Kissy 模块化实践

承玉
yiminghe@gmail.com
Draft 05.27.2011

大纲

- 0.模块化思想
- 1.前端代码演变
- 2.kissy 模块化实践
  - 2.1 kissy 核心模块
  - 2.2 kissy 业务应用

模块

模块化

- 一种将系统分离成独立功能部分的方法
  - 系统分割成独立功能部分
  - 严格定义模块接口
  - 模块间具备透明性
- 优缺点?

- 优点：
  - 可维护性
    - 灵活架构，焦点分离
    - 方便模块间组合，分解
    - 方便单个模块功能调试，升级
    - 多人协作互不干扰
  - 可测试性
- 缺点
  - 性能损耗
    - 系统分层
    - 模块间通信

语言层面支持

- Pascal
- Erlang
- Perl
- Python
- Ruby
- php

- Java
- c/c++
  - 同步模块（类）加载机制

c/c++

- A.h 申明模块类，结构，函数
- A.c/A.cpp/A.lib/A.so/A.dll 实现模块

- ----------Main.c/Main.cpp

```
#include "A.h"
int main(){
   //use A
   return 0;
 }
```

- 通过条件指令 #ifdef 防止重复载入同一模块申明
- 编译时指定搜索目录：头文件，库文件（链接，运行）
- 离线编译器进行压缩，合并.

php

- A.php 模块定义以及实现
- ----------main.php

```
require_once("A.php");
//use A
```

- 使用 require_once 载入，防止重复
- 配置 include_path 定义查找路径
- 没有离线编译过程，引擎动态优化

java/python

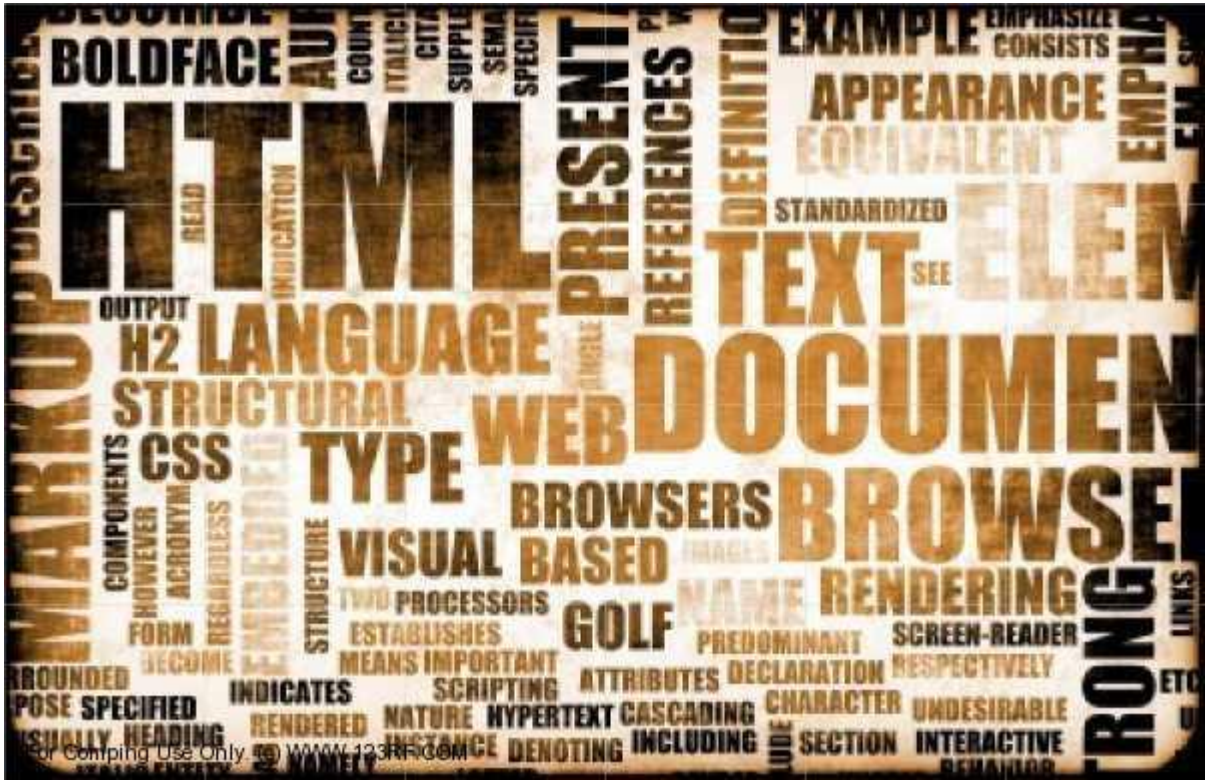
- A.java/A.py 模块定义实现
- ----------main.java/main.py

```
import A;

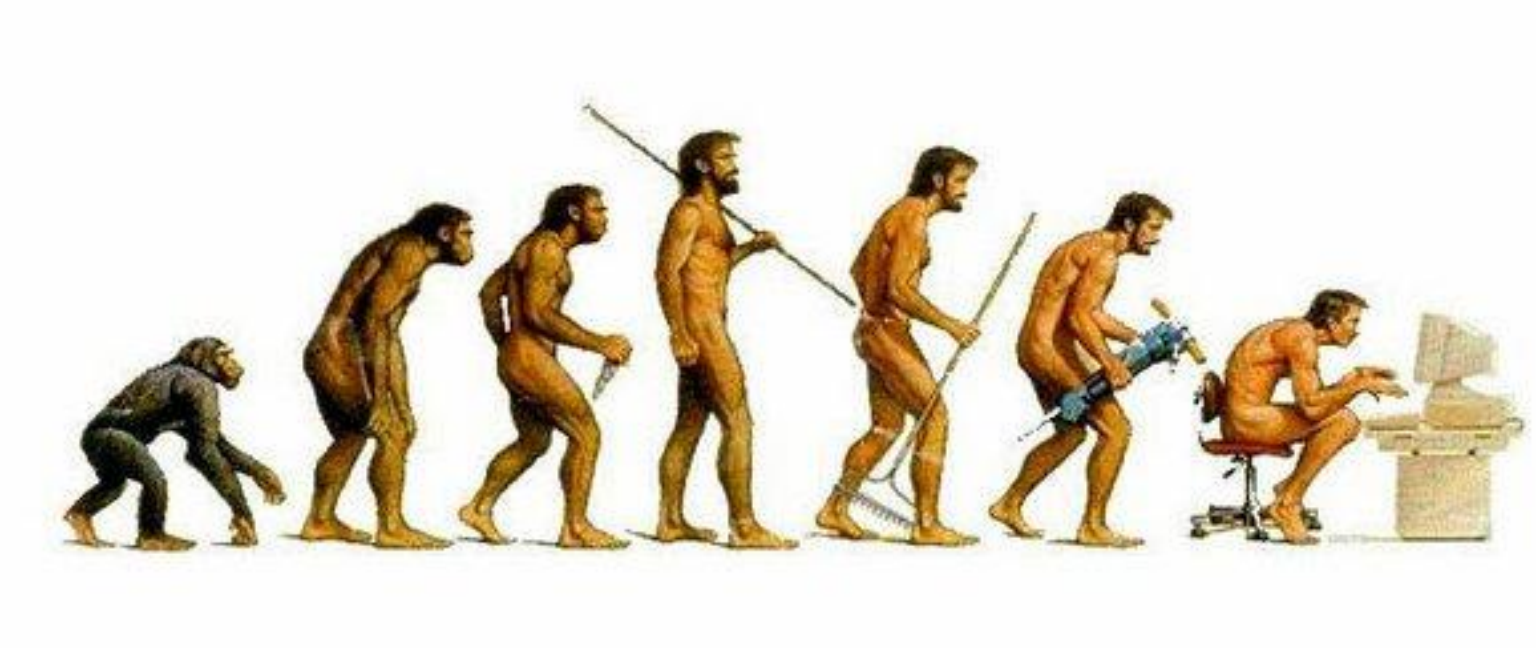
//use A
```

