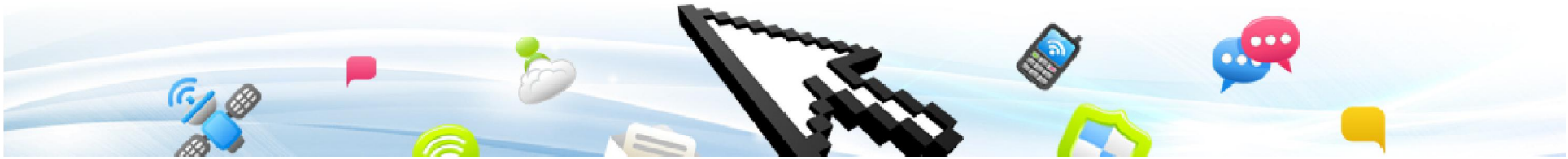
https://github.com/fis-dev/fis/wiki/基于map.json的前后端架构设计指导



举个小栗子

```
<html>
    <link href="A.css">
    <link href="B.css">
    <link href="C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```



举个小栗子

```
<html>
    <link href="A.css">
    <link href="B.css">
    <link href="C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    {if $user_has_C}
        <div>html of C</div>
    {/if}
</html>
```

举个小栗子

```
<html>
    <link href="A.css">
    <link href="B.css">
    <link href="C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    {if $user_has_C}
        <div>html of C</div>
    {/if}
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    {*if $user_has_C}
        <div>html of C</div>
    {/if*}
</html>
```

举个小栗子

```
<html>
    <link href="A.css">
    <link href="B.css">
    <link href="C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    {if $user_has_C}
        <div>html of C</div>
    {/if}
</html>
```

```
<html>
    <link href="A-B-C-D-E-F-G-H....css">
    <div>html of A</div>
    <div>html of B</div>
    ...
    {if $not_used_F}
        <div>html of E</div>
    {else}
        <div>html of F</div>
        <div>html of G</div>
    {/if}
    ...
</html>
```

```
<html>
    <link href="A-B-C.css">
    <div>html of A</div>
    <div>html of B</div>
    {*if $user_has_C}
        <div>html of C</div>
    {/if*}
</html>
```

fis的静态资源管理方案

```
<html>
    <link href="A.css">
    <link href="B.css">
    <link href="C.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

① 组件化拆分

```
<html>
{widget name="A"}
{widget name="B"}
{widget name="C"}
</html>
```

③ 模板运行时读取

```
根目录
  ├ index.tpl
  ├ A
  |  ├ A.tpl
  |  └ A.css
  ├ B
  |  ├ B.tpl
  |  └ B.css
  └ C
     ├ C.tpl
     └ C.css
```

② fis编译生成资源表

```
{
    "res" : {
        "A/A.tpl" : {
            "uri" : "/template/A.tpl",
            "deps" : [ "A/A.css" ]
        },
        "A/A.css" : {
            "uri" : "/static/css/A_7defa41.css"
        },

        "B/B.tpl" : {
            "uri" : "/template/B.tpl",
            "deps" : [ "B/B.css" ]
        },
        "B/B.css" : {
            "uri" : "/static/css/B_33c5143.css"
        },

        "C/C.tpl" : {
            "uri" : "/template/C.tpl",
            "deps" : [ "C/C.css" ]
        },
        "C/C.css" : {
            "uri" : "/static/css/C_ba59c31.css"
        }
    }
}
```

✓ {widget name="A"}

```
uris = [
    '/static/css/A_7defa41.css'
];
has = {
    'A/A.css' : true
};
```

```
{
    "res" : {
        "A/A.tpl" : {
            "uri" : "/template/A.tpl",
            "deps" : [ "A/A.css" ]
        },
        "A/A.css" : {
            "uri" : "/static/css/A_7defa41.css"
        },

        "B/B.tpl" : {
            "uri" : "/template/B.tpl",
            "deps" : [ "B/B.css" ]
        },
        "B/B.css" : {
            "uri" : "/static/css/B_33c5143.css"
        },

        "C/C.tpl" : {
            "uri" : "/template/C.tpl",
            "deps" : [ "C/C.css" ]
        },
        "C/C.css" : {
            "uri" : "/static/css/C_ba59c31.css"
        }
    }
}
```

✓ {widget name="A"}

```
uris = [
    '/static/css/A_7defa41.css'
];
has = {
    'A/A.css' : true
};
```

✓ {widget name="B"}

```
uris = [
    '/static/css/A_7defa41.css',
    '/static/css/B_33c5143.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true
};
```

```
{
    "res" : {
        "A/A.tpl" : {
            "uri" : "/template/A.tpl",
            "deps" : [ "A/A.css" ]
        },
        "A/A.css" : {
            "uri" : "/static/css/A_7defa41.css"
        },

        "B/B.tpl" : {
            "uri" : "/template/B.tpl",
            "deps" : [ "B/B.css" ]
        },
        "B/B.css" : {
            "uri" : "/static/css/B_33c5143.css"
        },

        "C/C.tpl" : {
            "uri" : "/template/C.tpl",
            "deps" : [ "C/C.css" ]
        },
        "C/C.css" : {
            "uri" : "/static/css/C_ba59c31.css"
        }
    }
}
```

静态资源管理系统

✓ {widget name="A"}

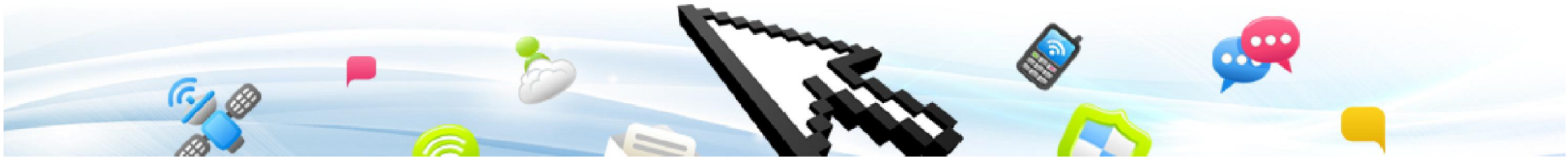
```
uris = [
    '/static/css/A_7defa41.css'
];
has = {
    'A/A.css' : true
};
```

✓ {widget name="B"}

```
uris = [
    '/static/css/A_7defa41.css',
    '/static/css/B_33c5143.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true
};
```

✓ {widget name="C"}

```
uris = [
    '/static/css/A_7defa41.css',
    '/static/css/B_33c5143.css',
    '/static/css/C_ba59c31.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```



✓ {widget name="A"}

```
uris = [
    '/static/css/A_7defa41.css'
];
has = {
    'A/A.css' : true
};
```

✓ {widget name="B"}

```
uris = [
    '/static/css/A_7defa41.css',
    '/static/css/B_33c5143.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true
};
```

✓ {widget name="C"}

```
uris = [
    '/static/css/A_7defa41.css',
    '/static/css/B_33c5143.css',
    '/static/css/C_ba59c31.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```

✓ 页面输出

```
<html>
    <link href="/static/css/A_7defa41.css">
    <link href="/static/css/B_33c5143.css"
    <link href="/static/css/C_ba59c31.css"
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```

```
fis.config.merge({
    pack : {
        '/static/pkg/aio.css' : '**.css'
    }
});
```

```
{
    "res" : {
        "A/A.tpl" : {
            "uri" : "/template/A.tpl",
            "deps" : [ "A/A.css" ]
        },
        "A/A.css" : {
            "uri" : "/static/css/A_7defa41.css",
            "pkg" : "p0"
        },

        "B/B.tpl" : {
            "uri" : "/template/B.tpl",
            "deps" : [ "B/B.css" ]
        },
        "B/B.css" : {
            "uri" : "/static/css/B_33c5143.css",
            "pkg" : "p0"
        },

