SFEI

The North Star Documentation

说明文档，项目代号“深岩大地”

Sierra Six
2012-12-06

Table of Contents

Section 0 - Preamble / 前言

0.1 本文档的写作目的是从概念和参考两个角度详尽地阐述本项目的各项特性、代码用途、扩展用法等。文档主体分为设计文档和技术文档两个部分，分别从两个视角来对整个项目进行近可能详细的描述。前者主要包括本项目的设计理念等，后者主要包括一些技术角度的参考资料。

0.2 稳定性索引

在本文档之中，你会看到一些指示该部分稳定性的标志。本系统仍处在不断的开发过程中，因此在它逐渐变得成熟的过程中，一些部分可能比其他部分更加可靠和稳定。有的部分经过了大量的测试，抑或是其它部分已经非常依赖于该部分，以至于该部分不太可能再作变动。有的部分较新，还没有很好地接受实践的检验，也有的部分因为一些原因已经被弃用，仅因为向下兼容而保留。稳定性索引的标志为：[SI0]

0.2.1 稳定性等级 0 - 已弃用：这部分功能存在一些问题，已经被弃用，一般情况下不要依靠该功能。该部分的文档会指出应当使用哪些功能来代替该被弃用的功能。

0.2.2 稳定性等级 1 - 实验性：这个特性是最近引进的，随着时间的推移可能改变或者被移除。请对该部分进行测试并提供反馈。

0.2.3 稳定性等级 2 - 不稳定：这个特性正在成型的过程中，需要更多的测试才能认为它是稳定的。如果没有大的意外，这个功能将会被保留。

0.2.4 稳定性等级 3 - 稳定：这个特性在测试中表现良好，但是可能会有小的变动。这个功能一般会被保留。

0.2.5 稳定性等级 4 - 已冻结：这个功能已经经过足够的测试，不太可能再发生变化。

0.2.6 稳定性等级 5 - 已锁定：除非发现非常严重的 bug，本功能不会发生改变。请勿提出改变该部分的功能的请求。

0.3 依赖组件

本系统需要以下部件已保证正常运行：[SI4]

0.3.1 Node.js, 版本 0.8.x

0.3.2 Mongodb, 版本 2.0+

0.3.3 依赖的 NPM 组件已经在 package.json 中说明。

Section 1 - Design Documentation / 设计文档

1.1 设计说明

1.1.1 创意来源

本网站构思来源于科学松鼠会、果壳网等大众科普网站以及 Twitter，微博等社交网站。网站的名字与它的项目代号相同，为“北极星”，含义是希望本网站的用户能像看到北极星一样在求知与普及科学的道路上前进。

在信息大爆炸的当代社会中，每个人每天都要经手大量的信息，无论是从身边的人，还是从人人、微博等社交网站。这些信息之中带有大量的不科学的、误导性的内容，因此我希望借这个网站向普通大众普及科学知识与批判性的思维方式，从而减少有害信息对社会的危害。

从功能的角度来看，本网站的创意是一个带有基础社交功能的科普类博客网站。网站的用户可以关注一些著名博主以便及时获取他所撰写的博客，同时也可以在某一博客下与博主或其他关注者进行讨论。

1.1.2 初期目标

1.1.2.1 完成网站前端、后台部分的开发，撰写详尽的说明文档。

1.1.2.2 将本网站部署到互联网上，租用服务器或者使用平台即服务云计算平台。

1.1.3 中期目标

1.1.3.1 开启公测，吸引小规模的用户长期使用本站的部分或全部功能。

1.1.3.2 排查并修正可能出现的 bug。

1.1.4 最终目标

1.1.4.1 组成一支专业的团队负责本网站的运营事务。

1.1.4.2 吸引大规模的用户访问本站，并通过多种渠道进行宣传。

1.1.4.3 选择可承受访问量大的服务器部署网站，并通过广告实现收支平衡。

1.1.4.4 如果有必要，对于出现的性能瓶颈进行优化。

1.1.5 亮点

1.1.5.1 使用 Node.js 与轻量级 NoSQL 数据库 Mongodb 开发：Node.js 是一项近几年逐渐变得热门的技术。该技术允许开发人员使用 JavaScript 进行服务器端编程，从而削减了前端与后台之间的门槛。同时由于使用 NPM 包进行模块管理，开发人员可以轻易分享自己编写的模块或是使用别人的开源模块。Mongodb 是一种开源的 NoSQL 数据库，相比 MySQL 那样的关系型数据库，它更为轻巧、灵活，非常适合在数据规模很大、事务性不强的场合下使用。