- import 引用其他模块
- classpath/sys.path 定义模块路径，自动补全后缀名查找对应模块文件
- Jar 来表现合并，class/pyc 来表现压缩.

前端代码



进化



混沌

```
<body onload="init()">
 <a href="#"
 onclick=""
 color="red">
 Click
 </a>
</body>
```

洪荒

- 脚本，标签，样式分离

```
<body>
  <a class='go'id='act'>
  click me
  </a>
  <script>
  window.onload=function(){};
  var a=document.getElementById('go');
  a.onclick=function(){};
  </script>
</body>
```

远古

- 外部脚本，全局勿扰，闭包

- ----------y.html

```
<body>
  <a class='go'>click<a>
  <script src='xx.js'></script>
</body>
```

- -------xx.js

```
(function(){})();
```

近代

- 代码逻辑分离

```
<script src='code1.js'>
</script>
<script src='code2.js'>
</script>
<script src='main.js'>
</script>
```

现代

- 动态并行加载
- 模块依赖
- 自动压缩打包
  - Labjs : script loader
  - Yui3/kissy : module loader
  - Closure library : module program

labjs

- 不需要静态引入外部脚本

```
<script>
  $LABjs
  .script("a.js")
  .script("b.js")
  .wait()
  .script("c.js");
</script>
```

YUI3/kissy

- 利用 function wrapper 分离 load (下载), execute (执行) 阶段。

- 引入模块概念
  - 模块定义
  - 模块注册
  - 模块使用

Yui3/kissy

- 模块定义

```
S.add("module1",function(S){
    S.Module1=function(){};
});
```

Yui3/kissy

- 模块注册

```
S.add({
    module1:{
        fullpath:"xxx.js"
    }
});
```

Yui3/kissy

- 模块使用

```
S.use("module1",function(S){
    //use S.Module1
});
```

Closure library

- 仅用于开发阶段的代码组织
- 通过 document.write script 模拟同步引入

```
goog.provide('goog.dom');
goog.require('goog.object');
goog.dom.setProperties = function(element
, properties) {
  goog.object.forEach(properties,
    function(val, key) {
    }
  );
};
```

Closure library

- 紧密结合closure compiler，明确区分开发与部署(@define)阶段。
- 开发阶段格外载入deps.js，指明依赖。
- 部署阶段通过工具依赖合并以及去除 require 指令。

未来

- Commonjs/module 规范?
  - Requirejs ?
  - Seajs?

- kissy 1.2 ?

- ```
  <script
      src='module.js'
      data-main='init'>
  </script>
  ```
  ```

- KISSY 模块化实践

KISSY 模块化演变

- < 1.2

```
// core.js
KISSY.app("Biz");

Biz.add({xx:{
  fullpath:"http://yy/xx.js",
  requires:[“zz"],
}});
```

定义模块

```
// xx.js
Biz.add("xx",function(B){
  B.XX = function(){}
});
```

缺点

- 中央配置无法模块解耦
  - 模块不具备内聚性
- 开发部署过程不容易分离
  - 文件打包与独立动态加载不容易实时切换

首页

- 'fp-mods': {

  fullpath:
  'http://a.tbcdn.cn/??s/kissy/1.1.6/switchable/switchable-pkg-min.js,s/kissy/1.1.6/suggest/suggest-pkg-min.js,s/kissy/1.1.6/datalazyload/datalazyload-pkg-min.js,s/kissy/1.1.3/flash/flash-pkg-min.js,p/fp/2011a/expressway/profile-min.js,p/fp/2011a/header/header-min.js,p/fp/2011a/attraction/attraction-min.js,p/fp/2011a/expressway/expressway-min.js,p/fp/2011a/category/category-min.js,p/fp/2011a/category/sub-promotion-min.js,p/fp/2011a/guide/alimama-ecpm-min.js,p/fp/2010c/js/fp-hubble-monitor-min.js,p/fp/2011a/hotsale/p4p-min.js,p/fp/2011a/footer/footer-min.js?t=20110513.js'
  }
- 不容易拆开 fiddler 调试

- KISSY 1.2 核心模块组织

避免污染全局模块

- 之前：

```
S.add("dom",function(S){
    S.DOM=function(){};
});
```

模块也有值

- 1.2 :

```
S.add("dom",function(S){
  var DOM=function(){}
  return DOM;
});
```

精简命名空间

- 之前：
- 两处命名空间

```
S.add("event-target",
function(){
  S.namespace("Event.Target");
  S.Event.Target=function(){};
});
```

精简命名空间

- 1.2 ：
- 依赖文件路径 event/target.js

```
S.add(function(){
  var EventTarget=function(){}
  return EventTarget;
});
```

依赖注入

- 之前：
- 定义模块

```
KISSY.add("switchable",
        function(S){
    var DOM = S.DOM;
});
```

依赖注入

- 1.2：
- 定义模块

```
KISSY.add("switchable",
        function(S,DOM){},{
        requires:["dom"]
});
```

依赖注入

- 之前：
- 使用模块