        "C/C.tpl" : {
            "uri" : "/template/C.tpl",
            "deps" : [ "C/C.css" ]
        },
        "C/C.css" : {
            "uri" : "/static/css/C_ba59c31.css",
            "pkg" : "p0"
        }
    },
    "pkg" : {
        "p0" : {
            "uri" : "/static/pkg/aio_0cb4a19.css",
            "has" : [ "A/A.css", "B/B.css", "C/C.css" ]
        }
    }
}
```

✓ {widget name="A"}

```
uris = [
    '/static/pkg/aio_0cb4a19.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```

```
{
    "res" : {
        "A/A.tpl" : {
            "uri" : "/template/A.tpl",
            "deps" : [ "A/A.css" ]
        },
        "A/A.css" : {
            "uri" : "/static/css/A_7defa41.css",
            "pkg" : "p0"
        },

        "B/B.tpl" : {
            "uri" : "/template/B.tpl",
            "deps" : [ "B/B.css" ]
        },
        "B/B.css" : {
            "uri" : "/static/css/B_33c5143.css",
            "pkg" : "p0"
        },

        "C/C.tpl" : {
            "uri" : "/template/C.tpl",
            "deps" : [ "C/C.css" ]
        },
        "C/C.css" : {
            "uri" : "/static/css/C_ba59c31.css",
            "pkg" : "p0"
        }
    },
    "pkg" : {
        "p0" : {
            "uri" : "/static/pkg/aio_0cb4a19.css",
            "has" : [ "A/A.css", "B/B.css", "C/C.css" ]
        }
    }
}
```

✓ {widget name="A"}

```
uris = [
    '/static/pkg/aio_0cb4a19.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```

✓ {widget name="B"}

```
uris = [
    '/static/pkg/aio_0cb4a19.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```

```
{
    "res" : {
        "A/A.tpl" : {
            "uri" : "/template/A.tpl",
            "deps" : [ "A/A.css" ]
        },
        "A/A.css" : {
            "uri" : "/static/css/A_7defa41.css",
            "pkg" : "p0"
        },

        "B/B.tpl" : {
            "uri" : "/template/B.tpl",
            "deps" : [ "B/B.css" ]
        },
        "B/B.css" : {
            "uri" : "/static/css/B_33c5143.css",
            "pkg" : "p0"
        },

        "C/C.tpl" : {
            "uri" : "/template/C.tpl",
            "deps" : [ "C/C.css" ]
        },
        "C/C.css" : {
            "uri" : "/static/css/C_ba59c31.css",
            "pkg" : "p0"
        }
    },
    "pkg" : {
        "p0" : {
            "uri" : "/static/pkg/aio_0cb4a19.css",
            "has" : [ "A/A.css", "B/B.css", "C/C.css" ]
        }
    }
}
```

✓ {widget name="A"}

```
uris = [
    '/static/pkg/aio_0cb4a19.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```

✓ {widget name="C"}

```
uris = [
    '/static/pkg/aio_0cb4a19.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```

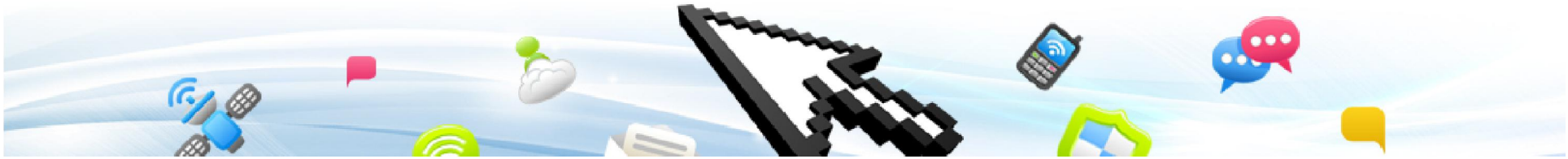
✓ {widget name="B"}

```
uris = [
    '/static/pkg/aio_0cb4a19.css'
];
has = {
    'A/A.css' : true,
    'B/B.css' : true,
    'C/C.css' : true
};
```

✓ 页面输出

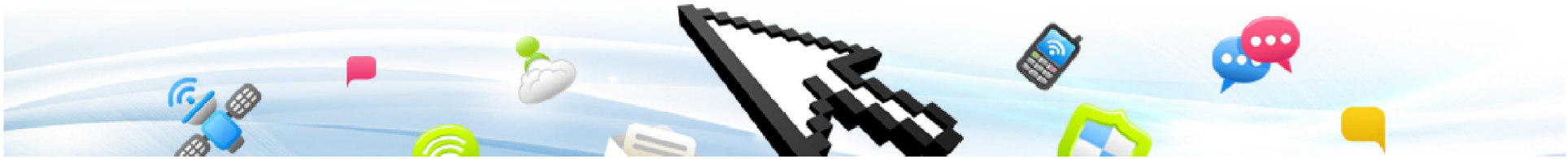
出现打包资源啦

```
<html>
    <link href="/static/pkg/aio_0cb4a19.css">
    <div>html of A</div>
    <div>html of B</div>
    <div>html of C</div>
</html>
```



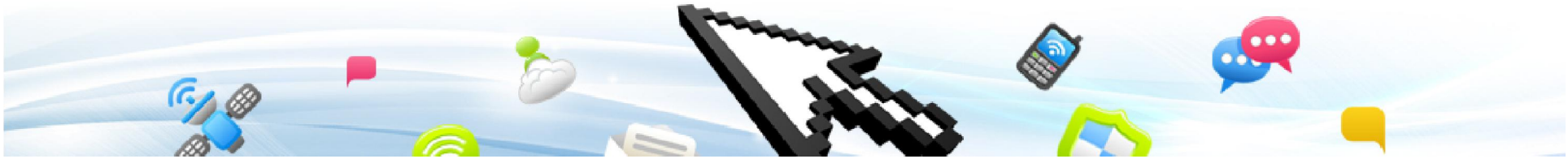
这样做的好处

- ✓ 抱歉，这货好处实在太多了。
- ✓ 已知的：
  - 统计 {widget}调用情况，自动生成最优配置，让网站可以 自适应优化
  - 所有 资源定位 的事情，都交给fis好了。
  - 资源都按需加载
  - 静态资源服务器可以放心开启强缓存
  - 控制线上的页面输出结果， 方便定位线上问题
  - 切换输出域名 把线上资源映射到本地
  - 根据终端类型 将不同的资源送达给不同的用户
  - 注册异步请求的资源，加快首屏渲染速度
  - 控制资源的输出方式，实现pipe_line、quickling等输出模式
  - fis只指导后端模板渲染框架的实现，但不强制要求这种实现
  - map表以json格式输出，支持各种平台，也可由纯前端来读取map表实现资源管理
- ✓ 更多用法有待挖掘

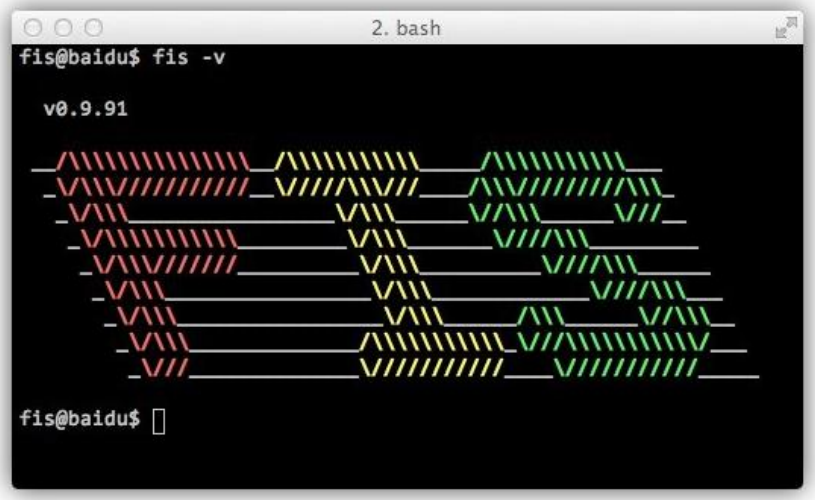


Baidu 百度

目录



- ✓ npm install –g fis
  - 环境要求：
    - 操作系统：任何能安装 nodejs 的操作系统
    - node版本：>= v0.8.0
    - jre版本：>= v1.5.0 【如果不需要本地调试服务器，可以忽略java环境要求】
    - php-cgi版本：>= v5.0.0 【如果不需要本地调试服务器，可以忽略php-cgi环境要求】
- ✓ fis -v