1.1.5.2 使用了 HTML5 与 CSS3。HTML5 与 CSS3 分别代表了下一代的 HTML5 与 CSS3 技术。HTML5 相比 XHTML1.0 加入了许多新的特性，虽然该标准尚不是 W3C 的正式标准，但是主流浏览器已经对它提供了支持。

1.1.5.3 实现了博客与简单社交功能的结合：很多情况下，博客功能和社交功能都是分开的，用户需要查看某位博主的博客就必须时常查看他的博客网站或是使用 RSS 订阅他的博客（如果他的博客网站提供了该功能的话）。在这个网站上，用户可以关注博主，从而每次访问是都能获得关于该博主的最新消息。

1.1.5.4 完善的权限控制：对于一些有敏感或机密信息的博客，可以设置它的安全等级，只有访问权限超出这一安全等级的用户才能查看该文章。同时也可以为文章设置例外，允许某个用户访问。

1.1.5.5 美观简洁，未来感十足的界面：整个界面非常淡雅、简洁，没有多余的装饰，各项样式使用地恰到好处，将整体页面的设计理念隐藏与无形之中，不引人注意，信息却能得到传达。

1.1.5.6 提供两种语言供用户选择，可以轻易扩展其他语言。

1.2 网站功能分析

1.2.1 **网站特色**：提供了一个博客与社交为一体的平台，用户可以在本网站阅读有关科普主题文章，阅读某特定博主所撰写的博客，在文章或博客下发表评论及与博主

或其他用户开展讨论，以及自己发表博客。用户可以通过文章分类来查找自己需要的文章，也可以通过设置访问权限与例外来防止对于自己博客的未授权访问。[SI4]

1.2.2 **用户群体**：本网站的设计用户群为对各类科学学科感兴趣的社会各界人士，期望文化程度为高中及以上，不过任何对于科学知识有了解的人士皆为本网站用户群。[SI3]

1.3 内容和功能

1.3.1 **内容**：网站内容主要包括管理员发布的各类科普文章，题材包括但不限于数学、物理、化学、生物、地理、医学以及生活常识。网站的内容还包括用户自行发布的文章。[SI3]

1.3.2 **功能**：网站的功能主要分两类，博客与社交。网站的用户可以发布博客文章，在别人或自己的文章下发表评论，以及与其他用户展开对文章内容的讨论。用户还可以通过关注其他用户以及时获取对应用户的动态。[SI3]

1.4 可视化目录清单

*本部分提供了网站文件相关功能的基本介绍，如果需要更详细的介绍，请参阅 2.2

/models - 网站后台功能模块，包括 [SI4]