```
S.use("dom",function (S){
    var DOM=S.DOM;
    //use DOM
});
```

依赖注入

- 1.2：
- 使用模块

```
S.use("dom",
  function (S,DOM){
    //use DOM
});
```

子模块组织

- 之前:
- 同一模块使用 host 分文件:

```
S.add("dom-attr",
    function(){},
    {host:["dom"]}
);

S.add("dom-base",
    function(){},
    {host:["dom"]}
);
```

子模块组织

- 1.2:
- 模块间依赖

```
S.add("dom",function(){},{
   requires:["dom/base","dom/attr"]
});
S.add("dom/attr",function(){},{
   requires:["dom/base"]
});
```

摆脱模块注册

- 之前：

```
S.add({
    'switchable':{
        fullpath:"switchable/switchable.js"
    },
    'overlay':{
        fullpath:"overlay/overlay.js"
    }
});
```

摆脱模块注册

- 1.2 ：
- 包与路径约定

```
S.config({
    packages:[{
        name:'',
        path:"."
    }]
});
 S.use("overlay")->./overlay.js
 S.use("switchable")->./switchable.js
```

代码版本更新

- 之前：

```
S.add({
     'switchable':{
         fullpath:"switchable.js?t=201105"
     },
     'overlay':{
         fullpath:"overlay.js?t=201105"
     }
});
```

代码版本更新

- 1.2
- 包级别更新机制：

```
S.config({
  packages:[{
      name:'',
      tag:'20110401',
      path:"./"
   }]
});
```

- 请求包内模块文件时加入后缀，强制更新

应用级的兼容性

- 兼容 1.1.7 以下应用代码
- ~~约定优先配置~~
- 配置优先约定

全局命名空间

- 重新污染全局 S

```
KISSY.add("core",function
    (S,UA,DOM,Event,Node,JSON,Ajax,Anim,Base,Cookie){
   var re={
     UA:UA, DOM:DOM, Event:Event, Node:Node, JSON:JSON,
       Ajax:Ajax, Anim:Anim, Base:Base, Cookie:Cookie,
       one:Node.one, all:Node.List.all, get:DOM.get,
       query:DOM.query
   };
   S.mix(S,re);
   return re;
},{ requires:[ "ua", "dom", "event", "node", "json",
    "ajax", "anim", "base", "cookie" ]});
```

KISSY.js 与 core

- KISSY.js 结尾初始化 core
  - KISSY.use("core");

核心模块 与 core

- 内置配置

```
KISSY.config({
  combine:{
    core:['dom','event','anim',…]
  }
});
```

- 遇到需要载入 dom,event 等核心模块时，从 core 所在的 javascript 文件载入。

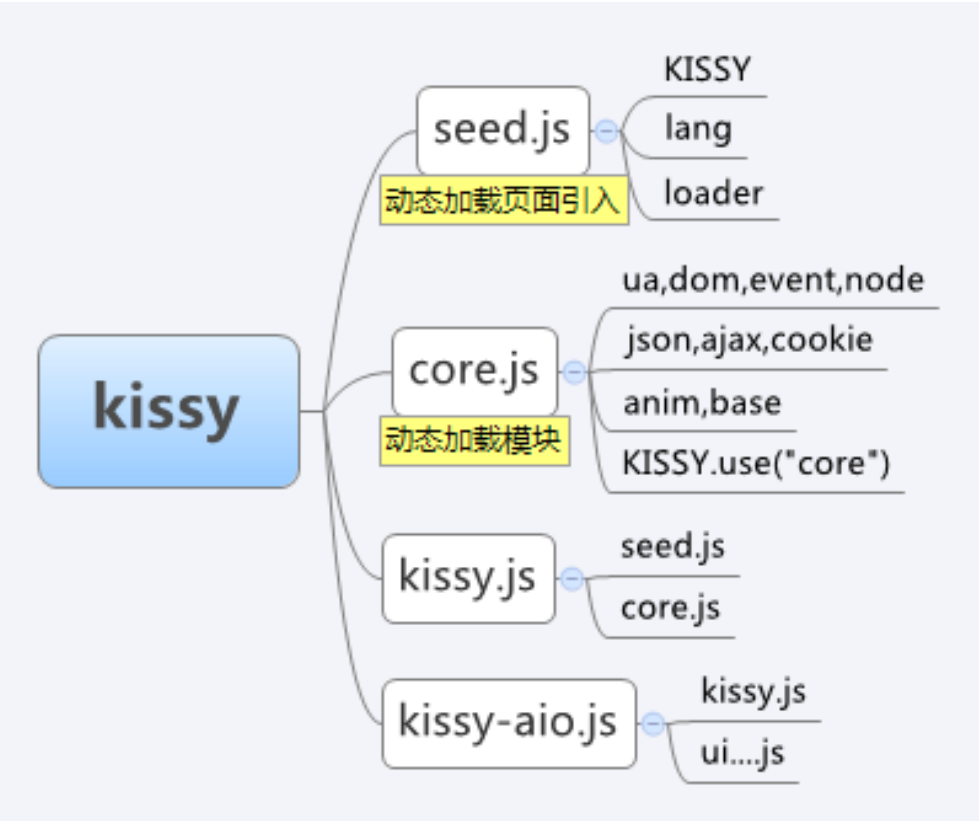
依赖注入

- 可以不使用，模块定义时依然污染全局空间。