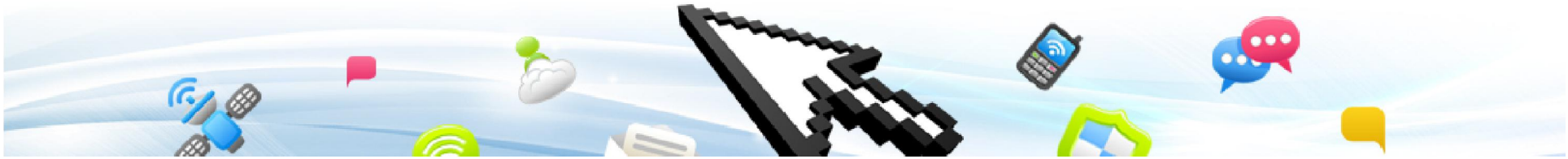


https://github.com/fis-dev/fis/wiki/%E5%BF%AB%E9%80%9F%E4%B8%8A%E6%89%8B

三条命令，满足你的所有开发需求

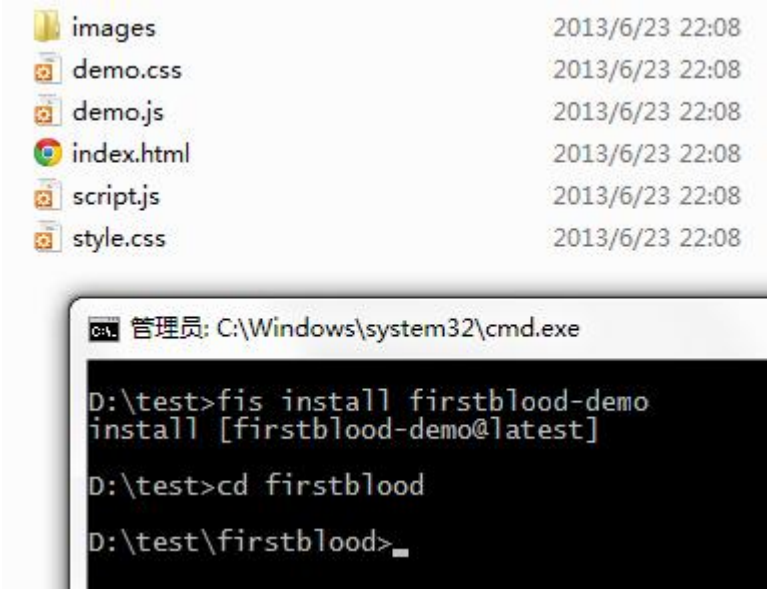
- ✓ fis release： 编译并发布你的项目
- ✓ fis install：安装fis仓库提供的各种 组件，框架，库，样例项目，甚至配置文件 等模块
- ✓ fis server：启动一个 1.8M 大小的内置调试服务器，它采用php-java-bridge技术实现， 依赖java、php-cgi外部环境 ，可以完美支持运行php程序。

```
Usage: fis <command>

Commands:

  release     build and deploy your project
  install     install components and demos
  server      launch a php-cgi server

Options:

  -h, --help     output usage information
  -v, --version  output the version number
  --no-color     disable colored output
```

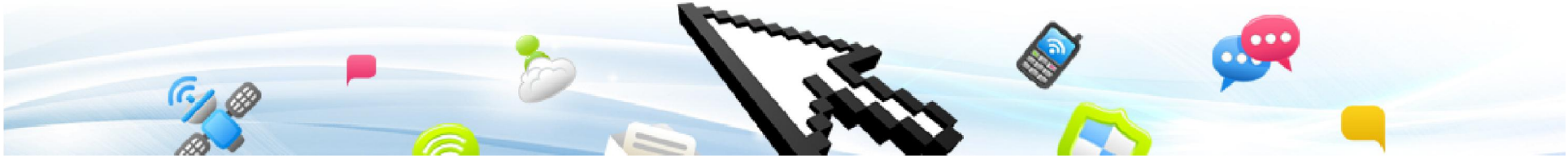


fis install <name>

- ✓ install命令被设计用来 **各种安装**，无论你是想初始化一个模块，还是想下载一个前端基础库，亦或下载一份配置文件，总之但凡开发需要的，只要fis仓库里有，你就用它来安装就对了。 理论上任何资源都可以通过这个命令来获取，因为它的实现非常简单： **从fis代码仓库下载->解压到当前目录。**

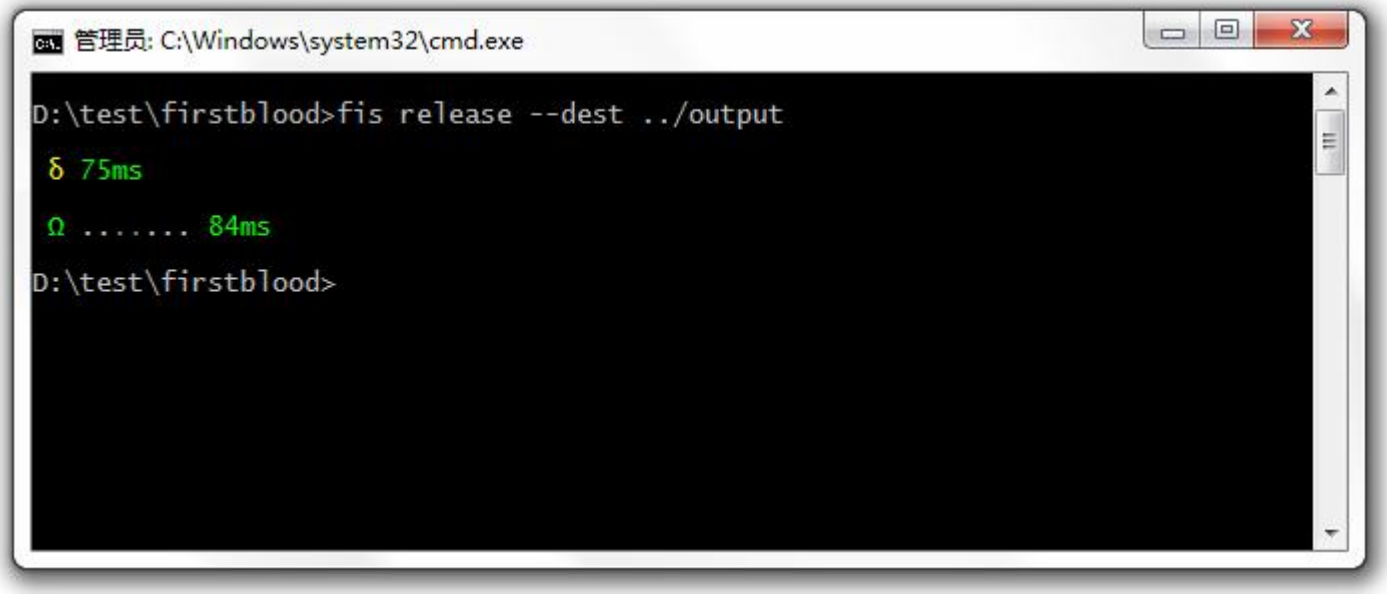


https://github.com/fis-dev/fis/wiki/%E5%BF%AB%E9%80%9F%E4%B8%8A%E6%89%8B#fis-install-name



fis release [options]