access.js - 用户认证对象的构造函数

db.js - 数据库对象的构造函数及其方法

init.js - 包含初次部署本网站时及根管理员用户出错时需要执行的代码

localization.js - 包含 Renderer 需要使用的语言信息

log.js - 日志对象的构造函数及其方法

renderer.js - 渲染器对象的构造函数及其方法

root.js - 根管理员用户信息

security.js - 过滤 POST 操作中的安全性问题

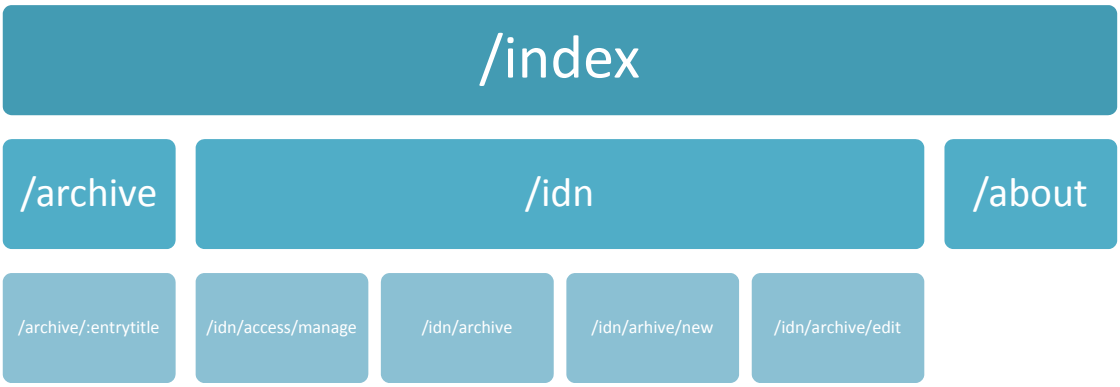
/node_modules - Node.js 功能模块 [SI5]

/public - 网站每个页面的共用数据，包含客户端需要的图片、css、js 文件 [SI5]

/routes - 管理网站的路由功能 [SI5]

/views - MVC 视图模板，目前包含两种语言：简体中文与英语 [SI3]

1.5 网站层次结构



层次结构如图。 [SI4] 相关解释如下，需要更详细的信息请访问 2.2.2

/index - 主页面，包含热点文章、热点用户等

/about - 关于页面，本网站的基本说明

/archive - 档案库，可查看文章列表

/archive/:entrytitle - 查看 / 评论题为<entrytitle>的文章

/idn - 用户登录 / 用户基本信息管理

/idn/access/manage - 用户管理（需要管理员权限）

/idn/archive - 选择文章编辑 / 删除

/idn/archive/new - 发表新文章

/idn/archive/edit - 编辑文章，需要 GET 参数 t

Section 2 - Technical Document / 技术文档

2.1 总览

本网站基于 Node.js 与 Mongodb 开发。Node.js 是一个基于 Chrome JavaScript 运行时的平台，可以用于轻易地创建高性能、扩展性好的网络应用程序。Node.js 使用了一种事件驱动的，非阻塞式的输入/输出模型，这使它成为了一种轻量级、高效率的工具，非常适用于构建跨平台的实时大规模数据应用。

MongoDB 是一个可扩展的高性能开源 NoSQL 数据库。MongoDB 是由 C++编写的，数据以类似 JSON 的格式存储，与 Node.js 有良好的兼容性。

本网站基于 MVC 架构设计，视图模型使用了 ejs，这是一种与 html 语法很相似的视图。Ejs 视图决定整体排版，同时通过 Renderer 对象实现多个同种 html 代码的生成。

2.2 网站后台

2.2.1 主程序 app.js

主程序功能主要包括：

2.2.1.1 获取需要的组件； [SI4]

2.2.1.2 对服务器进行配置，包括视图引擎，session 等； [SI4]

2.2.1.3 通过 models/init.js 进行初始化，检查根管理员用户是否存在； [SI4]

2.2.1.4 如果一切正常，启用 80 端口开始监听请求。 [SI5]

2.2.2 路由

路由相关代码存储于 routes 文件夹。为清晰起见，不同大类的路由控制分别存储于不同的 js 文件中。路由的模块导出一个函数，该函数有且仅有一个参数 app，即 app.js 中创建的服务器对象。根据需要，可以改变函数中的获取组件数目。

```
module.exports = function(app){
  var systemVersion = require('../package').version;
  var Renderer = require('../models/render');
  var renderer = new Renderer('');
```

```
  var setLang = require('../models/localization').setLang;
  var Db = require('../models/db');
  var db = new Db();
  var Log = require('../models/log');

  app.get(<path>, function(req, res){ ... });

  app.post(<path>, function(req, res) { ... });

  ...
  More of router functions...
  ...
}
```

/routes/index.js - 主页相关。可用的方法与路径：[SI5]

GET / - 访问主页。View: index + indexlayout

GET /index - 同上

GET /about - 访问关于页面。View: about + layout

/routes/archive.js - 档案库页面相关。可用的方法与路径：[SI4]

GET /archive - 访问档案库，列出可访问的文章列表。View: archive + layout

GET /archive/:entrytitle - 访问题为:entrytitle 的文章。View: arhive_view + layout

POST /archive/get - 使用 AJAX 获取特定分类标签的文章。**仅用于 AJAX。**

POST /archive/comment - 发表评论。发送的参数应包括：accessid - 评论用户名, body - 评论正文, entryid - 该文章的 objectID。**仅用于 AJAX。**