```
S.add("biz",function(S){
    S.Biz=function(){};
});
```

代码结构

- seed/core/kissy/kissy-aio



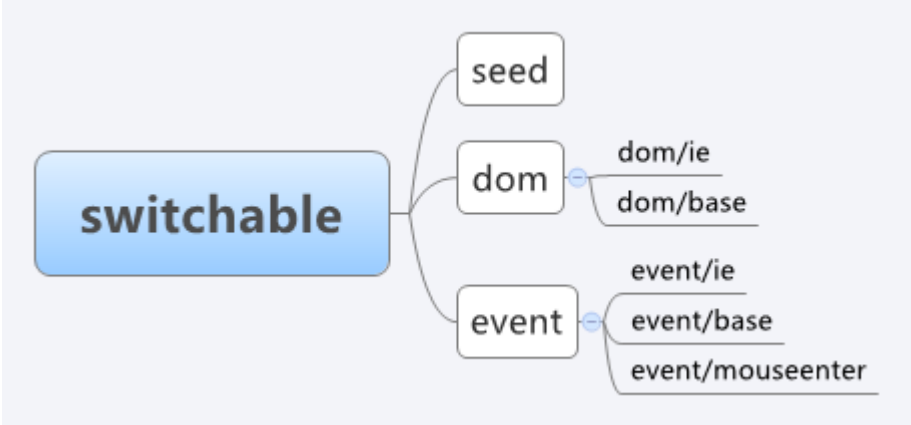
Kissy module compiler

- Closure compiler + kissy module combiner
- 只需指定最终使用的模块
- Java -jar module-compiler.jar

只打包用到的模块

- only use switchable，只打包用到的代码
- 依赖透明，no combo.

- requires=switchable
- output=switchable.js



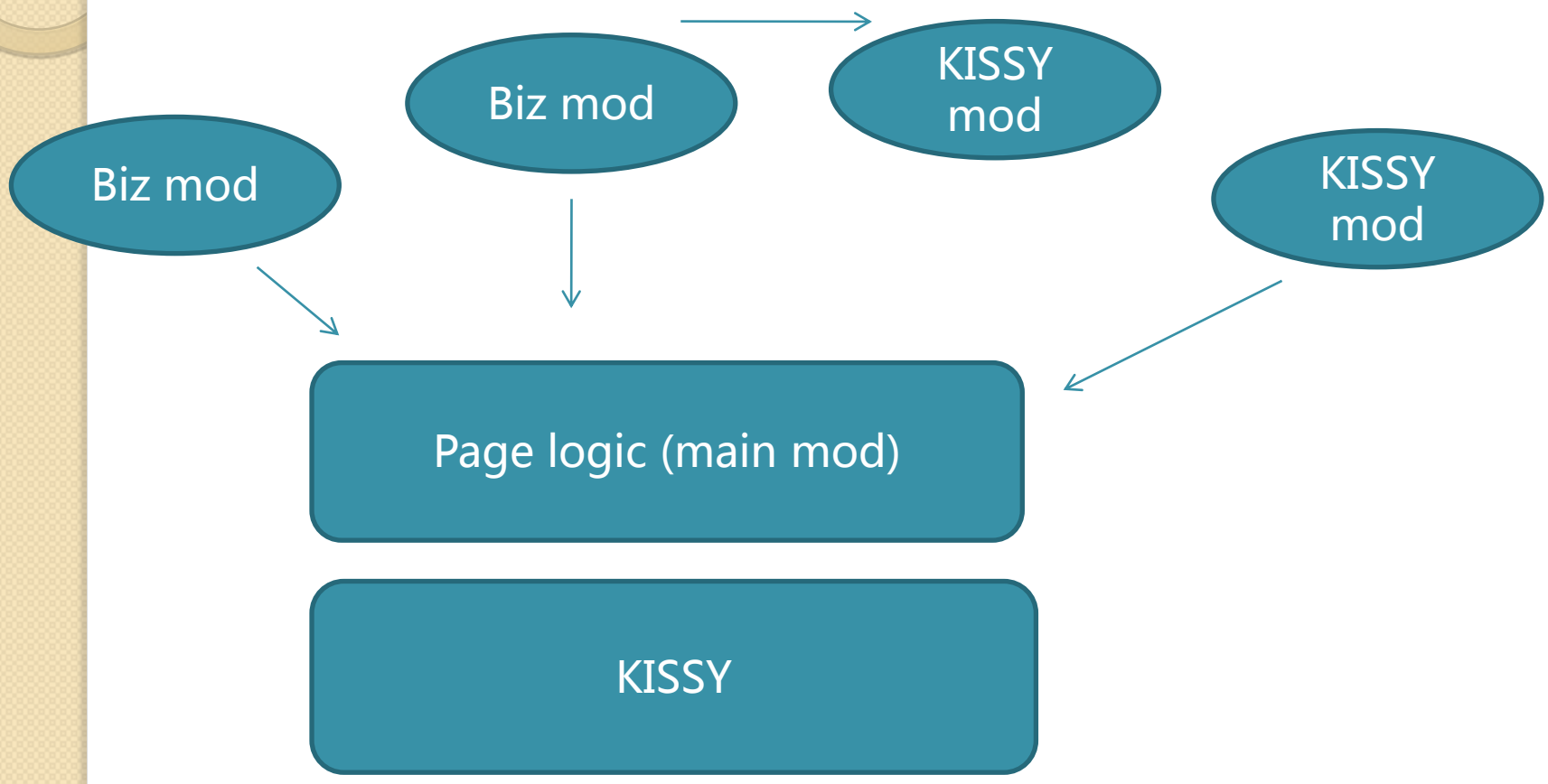
- KISSY 1.2 业务中的应用

app deprecated

- `KISSY.app("Biz");`
  - 需要静态引入文件 biz-core.js
    - 业务模块需要和 Biz 交互
    - 无谓得加了一层
  - 业务模块无法依赖 KISSY 内置模块

```
// custom 依赖 Biz 的 overlay ，无法引用 KISSY 内置模块
Biz.add("custom",function(B,overlay
){
},requires:["overlay"])
```

前端页面构建



- 交易平台化，模块划分与通信

页面组成模块编写

- 确认充值模块依赖表单验证模块

```
//-------------tc/mods/charge.js
KISSY.add(function(S,Form){
        //调用通用表单验证对自己数据进行验证
        //  验证通过后广播
        this.broadcast(“verify-ok”);
},{requires:"tc/mods/form"});
```

1. 模块命名空间
2. 模块依赖注入
3. 模块通过直接依赖或者全局广播使用其他模块的服务
4. 单个模块具备完整的自描述

启动模块编写

- 页面有且只有一个启动入口模块