- ✓ release是一个非常强大的命令，它的主要任务就是进行代码的 **编译** 与 **部署**，它的参数囊括了前端开发所需的各种基础功能

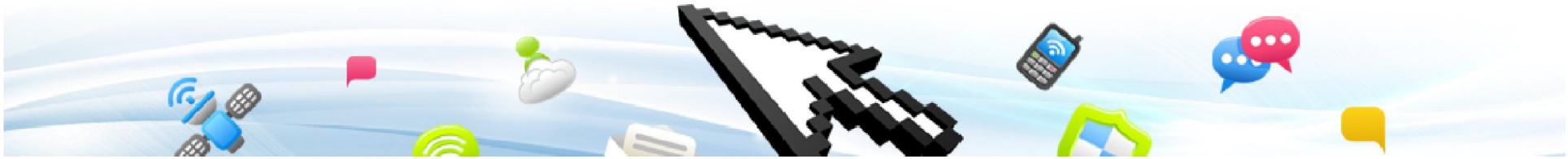


https://github.com/fis-dev/fis/wiki/%E5%BF%AB%E9%80%9F%E4%B8%8A%E6%89%8B#fis-release-options

fis release [options]

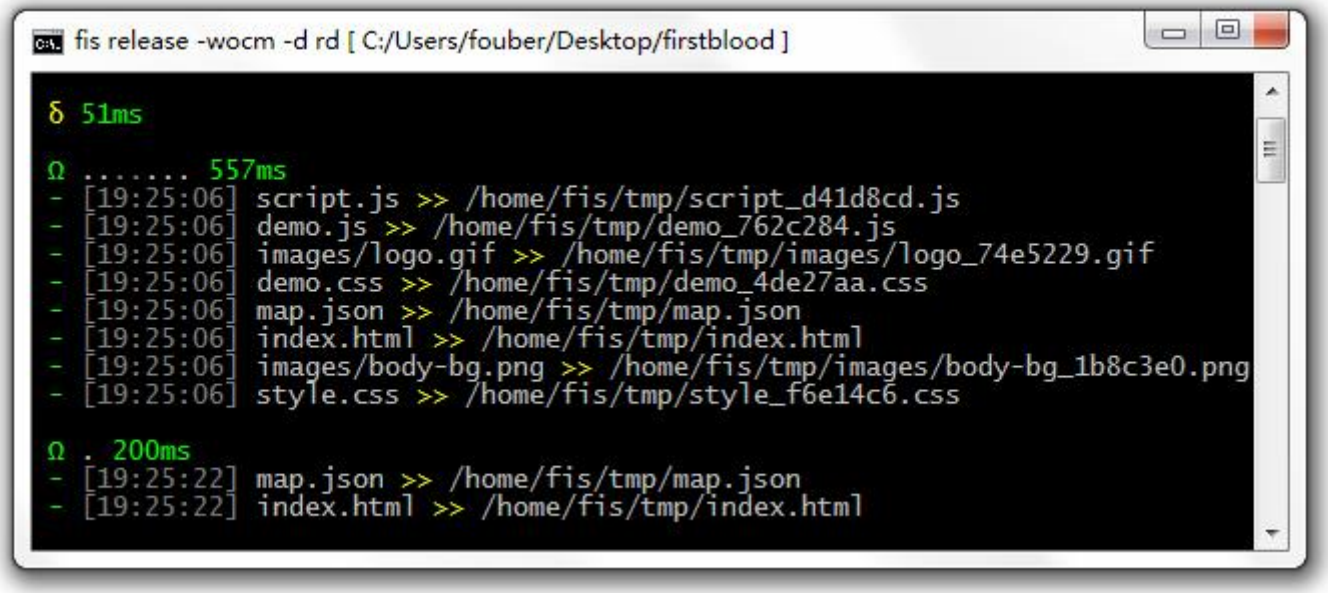
- ✓ 压缩、打包、添加md5、添加domain、测试、校验
  - fis release --optimize --pack --md5 --domains --test --lint --dest ../output
  - 或者 fis release -opmDtld ../output
- ✓ 监听文件变化
  - fis release --watch --dest ../output
  - 或者 fis release -wd ../output
- ✓ 自动刷新浏览器
  - fis release --live --dest ../output
  - 或者 fis release -Ld ../output
- ✓ 文件上传
  - fis release --dest remote
  - fis release --dest remote,output

```
1  fis.config.merge({
2      deploy : {
3          'remote' : {
4              receiver : 'http://www.example.com/receiver.php',
5              to : '/home/username/path/to/save'
6          }
7      }
8  });
```

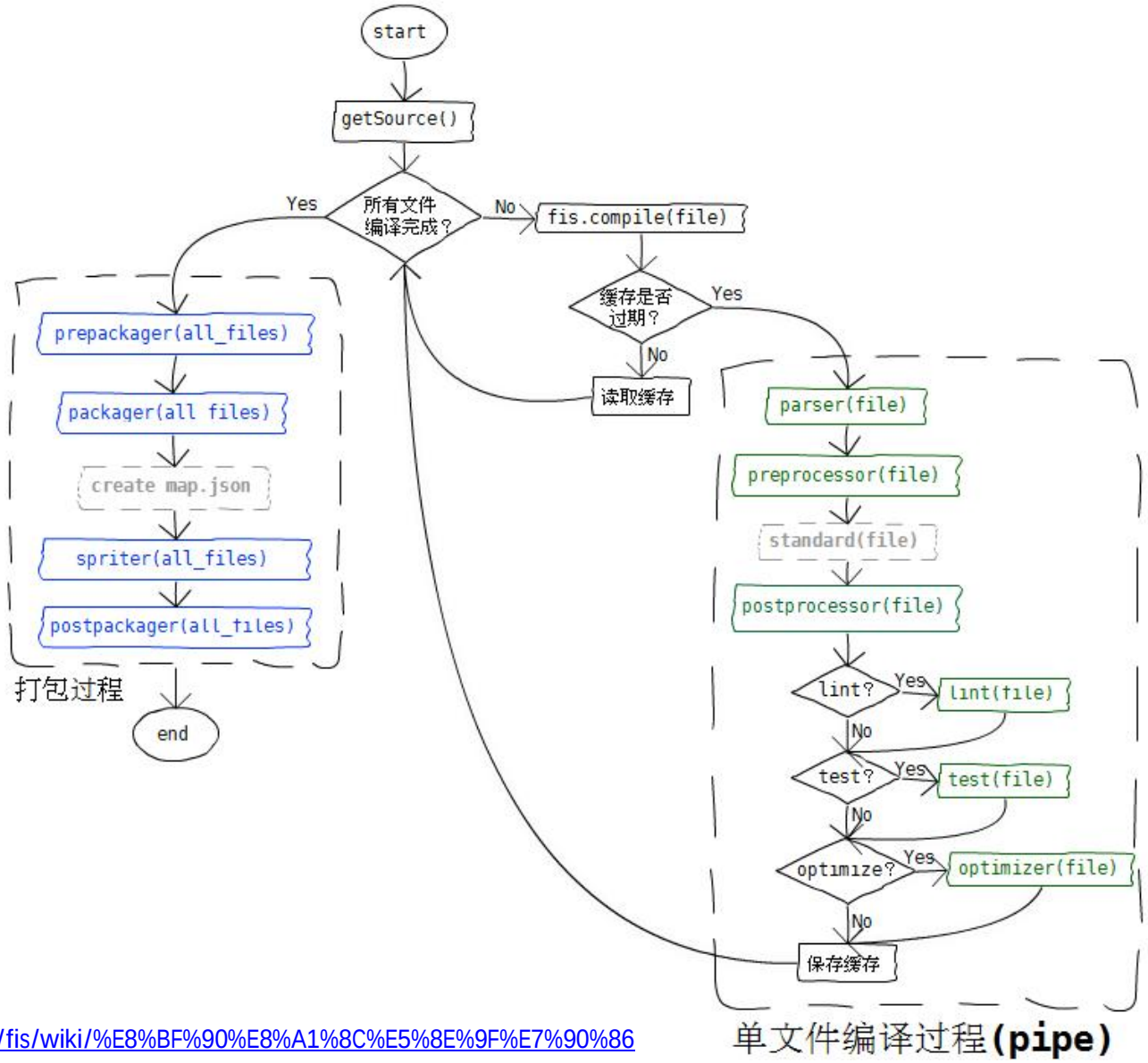


fis release [options]