POST /archive/comment/remove - 删除某一条评论。**仅用于 AJAX。**

POST /archive/comment/removeall - 删除所有评论。需要管理员权限。**仅用于 AJAX。**

/routes/idn.js - 图形数据网主页相关。可用的方法与路径：[SI3]

GET /idn - 如果 session.access 为空，则返回登录页面 idn_authenticate + idnlayout，否则返回用户页面 idn + idnlayout

GET /idn/logout - 登出，重定向到/idn

GET /idn/log - 查看日志。View: archive + layout

POST /idn/getMyArticles - 获取用户的文章。**仅用于 AJAX。**

POST /idn/getFeeds - 获取用户关注的信息。**仅用于 AJAX。**

POST /idn/getContacts - 获取用户关注的其他用户。**仅用于 AJAX。**

/routes/idn_access.js - 用户登录相关。可用的方法与路径：[SI3]

GET /idn/access - 编辑用户个人信息。access + idnlayout

GET /idn/access/register - 用户注册。idn_register + idnlayout

GET /idn/access/request/success - 用户注册成功。idn_register_success + idnlayout

GET /idn/access/manage - 管理用户。idn_access_manage + idnlayout

GET /idn/access/manage/request - 获取已经发送的请求。返回一个 request 对象 **仅用于 AJAX。已弃用。**

POST /idn/access/register - 发送注册请求，成功则返回{'ok': true}，客户端 js 控制跳转到/idn/access/request/success。**仅用于 AJAX。**

POST /idn/access/manage/request - 批准或拒绝一个请求，并将之从数据库中删去。成功则返回{'ok': true}。**仅用于 AJAX。已弃用。**

POST /idn/access/manage/getexisting -返回包括所有用户名的 html 字符串。用于编辑已经注册的用户的权限。**仅用于 AJAX。**

POST /idn/access/manage/new - 直接创建一个新的用户。**仅用于 AJAX。**

POST /idn/access/getlist - 返回包括所有用户名的 html 字符串。用于创建/编辑文章时添加例外。**仅用于 AJAX。**

POST /idn/access/manage - 修改相应的用户信息，请求应包括{_id, level, mlevel, admin}，将会更新数据库中_id 相符的用户。成功则返回{'ok': true}**仅用于 AJAX。**

POST /idn/access/manage/remove - 删除数据库中_id 相符的用户。成功则返回{'ok': true}。**仅用于 AJAX。**

POST /idn/access - 用户登录。如果登录成功则实例化一个 Access 对象并储存与 req.session 中。成功时客户端重载当前页面。**仅用于 AJAX。**

/routes/idn_archive.js - 文章查看/新建/编辑相关。可用的方法与路径：[SI3]

GET /idn/archive - 查看所有权限允许的文章。archive + idnlayout

GET /idn/archive/new - 新建文章。archive_new + idnlayout

GET /idn/archive/edit - 编辑文章，要求参数 t 中带有文章标题。archive_edit + idnlayout

POST /idn/archive/new - 将新建的文章存储于数据库。成功则返回{'ok': true, 'title': entry.entrytitle}，客户端重定向到/archive/文章名。**仅用于 AJAX。**

POST /idn/archive/edit - 更新数据库中的文章。成功则返回{'ok': true, 'title': entry.entrytitle}，客户端重定向到/archive/文章名。**仅用于 AJAX。**

POST /idn/archive/gettags - 获取已有的文章分类标签数组。**仅用于 AJAX。**

POST /idn/archive/remove - 删除文章。**仅用于 AJAX。**

/routes/idn.contact.js - 用户信息、关注、社交相关。可用的方法与路径：[SI2]