```
//--------------tc/page/main.js
KISSY.add(function(S,Charge){
//启动页面各个组成模块
//页面级逻辑
},{requires:"tc/mods/charge"});
```

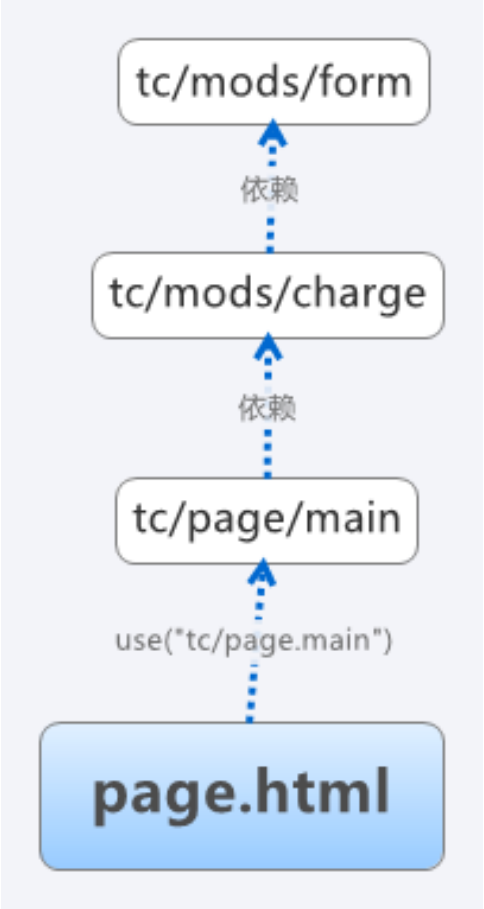
- 特征：
  1. 一个页面只有一个启动模块
  2. 启动模块负责初始化组成模块

页面代码

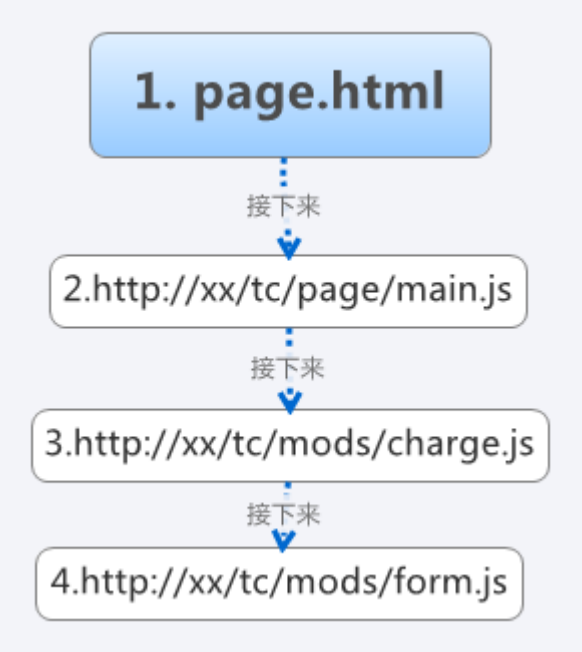
- 页面负责启动模块初始化

```
//----------------- page.html
<script>
KISSY.use("tc/page/main",function(S
  ,Main){
 //初始化启动模块
});
</script>
```

模块与代码下载



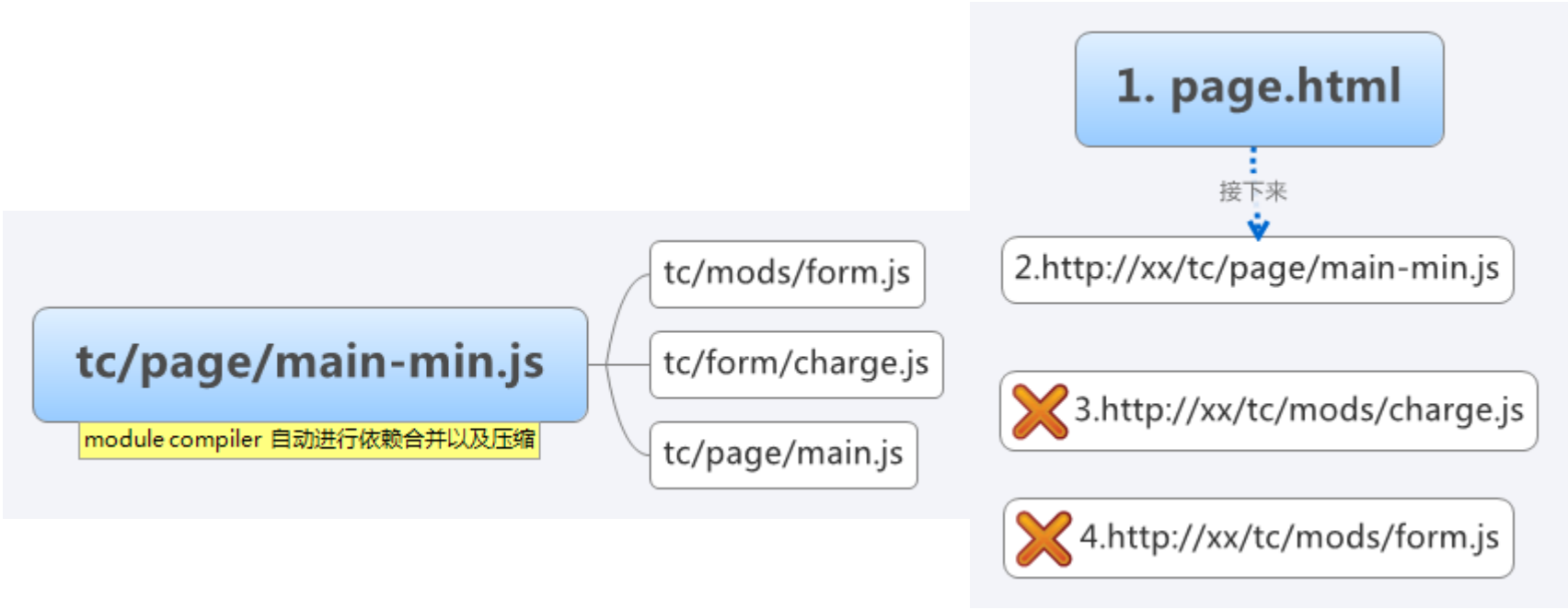
开发阶段载入 kissy.js



模块与代码下载

线上载入 kissy-min.js

- 线上环境



线上调试采用 ucool 或 url 加 ?ks-debug 即可返回开发阶段

文件更新机制（一）

- 基于path

```
~110526/
 `~tc/
   |~cart/
   | |-address-test-min.js
   | |-address-test.combine.js
   | |-autobuy-min.js
   | |-autobuy.combine.js
   | |-cart-min.js
   | |-cart.combine.js
   | |-fastBuyAccount-min.js
   | |-fastBuyAccount.combine.js
   | |-fastbuyphone-min.js
   | |-fastbuyphone.combine.js
   | |-fastBuyWow-min.js
   | |-fastBuyWow.combine.js
```