- ✓ 文件监听+自动刷新+优化+md5+domain+测试+校验+打包+多机器上传
  - fis release -wLomDtlp -d rd,qa,output

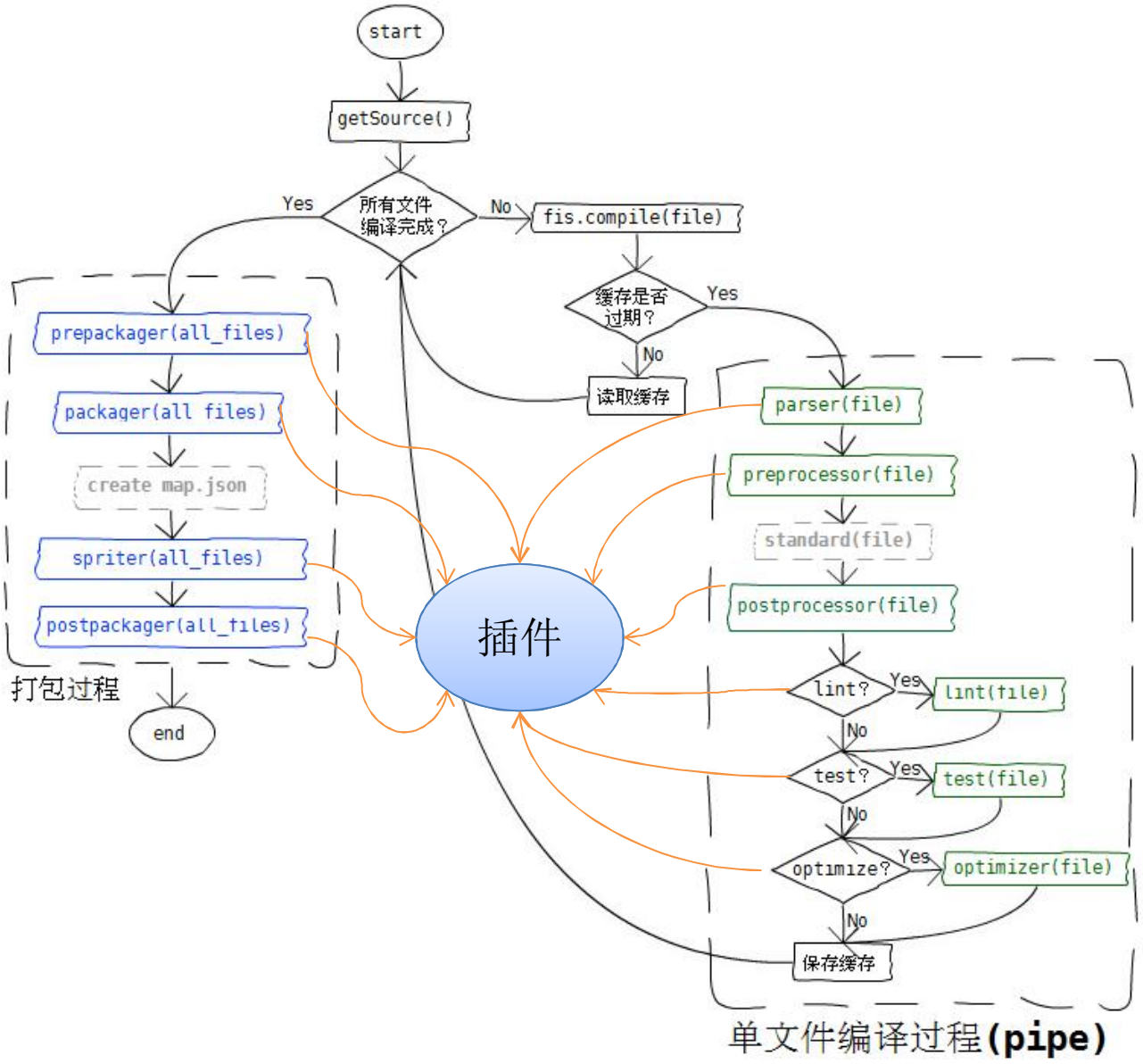


fis release的工作原理



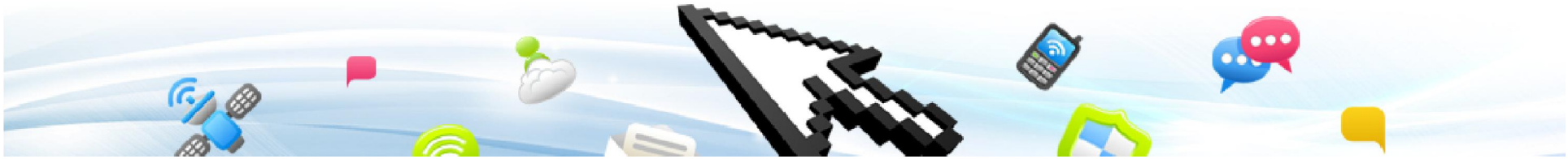
https://github.com/fis-dev/fis/wiki/%E8%BF%90%E8%A1%8C%E5%8E%9F%E7%90%86

fis release的工作原理



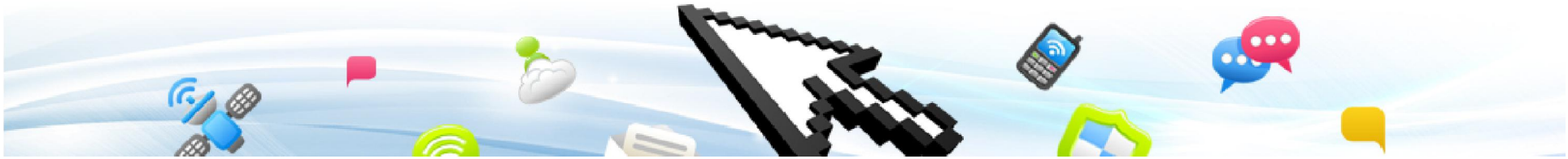
插件调用机制

https://github.com/fis-dev/fis/wiki/%E6%8F%92%E4%BB%B6%E8%B0%83%E7%94%A8%E6%9C%BA%E5%88%B6



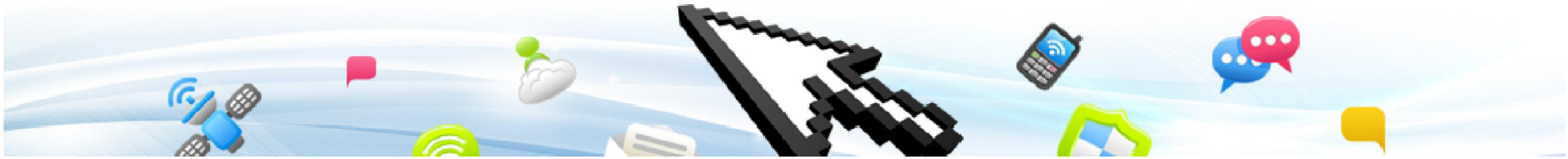
插件调用机制


fis-parser-coffee-script
fis
node_modules
fis-command-install
fis-command-release
fis-command-server
fis-kernel
lib
node_modules
fis-kernel.js
fis-optimizer-clean-css
fis-optimizer-html-minifier
fis-optimizer-uglify-js
fis-postprocessor-jswrapper
fis.js
/*
* fis
* http://fis.baidu.com/
*/
var fis = require('fis');
fis.require('parser', 'coffee-script');
fis.require('command', 'release');



fis编译系统的三种扩展方式

- ✓ 使用fis的用户，自己需要某种插件，可以在fis安装目录的 **同级**，安装自己扩展的插件。比如：
  - $ npm install -g fis
  - $ npm install -g fis-parser-coffee-script
- ✓ fis团队会衡量某个插件的通用性，把它放到fis的依赖里，最优先加载。目前已经内置的插件包括：
  - fis-kernel：fis编译机制内核
  - fis-command-release：fis release命令的提供者，处理编译过程，并提供文件监听、自动上传等功能
  - fis-command-install：fis install命令的提供者，用于从fis仓库下载组件、配置、框架、素材等资源
  - fis-command-server：fis server命令的提供者，用于开启一个本地php-cgi服务器，对项目进行预览、调试。
  - fis-optimizer-uglify-js：fis的优化插件，调用uglify-js对文件内容进行js压缩。
  - fis-optimizer-clean-css：fis的优化插件，调用clean-css对文件内容进行css压缩。
  - fis-optimizer-html-minifier：fis的优化插件，调用html-minifier对文件内容进行html压缩。
  - fis-postprocessor-jswrapper：fis的后处理器插件，用于对js文件进行包装，支持amd的define包装或者匿名自执行函数包装。
- ✓ 开发一个依赖于fis模块的npm包，并在这个包里定制所需要的插件。这种方式与上一条类似，也是将插件安装在fis的同级目录下。