GET /idn/profile - 查看用户自己的信息，包括自己的文章和自己的个人信息。idn_profile + idnlayout

GET /idn/profile/edit - 编辑用户自己的信息。idn_profile_edit + idnlayout

GET /idn/profile/edit/imgselect - 用户头像选择。idn_profile_imgselect + idnlayout

GET /idn/profile/:profileid - 查看其他用户的信息。idn_profile + idnlayout

POST /idn/profile/edit - 编辑用户信息。**仅用于 AJAX。**

POST /idn/profile/edit/imgselect - 修改用户头像。**仅用于 AJAX。**

POST /idn/profile - 获取用户简介。用于用户信息的小弹出框体。**仅用于 AJAX。**

POST /idn/contact/follow - 关注某用户。**仅用于 AJAX。**

POST /idn/contact/unfollow - 取消对某用户的关注。**仅用于 AJAX。**

2.2.3 Db 对象

Db 对象用于从数据库中获取数据并执行相关的操作。通常涉及到数据库查询的操作需要一个回调函数作为参数来对获取的数据进行操作。

2.2.3.1 构造函数：function Mongo() - 返回一个新的 Db 对象。构造函数会首先检查 process.env.VCAP_SERVICES 是否存在，如存在则根据 Cloudfoundry 云计算平台的环境变量初始化数据库参数，否则按照本地模式初始化数据库。构造函数将设置新对象的属性：hostname, port, username, password, name, db. [SI3]

2.2.3.2 属性 [SI3]

hostname - 数据库服务器的名称

port - 数据库服务器的端口

username - 用户名

password - 密码

name - *

db - 数据库名称

2.2.3.3 方法 [SI5]

generate_mongo_url() - 根据对象参数，生成数据库 url 用于连接数据库

generate_ObjectId(ObjId) - 由对象 ID 字符串生成对象 ID

find_and_render(type, collection, query, sort, limit, callback, renderer, admin) - 查找符合 query 的条目，并通过相应的规则调用 renderer 相关方法生成 html 代码。

type - renderer 对象需要使用的模板：article_entry, article_entry_thumbnail, access_entry, access_request_entry, dropdown_li, log

collection - 搜索的目标集合

query - 查询参数

sort - 结果排序方式 {属性名: -1 (降序) / 1 (升序)}

limit - 显示结果数目，0 为不限

callback - 查询和渲染完成后执行的回调函数 function(contents)

renderer - 生成 html 代码所用的渲染器

admin - 是否渲染管理员内容

findOne(collection, query, callback) - 根据查询参数，返回第一个查询结果

update(collection, query, data, override, callback) - 根据查询参数，更新相应的条目。

data - 用于更新的数据

override - 是否用 data 完全覆盖之前的条目

insert(collection, data, callback) - 在数据库中插入新条目

remove(collection, query, callback) - 从数据库中删去相应的条目

2.3.4 Renderer 对象

Renderer 对象用于生成 html 代码。通过 localization.js，它能够支持生成多国语言的 html 代码。

2.3.4.1 构造函数：function (lang) - 根据语言初始化一个新的 Renderer 对象。[SI3]

2.3.4.2 属性 [SI3]

lang - 该对象生成 html 代码时使用的语言

accept_language - 对象支持的语言数组

2.3.4.3 方法 [SI3]

setLang(lang) - 设定该对象的语言

getView() - 获取该对象的语言对应视图目录的路径前缀

nav_dropdown_li(accessid, level) - 渲染右上角下拉菜单

nav_extend(params) - 渲染登录后的新导航选项

article_entry(entry, enable_edit) - 渲染一篇文章（不包括正文）

article_entry_thumbnail(entry, enable_edit) - 渲染一篇文章的预览

access_entry(entry) - 渲染一条用户编辑项目

access_request_entry(entry) - 渲染一条用户注册请求

buttons_admin() - 渲染管理员专用按钮

no_entry_found() - 渲染“未找到”

dropdown_li(text) - 渲染下拉菜单条目

label(text) - 渲染一个标签

article_log(log) - 渲染一条日志

checkbox_wlabel(text) - 渲染带标签的复选框

invalid_code() - 渲染无效信息

user_info(access) - 渲染用户信息

buttons_edit(title) - 渲染文章编辑功能按钮

comment(comment, access) - 渲染评论条目

comment_new(access) - 渲染新评论表单

profile_extend(myself, following) - 渲染关注、取消关注等按钮

profile_desc(access) - 渲染用户个人信息

profile_thumbnail(access) - 渲染用户个人信息（小窗口）

profile_not_found() - 渲染用户未找到信息

profile_article(access, req) - 渲染用户信息（大）

2.3.4.4 localization.js [SI3]