```
|+110526/
|+2.0/
|~4.0/
| |~tc/
| | |+cart/
| | |+core/
| | |+form/
| | |~mods/
| | | |+address/
| | | |+anony/
| | | |+authentication/
| | | |+checkCode/
| | | |+fare/
| | | |+fareInsure/
| | | |+fastBuyAgreed/
```

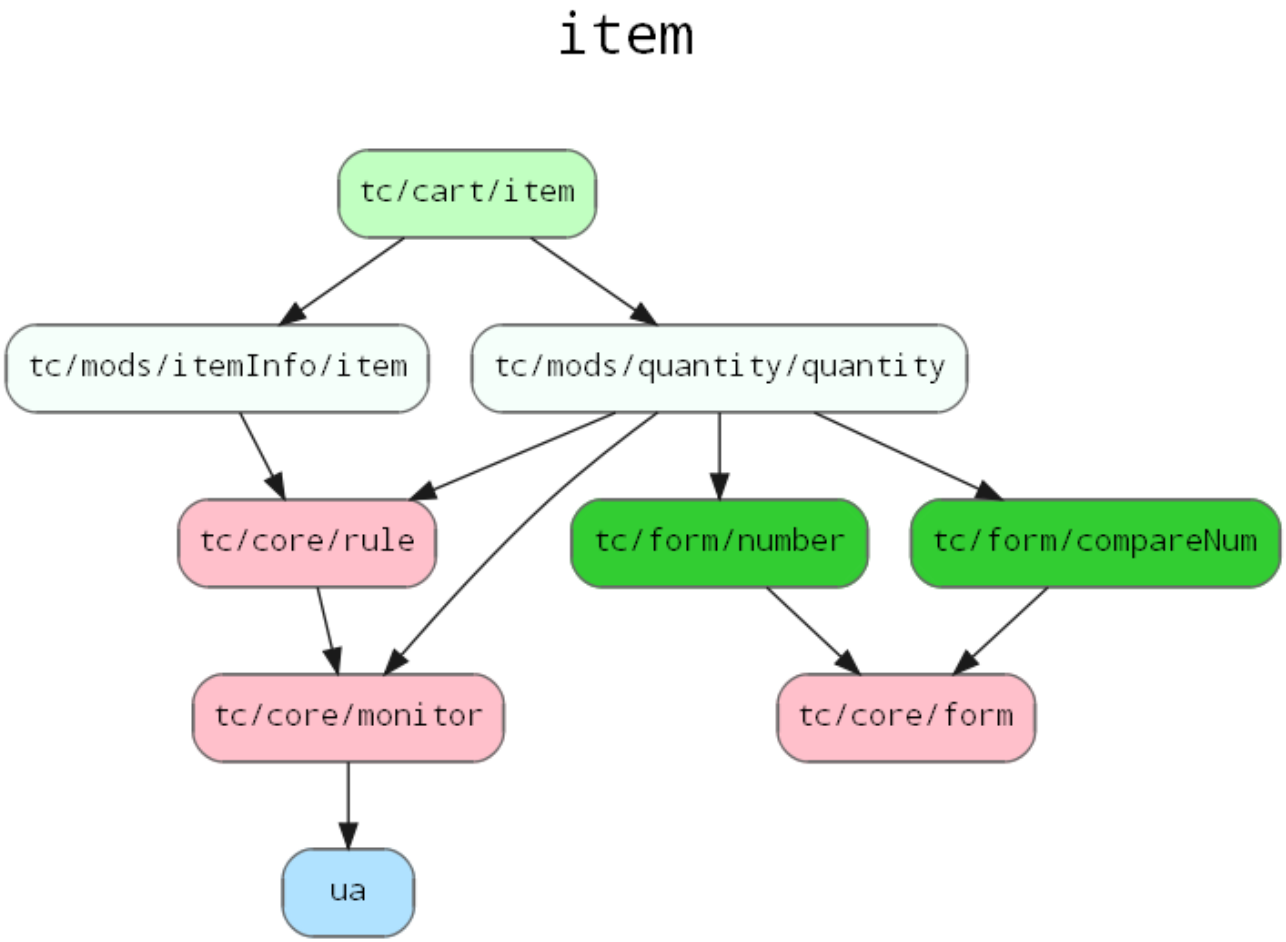
文件更新机制（二）

- 基于path/tag，二次封装

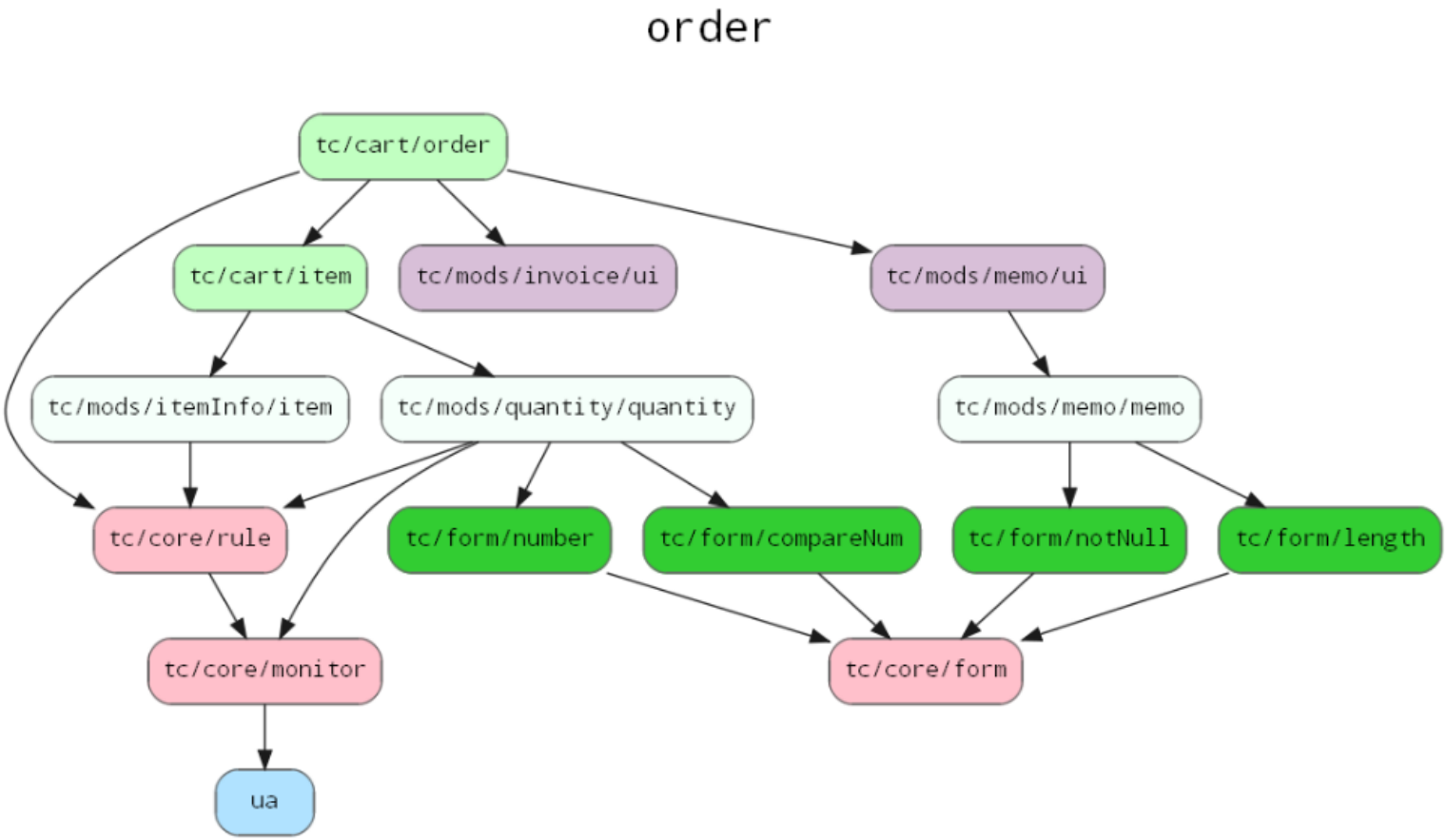
```
var isDaily = document.domain.indexOf('daily.') > -1;
cdn = cdn || (isDaily ? 'http://assets.daily.taobao.net' : 'http://a.tbcdn.cn');
S.config({packages: [{
    name: 'tc',
    path: cdn + '/apps/tradeface/' + timestamp + '/',
    tag: tag || timestamp,
    //path: '../',
    charset: 'gbk'
}]});
return S;
```