Bai du 百度

插件使用配置

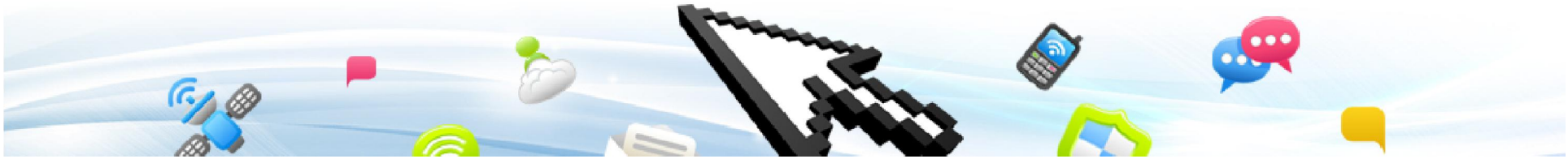
```
//fis-conf.js
fis.config.merge({
    modules : {
        parser : {
            //coffee后缀的文件使用fis-parser-coffee-script插件编译
            coffee : 'coffee-script',
            //less后缀的文件使用fis-parser-less插件编译
            //处理器支持数组，或者逗号分隔的字符串配置
            less : ['less'],
            //md后缀的文件使用fis-parser-marked插件编译
            md : 'marked'
        }
    },
    roadmap : {
        ext : {
            //less后缀的文件将输出为css后缀
            //并且在parser之后的其他处理流程中被当做css文件处理
            less : 'css',
            //coffee后缀的文件将输出为js文件
            //并且在parser之后的其他处理流程中被当做js文件处理
            coffee : 'js',
            //md后缀的文件将输出为html文件
            //并且在parser之后的其他处理流程中被当做html文件处理
            md : 'html'
        }
    }
});
```

fis系统的扩展点

✓ 单文件编译扩展

- parser(编译器插件)
  - fis-parser-coffee-script：把coffee-script编译成js
  - fis-parser-bdtmpl：使用baiduTemplate将前端模板文件编译成js
  - fis-parser-less：将less文件编译成css
  - fis-parser-marked：把markdown文件编译成html
  - fis-parser-utc：把underscore前端模板编译成js
- preprocessor(标准预处理器插件)
  - fis-preprocessor-image-set：如果css中使用的背景图比如a.png，旁边有一个a_2x.png文件，则将图片的背景设置为- webkit-image-set形式。此功能为retina屏适配项目开发。
- postprocessor(标准后处理器插件)
  - fis-postprocessor- jswrapper：用于对js文件进行amd定义包装。
- lint(校验器插件)
- test(自动测试插件)
- optimizer(代码优化器插件)
  - fis-optimizer-uglify-js：调用uglify-js对js文件进行压缩优化。
  - fis-optimizer-clean-css：调用clean-css对css文件进行压缩优化。
  - fis-optimizer-html-minifier：调用html-minifier对html、htm文件进行压缩优化。

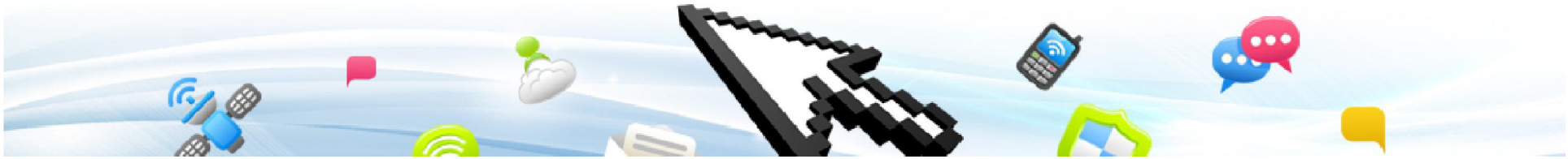
https://github.com/fis-dev/fis/wiki/%E6%8F%92%E4%BB%B6%E6%89%A9%E5%B1%95%E7%82%B9%E5%88%97%E8%A1%A8



fis系统的扩展点

✓ 打包扩展

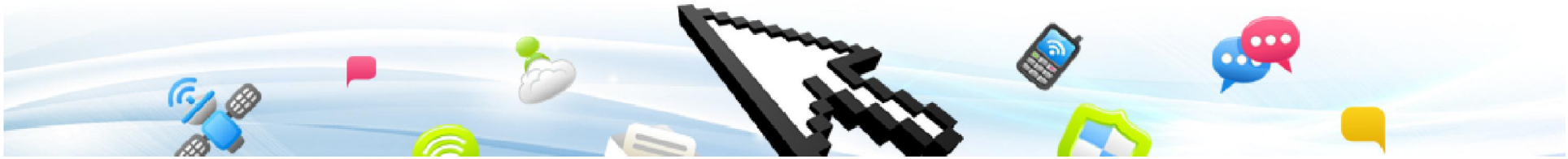
- prepackager(打包预处理器插件)
- packager(打包处理器插件)
  - fis-packager-map：将打包资源输出给map表。
- spriter(sprite处理器插件)
- postpackager(打包后处理器插件)



fis系统的扩展点

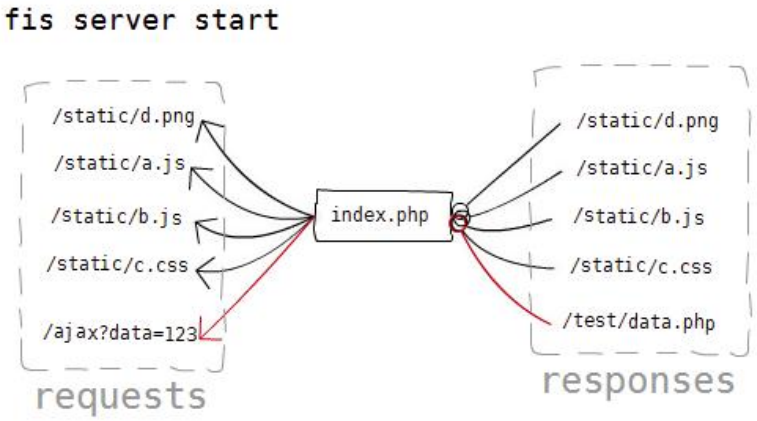
✓ 命令行扩展

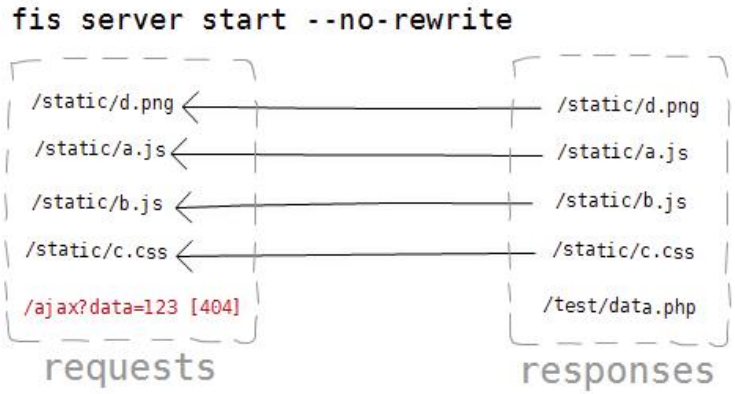
- command(命令行插件)
  - fis-command-release：fis release命令的提供者，处理编译过程，并提供文件监听、自动上传等功能
  - fis-command-install：fis install命令的提供者，用于从fis仓库下载组件、配置、框架、素材等资源
  - fis-command-server：fis server命令的提供者，用于开启一个本地php-cgi服务器，对项目进行预览、调试。



