Hugo简介

Hugo vs Hexo

Hugo 是用 go 语言实现的一款静态网站生成器，速度比 **Hexo** 快得多。Hugo 只有一个二进制可执行文件，不用安装 go 就可以运行，与 Hexo 相比省事多了，Hexo 需要安装 nodejs 以及一大堆的 nodejs 包。

快速入门

1. 安装

Hugo 的安装很简单，去 **github** 上下载已经编译好的二进制文件，Linux/Mac 等系统可以放到 /usr/local/bin 下面，windows 可能需要修改系统环境变量中的 PATH。总之，能在终端执行 `hugo` 命令就可以了。

2. 生成站点目录

```
hugo new site xx.com
```

执行上面的命令后，会生成站点目录 xx.com，目录中包含如下内容：

```
archetypes/  config.toml  content/  resources/  static/  themes/
```

其中 `config.toml` 是站点配置文件，其余都是目录。

3. 添加主题

Hugo 不自带默认主题，可以去这里挑选一个主题，官方示例用的 ananke，下载后将 ananke 目录放到 `themes` 目录下，然后修改 config.toml 设置主题:

```
theme = "ananke"
```

4. 新建 markdown 文件

```
cd xx.com && hugo new notes/hugo.md
```

上面的命令会在 content/notes 目录下生成 hugo.md 文件。markdown 文件是根据 `archetypes` 目录下的模板文件生成的，可以按需要修改这些模板文件。

如果将 config.toml 中的 `permalinks` 注释掉，Hugo 就会按默认方式，根据文件路径生成 url，如 hugo.md 对应的 url 就是: `/notes/hugo/`。

5. 启动 hugo 自带的 http server

```
hugo server -D -p 8888
```

默认绑定的 ip 是 127.0.0.1，可以在浏览器中访问 127.0.0.1:8888 看效果。

6. 生成静态站点目录

```
cd xx.com && hugo
```

上面的命令会在 xx.com 目录下生成一个 public 目录，它包含了站点的全部静态内容，可以直接挂在 nginx 下面运行。

用 Mathjax 支持 LaTeX 数学公式

LaTeX 数学公式可以用 **Mathjax** 渲染，只需在网页中引入 Mathjax.js 就行了。但 markdown 与 LaTeX 有一个冲突：`_` 在 markdown 中表示强调，在 LaTeX 中表示下标，md 转换成 html 时，`_` 已经被替换掉了，所以等到 Mathjax 渲染数学公式时可能会出错。Hugo 官方及网上给出了一些解决方案，或多或少有些问题，此处修正了其中的一些缺陷，总结如下：

1. 新建 mathjax html 模板

在主题的 layouts/partials 目录下新建一个模板文件 mathjax.html，内容如下:

```
<script type="text/javascript" async
        src="//cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.5/MathJax.js?config=TeX-MML-AM_CHTML">
    MathJax.Hub.Config({
        //CommonHTML: { linebreaks: { automatic: true } },
        tex2jax: {
            inlineMath: [['$','$'], ['\\(','\\)']],
            displayMath: [['$$','$$']],
            processEscapes: true,
            processEnvironments: false,
            skipTags: ['script', 'noscript', 'style', 'textarea', 'pre'],
        }
    });
    MathJax.Hub.Queue(function() {
        var all = MathJax.Hub.getAllJax(), i;
        for(i = 0; i < all.length; i += 1) {
            var x = all[i].SourceElement().parentNode.className;
            if (x == '') {
                all[i].SourceElement().parentNode.className += 'has-jax';
            } else if (x.indexOf('has-jax') == -1) {
                all[i].SourceElement().parentNode.className += ' has-jax';
            }
        }
    });
</script>
```

上述代码，在 html 中引入 Mathjax.js。`Config` 函数对 Mathjax 进行配置，而 `Queue` 函数给含有数学公式的 html 元素加上 `class="has-jax"`，Mathjax 发现 `has-jax` 时就会渲染其中的数学公式。网上的代码有一个问题，会对一个 html 标签加多次 has-jax，此处修复了这个问题。

2. 在 footer 中加入 mathjax

修改主题 layouts/partials 目录下的 footer.html 文件，在适当的位置加入下面的一行：

```
{{ if .Params.mathjax }} {{ partial "mathjax.html" . }} {{ end }}
```

加上 `if .Params.mathjax` 判断条件，需要在 markdown 文件的 front-matter 区域加上 mathjax: true 启用 Mathjax。默认是不启用的，以免影响网页的加载速度，毕竟不是所有页面都含有数学公式。

3. 修改 css 样式文件

修改主题 static/css 目录下的 style.css 样式文件，加入下面的代码：

```
code.has-jax {
  font: inherit;
  font-size: 100%;
  background: inherit;
  border: inherit;
  color: #333;
}
```

上述代码是为了修复行内公式的样式，注意 color 一般设置成 html 默认的文本颜色。

4. 输入 LaTeX 数学公式

```
# 行内公式的输入
$ y = x^2 $                     # 无冲突正常输入
`$ x_1, x_2, \cdots, x_n $`     # 有冲突两边加上`

# 非行内公式，无冲突正常输入
$$ a^{p-1} \equiv 1 \pmod{p} $$  #

# 非行内公式，有冲突时放到 <div> 中
<div>
$$
s_1 \equiv s_0 \cdot a \pmod{m} \\\\
s_2 \equiv s_1 \cdot a \pmod{m} \\\\
\vdots \\\\
s_{m-1} \equiv s_{m-2} \cdot a \pmod{m}
$$
</div>
```