输出语言相关的对象。

```
exports.en = {
  'Terminate_Connection': 'Terminate Connection',
  'Not_Authenticated': 'Not Authenticated',
  ...
}
```

输出函数 setLang = function (req, renderer)

通过 req.query.lang，req.session.lang，及 req.headers['accept-language']设定 renderer 的语言。

2.3.5 Access 对象

2.3.5.1 构造函数：function Access(access) - 根据参数创建一个新的 Access 对象。[SI4]

2.3.5.2 属性 [SI4]

id - 用户名

codehash - 密码 SHA1 值

clearance - {level: x, mlevel: y, admin: true/false}

level - 访问权限等级

mlevel - 修改权限等级

admin - 是否为管理员

desc - 用户个人信息

follows[] - 用户的关注

2.3.5.3 该对象没有方法。[SI4]

2.3.6 Log 对象

2.3.6.1 构造函数 function Log({type, tag, user, desc[, time]}) - 根据参数创建一个新的 Log 对象。如果没有提供时间，将使用当前服务器时间。[SI3]

2.3.6.2 属性 [SI3]

type - 日志的类型，如 Access, Archive 等

tag - 日志的子类型，如 New Entry 等

user - 执行操作的用户

desc - 更加详细的描述

time - 日志发生的时间

2.3.6.3 方法 [SI3]

toJSON() - 将对象转变为 JSON，准备写入数据库

store() - 首先调用该对象的 toJSON()方法，然后将其写入数据库

render(renderer) - 将该对象渲染为 html 代码。需要使用 renderer 对象

2.3.7 安全控制

为防止 post 请求导致对数据库的未授权的访问，在涉及到数据库访问的 post 请求中应用了一些安全协议。

2.3.7.1 security.js 导出一个过滤函数，如果访问条件相符，则返回 true，否则返回 false。[SI4]

2.3.7.2 函数接受一个 json 对象作为参数：data = {req: req, res: res, protocols = [] [, operator='AND'/'OR'] [, params = []]}

protocols 是一个字符串，指示使用的安全协议；也可以是一个数组，包括要使用的协议。如果是数组，则 operator 指示这些协议之间的逻辑关系。

params 包括协议需要的参数，注意协议和该协议的参数需要在对应数组的同一位置。[SI3]

2.3.7.3 提供的安全协议有：[SI3]

"admin": 仅允许管理员访问。

"level": 仅允许访问权限等于或高于参数值的用户访问。

"mlevel": 仅允许修改权限等于或高于参数值的用户访问。

"authenticated": 仅允许已登录的用户访问。

"idmatch": 仅允许参数中提供的 id 访问。

2.3.7.4 security.js 也导出 protocols 对象，可以通过该对象直接使用协议。[SI0]

2.3.8 数据库设置

本系统使用 Mongodb NoSQL 数据库。其集合（Collection）设置如下

2.3.8.1 Collection Access - 用户信息。每个集合条目应包含下列属性：[SI1]

id - 用户名

codehash - 密码 SHA1 值

clearance{} - 用户权限，包括

level - 访问等级，可以访问安全等级不高于该数值的文章

mlevel - 修改等级，可以修改安全等级不高于该数值的文章

admin - 是否为管理员

follows[] - 关注的用户

2.3.8.2 Collection Archive - 档案库。集合中应包含一个标签条目和若干文章条目。[SI3]

标签条目应包括：

type: 'tags'
tagname: [tag1,tag2,...]