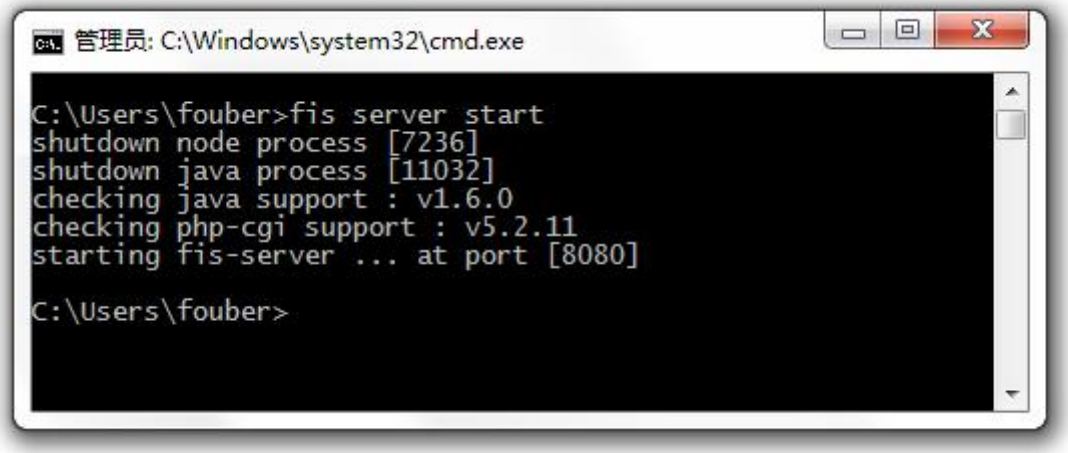
fis server <command> [options]

- ✓ fis server start --no-rewrite
- ✓ fis server install <name>
  - 在server的document root目录下安装调试服务，用于模拟数据、调试页面、控制请求抓发等功能
- ✓ fis release
  - 不使用--dest参数，则指向fis server的document root目录
  - 将文件投送过去之后，刷新浏览器，查看运行效果



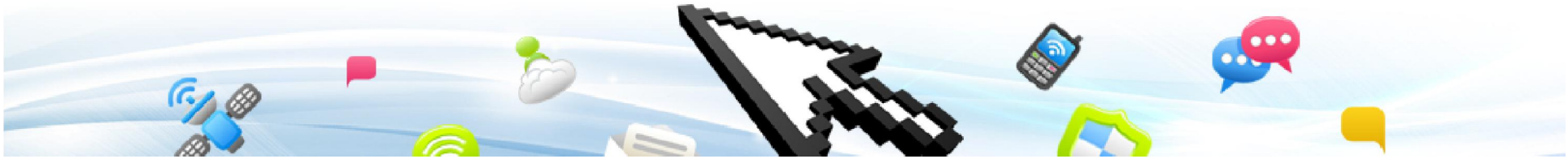


fis server start

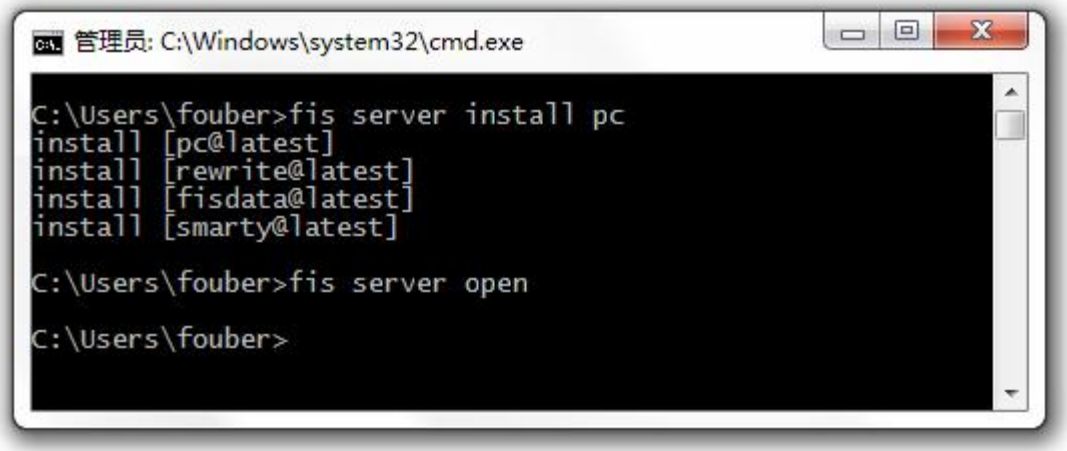


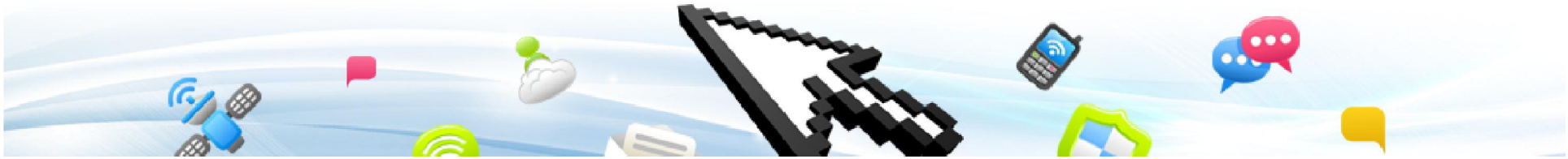
Forward request " / " to file [C:\Users\[redacted]\AppData\Local\.fis-tmp\www\index.php].

https://github.com/fis-dev/fis/wiki/%E5%BF%AB%E9%80%9F%E4%B8%8A%E6%89%8B#fis-server-command-options