```
KISSY.TC.setTimeStamp('110526', 'http://assets.daily.taobao.net', '1234567890');
```

依赖关系单一（正向）

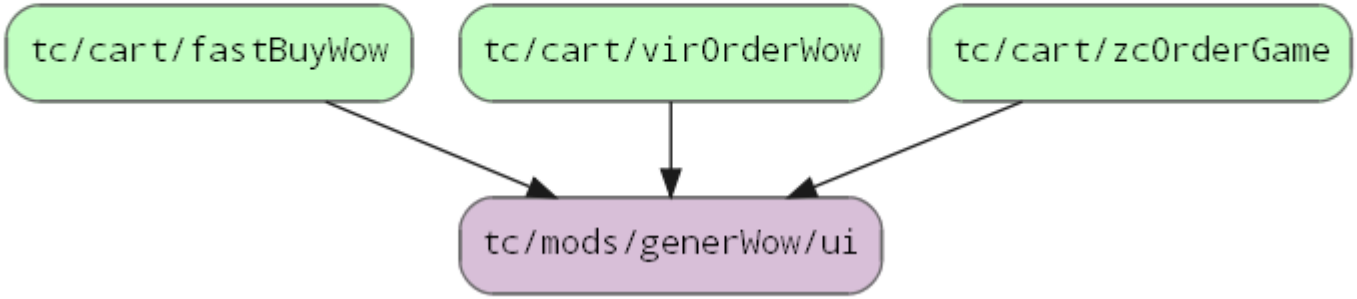

item
tc/cart/item
tc/mods/itemInfo/item
tc/mods/quantity/quantity
tc/core/rule
tc/form/number
tc/form/compareNum
tc/core/monitor
tc/core/form
ua

依赖关系单一（正向）

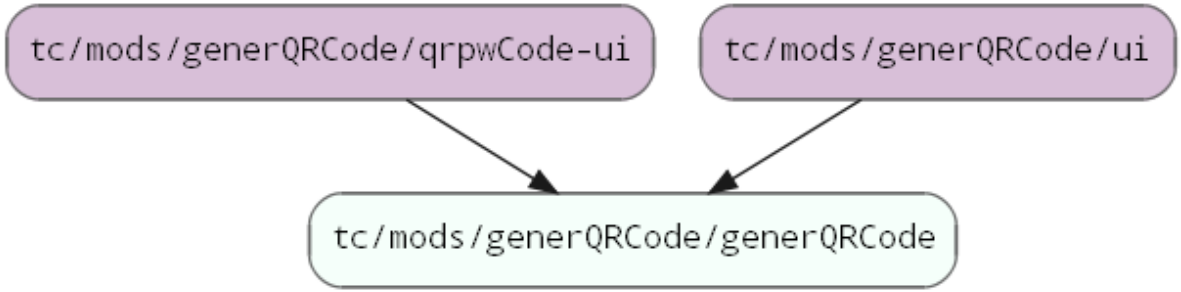

order
tc/cart/order
tc/cart/item
tc/mods/invoice/ui
tc/mods/memo/ui