文章条目应包括：

entrytitle - 文章标题

abstract - 文章摘要

content - 文章正文，以 html 格式呈现

accesslevel - 安全等级

exceptions[] - 例外，即访问等级不足也能访问的用户

time - 最近一次修改时间

tags - 文章的分类标签

authorid - 作者的 ObjectId

2.3.8.3 Collection Logs - 日志。每个集合的条目即为一个 Log 对象。详见 2.3.6 [SI3]

2.3 网页前端

2.3.1 页面布局

页面主要排版格式为导航栏与正文容器。

2.3.1.1 导航栏背景为黑色，固定于页面上方，不随页面位置移动。导航栏的主要栏目链接有（主页与图形数据网部分有不同的导航链接）：[SI4]

Main Cluster / 主序列 - 系统主页

Archive / 档案库 - 文章列表

Imaging Database Network / 图形数据网 - 用户登录/用户中心

System Main / 主序列 - 首页（图形数据网）

Entry Point / 起始序列 -用户登录/用户中心（图形数据网）

Archive / 档案 - 可访问的文章列表（图形数据网）

Access Manage / 权限管理 - 用户管理（图形数据网，管理员）

Log / 日志 - 查看日志（图形数据网，管理员）

Select Language / 选择语言 - 选择页面使用的语言（不包括文章）

NTWRK>> - 用户状态（用户名，登出）和系统状态（版本号），以及关于页面

2.3.1.2 正文容器相对整个页面居中，并且会根据屏幕大小实时调整宽度。根据请求的页面不同，正文的排版方式主要有以下几种 [SI4]

列表视图，如查看文章列表等。其特点是同一标签重复使用。

整体视图，如查看单个文章，编辑文章等。其特点是只有一个结构化的容器。

综合视图，如主页。其特点是多种排版方式混合使用。

2.3.2 层叠样式表

2.3.2.1 本网站使用的层叠样式表以 Twitter Bootstrap 为主，同时包括一些自定义的样式。

2.3.2.2 自定义的样式包括：[SI3]

.version - 应用于导航栏右上角显示的系统版本号

.logoHeader - 应用于主页的 logo <0-已弃用>

.idnlogoHeader - 图形数据网主页的 Logo

body - 改变页面背景

section.blogsArchive - 文章全文样式

article.blogPreview,

article.accessEntry - 列表条目样式

.exceptions_menu > li > a - 调整添加例外下拉框字体大小

.accesslevel_div > .add-on > a > .icon-remove - 调整移除图标透明度

img.brand - 调整导航栏图片缩进

.zh - 调整编辑为中文的段落字体

.en - 调整标记为英文的段落字体

i.icon-loading - 载入中动画图标

i.icon-loading .icon-white,

a:hover > .icon-loading - 载入中动画图标白色版，及高亮情况下的载入图标

2.3.3 使用的图像

2.3.3.1 本网站主要使用的是 png 格式的位图与 svg 格式的矢量图。svg 格式的矢量图需要较新的浏览器（非 IE7 或以下）支持。[SI3]

2.3.3.2 主页背景图 bg_dna.jpg 支持无缝衔接。[SI4]

2.3.4 JS 与 AJAX

本网站使用了 Twitter Bootstrap 与 jQuery，同时使用 ckeditor 作为用户发表文章时的富文本编辑器。根据不同页面的需要，不同功能的 js 代码分为不同的文件存储，并在视图中通过非阻塞性 js 调用。这些 js 文件被存储在 public/js/中，包括：

access_manage.js - 用户管理功能。包括批准用户申请、修改用户权限、删除用户等。通过 AJAX 完成这些工作。[SI3]

archive_edit.js - 文章编辑功能。用 AJAX 提交编辑过的文章，若提交成功，将跳转到该文章的页面。[SI3]

arhive_new.js - 创建文章。用 AJAX 提交创建的文章，若提交成功，将跳转到该文章的页面。[SI3]

authenticate.js - 使用 AJAX 进行登录，如果登录成功则刷新本页。已弃用，这部分的功能已经被整合到 idn.js 中。[SI0]

filter.js - 过滤器相关功能。使用 AJAX 获得分类标签，应用过滤器时重定向通过 GET query 对象请求服务器返回相关分类的条目。[SI3]

request.js - 用 AJAX 提交用户注册请求。[SI3]

idn.js - 使用 AJAX 进行登录，如果登录成功则刷新本页。

fetcher.js - 使用 AJAX 动态获取特定分类标签的文章预览。

profile-edit.js - 使用 AJAX 提交用户编辑信息。

profile_imgselect.js - 使用 AJAX 提交用户选择的头像。

register.js - 使用 AJAX 提交用户注册信息。