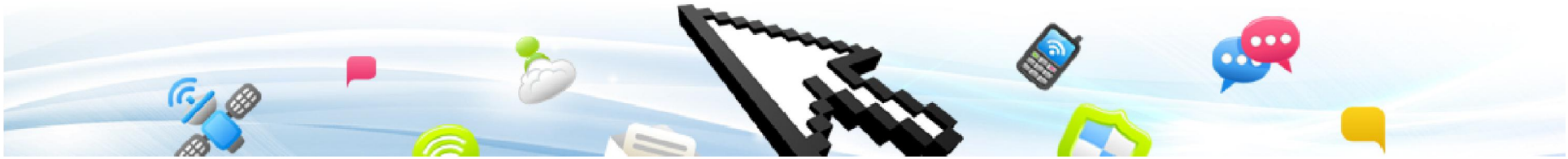
fis server install <name>





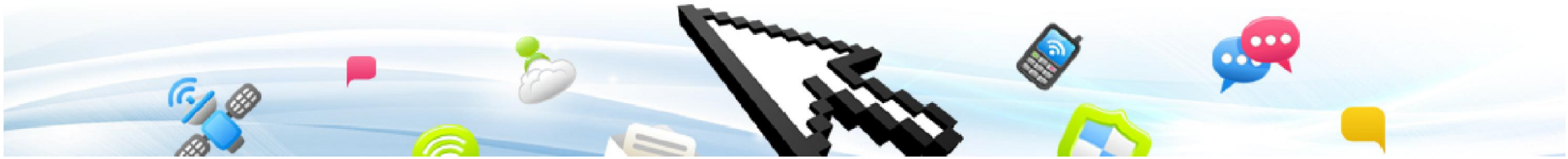
Baidu 百度

目录



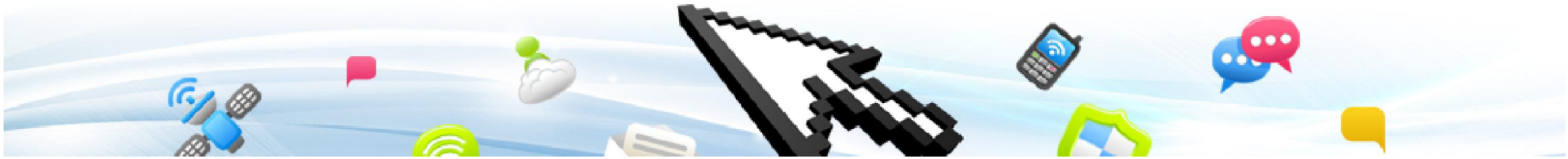
开发资源聚合与分享

- ✓ fis将甄选优秀的前后端框架、示例、配置、组件、素材等资源逐个版本的发布到fis仓库，方便前端工程师通过 fis install 命令一键获取。
- ✓ fis提供 fis-command-publish 插件，喜欢分享自己工作成果的工程师可以安装它，然后通过 fis publish 命令来一键分享
- ✓ 以上两条命令预示着fis将有非常丰富的社区资源。

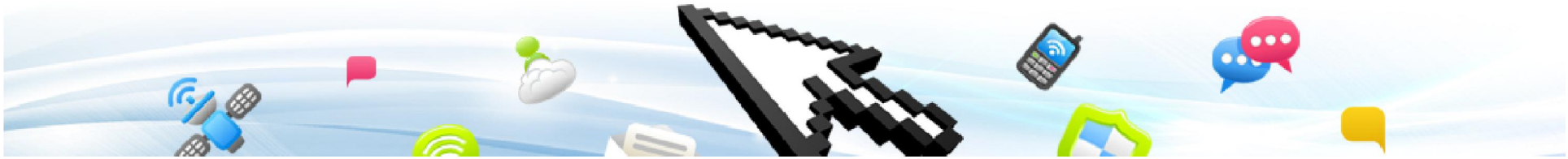


前端编译工具整合

- ✓ 无论多少插件，fis系统对用户使用暴露的是fis release这条命令
- ✓ 使用less、coffee-script等编译工具处理之后，fis系统会做进一步的语言能力扩展，这意味着less、coffee等语言在fis体系内仍然具备fis所提供的三种语言能力【资源定位、资源嵌入、依赖声明】
- ✓ 与现有的编译工具不同，fis针对前端开发任务对插件做了特定的规范，将工具与流程整合，使得工程师在使用中完全感觉不到学习和使用成本，而fis或其他第三方团队可以不断丰富fis的插件系统，从而实现前端工具在fis平台上的聚合。
  - https://npmjs.org/browse/keyword/fis



目录



回到当初——某产品线成长日记

✓ 某团队要做一个全新的互联网产品

- 第一天（规范定制）：
  - 定制必要的开发规范
- 第二天（技术选型）：
  - 选择前端组件化框架（requirejs, seajs, ...）
  - 选择前端基础库（jquery, tangram, ...）
  - 选择前端组件库（jqueryui, magic, ...）
  - 选择模板语言（smarty, php, ...）
  - 选择模板插件（xss-repair）

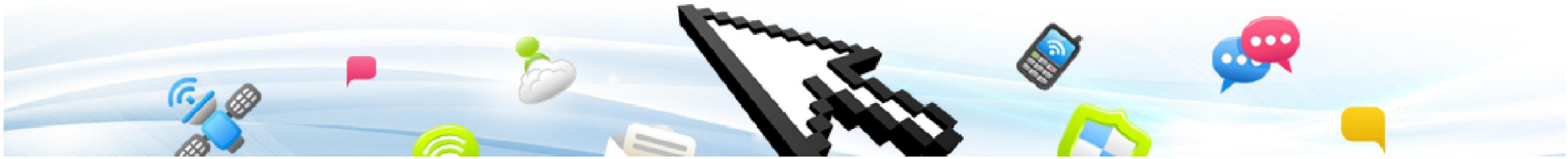
fis install <name>

- 一个月后（自动化与拆分）：
  - 选择或开发自动化工具（编译、打包、压缩、校验）
  - 组件化、模块化拆分系统，便于复用与维护

fis release [options]

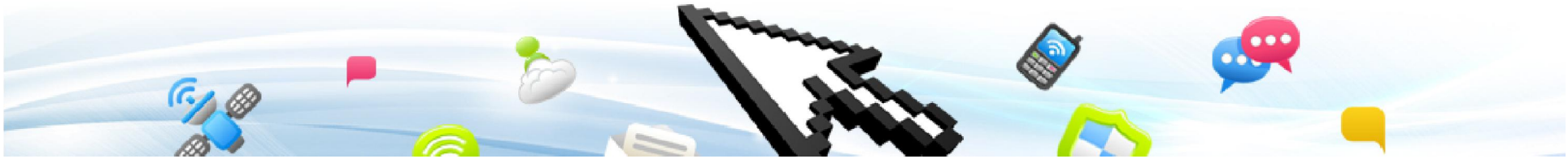
- 半年以后（性能优化）：
  - 小心剔除已下线功能的资源
  - 优化http请求
  - 适当调整工具以适应性能优化
  - 使用架构级优化方案：BigPipe、PageCache、Quickling、BigRender等

map.json + 前后端框架

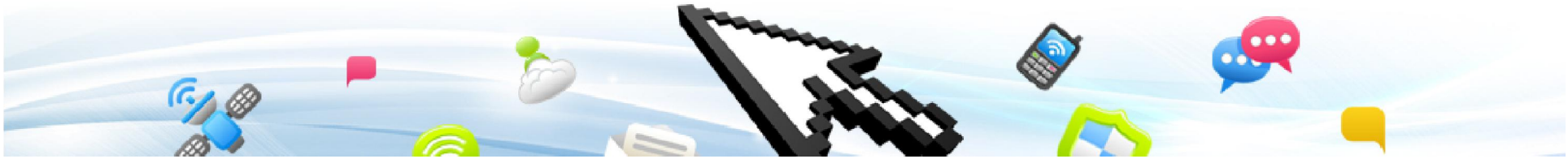


日常开发

- ✓ 本地开发与调试
  - fis server install <something>
  - fis server start
  - fis release --watch
- ✓ 获取资源
  - fis install tangram ./lib
  - fis install slider ./component
- ✓ 玩玩coffee、less，甚至markdown
  - npm install -g fis-parser-coffee-script
  - npm install -g fis-parser-less
  - npm install -g fis-parser-marked
  - vi path/to/project/fis-conf.js
- ✓ 与后端工程师联调
  - fis release --watch --dest rd
- ✓ 提测给qa测试
  - fis release -otImp --dest qa
- ✓ 上线：
  - fis release -otImpD --dest output



谢 谢 ！



百度技术沙龙

关注我们：t.baidu-tech.com

资料下载和详细介绍：infoq.com/cn/zones/baidu-salon

“畅想•交流•争鸣•聚会”是百度技术沙龙的宗旨。 百度技术沙龙是由百度与InfoQ中文站定期组织的线下技术交流活动。目的是让中高端技术人员有一个相对自由的思想交流和交友沟通的的平台。主要分讲师分享和OpenSpace两个关键环节，每期只关注一个焦点话题。

讲师分享和现场Q&A让大家了解百度和其他知名网站技术支持的先进实践经验，OpenSpace环节是百度技术沙龙主题的升华和展开，提供一个自由交流的平台。针对当期主题，参与者人人都可以发起话题，展开讨论。

InfoQ 策划·组织·实施

关注我们：weibo.com/infoqchina