tc/mods/itemInfo/item
tc/mods/quantity/quantity
tc/mods/memo/memo
tc/core/rule
tc/form/number
tc/form/compareNum
tc/form/notNull
tc/form/length
tc/core/monitor
tc/core/form
ua

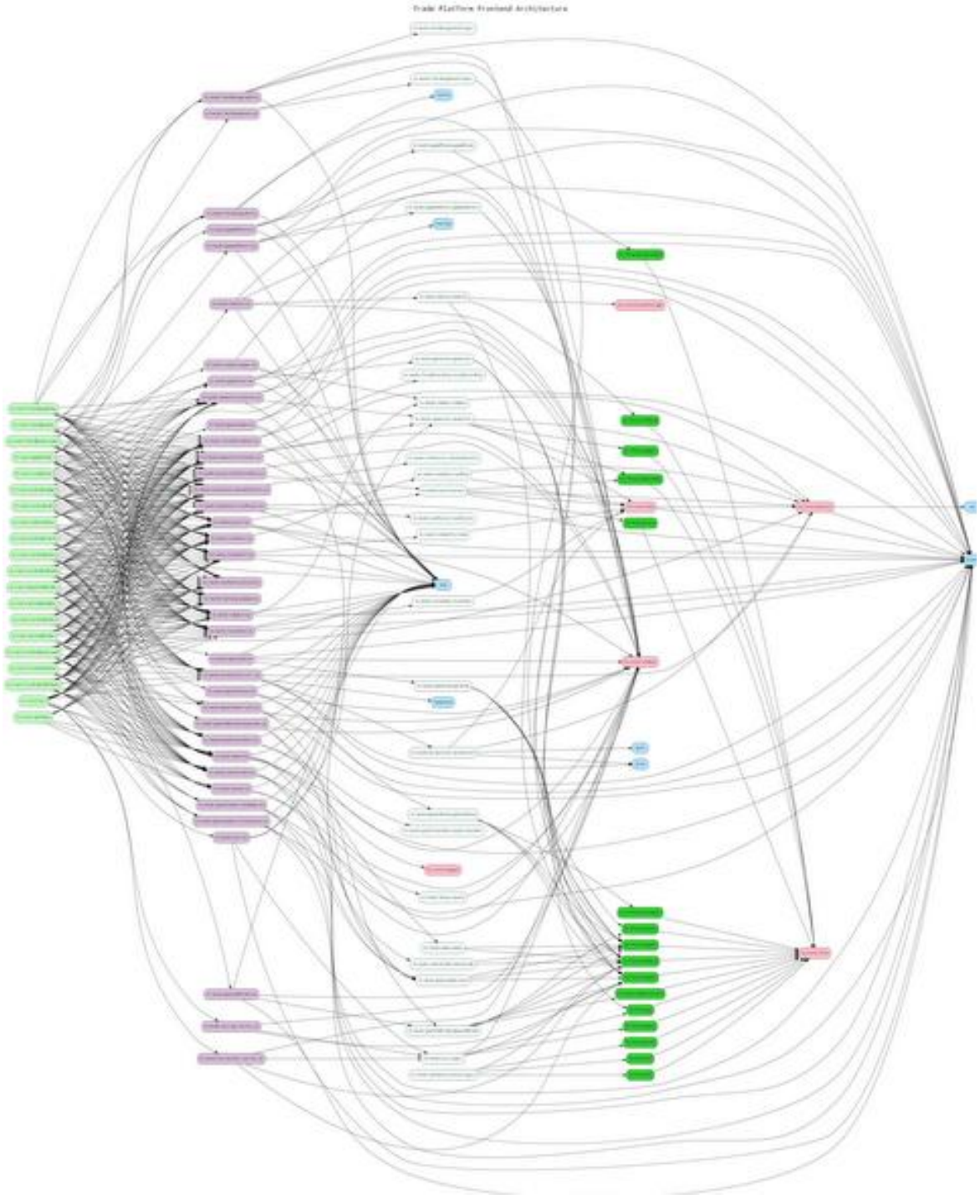
依赖关系单一（反向）

tc/mods/generWow/ui

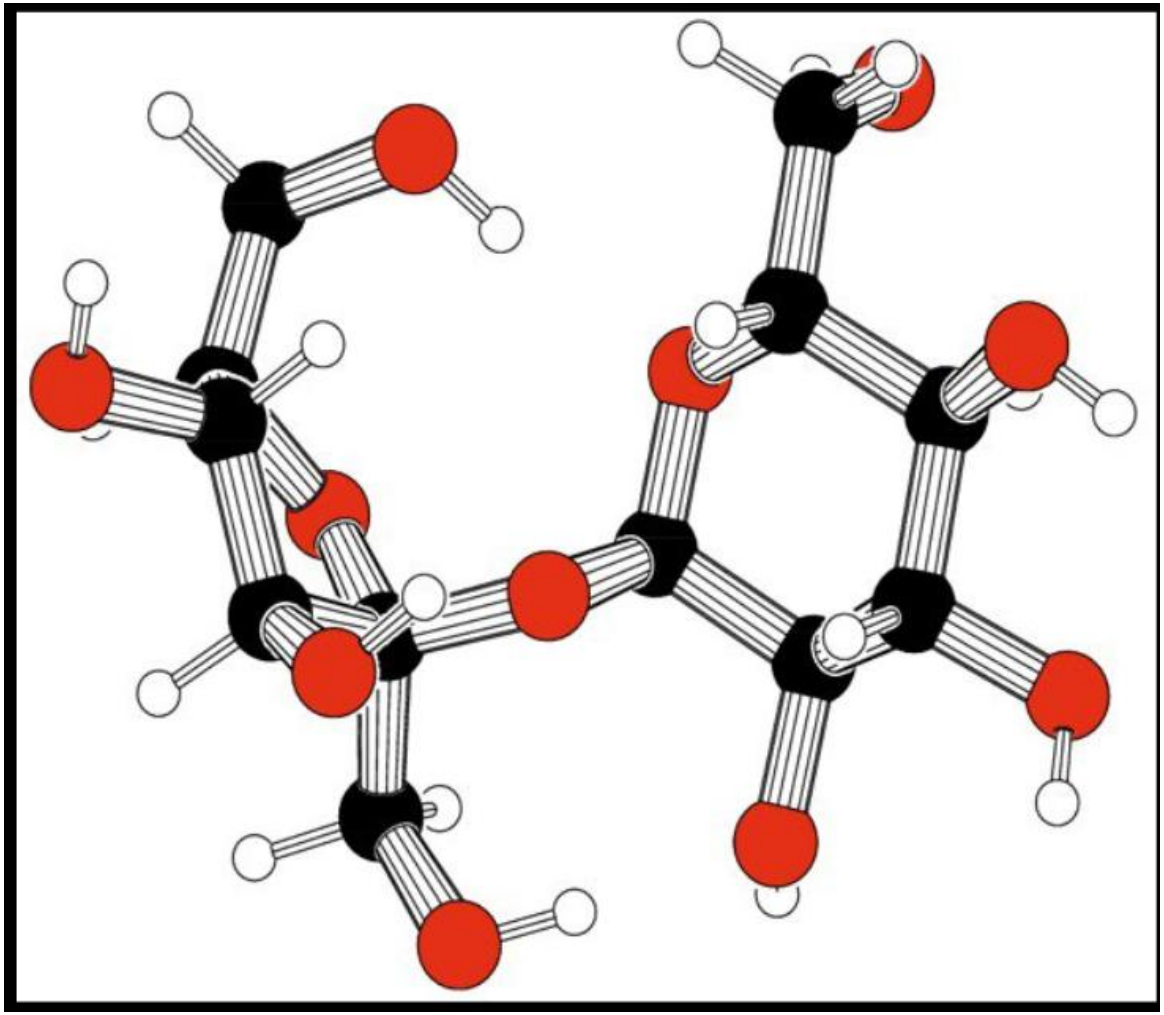


tc/mods/generQRCode/generQRCode





Module is future



感谢

- 李牧 kissylite
- 玉伯 seajs
- 拔赤 kissy loader
- Andy Chung requirejs