```
http://www.puppeteerjs.com/
```

```
let a = new URL('https://www.baidu.com') //解析url地址
```

使用axios

```
npm install axios
```

输出axios可以看到很多方法

最基本的爬虫

```
let a = 'https://www.qweather.com/business/map'
const axios = require('axios')
//或者request.get  首先引入request
axios.get(a)
.then(res=>{
    console.log(res);
})

let httpURL = 'https://www.baidu.com'
axios.get(httpURL,{
    headers:{'X-Requested-With':"XMLHttpRequest"},
}).then(function(res) {
    console.log(res)
})
```

浏览器打开网页时做了什么我们也要做什么 伪装成浏览器

爬取电影网站
//获取起始页面所有分类
//获取分类里的电影链接
//根据电影链接获取电影的详细信息

```
正则表达式爬取需要的内容
```

$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Mul 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。
( )	标记一个子表达式的开始和结束位置。子表达式可以获取供以 这些字符，请使用 \(和 \)。
*	匹配前面的子表达式**零次或多次**。要匹配 * 字符，请使用 *。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+。
.	匹配**除换行符 \n 之外**的任何单字符。要匹配 . ，请使用 \. 。
[	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式**零次**或**一次**，或指明一个非贪婪限定符。 请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、

	符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，当i表达式中使用时，表示不接受该方括号表达式中的字符集合。要身，请使用 \^。
{	标记限定符表达式的开始。要匹配 {，请使用 \{。
|	指明两项之间的一个选择。要匹配 |，请使用 \\|。

```
let reg = /<span></span>/
/let result = reg.exec(body)
//exec匹配内容 需要循环爬取
let reg = /<a href=".*?">.*?<\/a>/igs
    let arr = []
    let result
    while(  result = reg.exec(res.data)  )
    {
        arr.push(result[0])
                    //
    }
   console.log(arr)

   var str="<p>1213214312441</p><p>1213214312441</p><p>1我是你巴巴</p><p>sb41</p><p>12132
document.write(str.match(/<p>(.*?)<\/p>/g))

```

实验表明 match更好用

爬取时要注意 （）.等需要转义

---

cheerio插件 （以jquery为核心）

```
let cheerio = require("cheerio")
//axios获取res
let $ = cheerio.load(res.data)
axios.get(httpUrl).then(res=>{
    let $ =  cheerio.load(res.data)
```

```
    $(".col-sm-9 a").each((i,ele) =>{
        console.log($(ele).attr("href"))           //注意给ele加上$
        let title = ....find(".random_title").text()
    })
})
```

爬取表情包 图片的地址需要 http -> http

```

async function  parsePage(url) {
    let res = await axios.get(url)
    let $ =  cheerio.load(res.data)
    $(".artile_des table tbody tr td a img").each((i,ele) =>{
        let mesUrl = $(ele).attr("src")
        let newUrl = mesUrl.substr(0,4) + 's' + mesUrl.substr(4,mesUrl.length - 4 + 1)
        console.log(newUrl);
    })
}
```

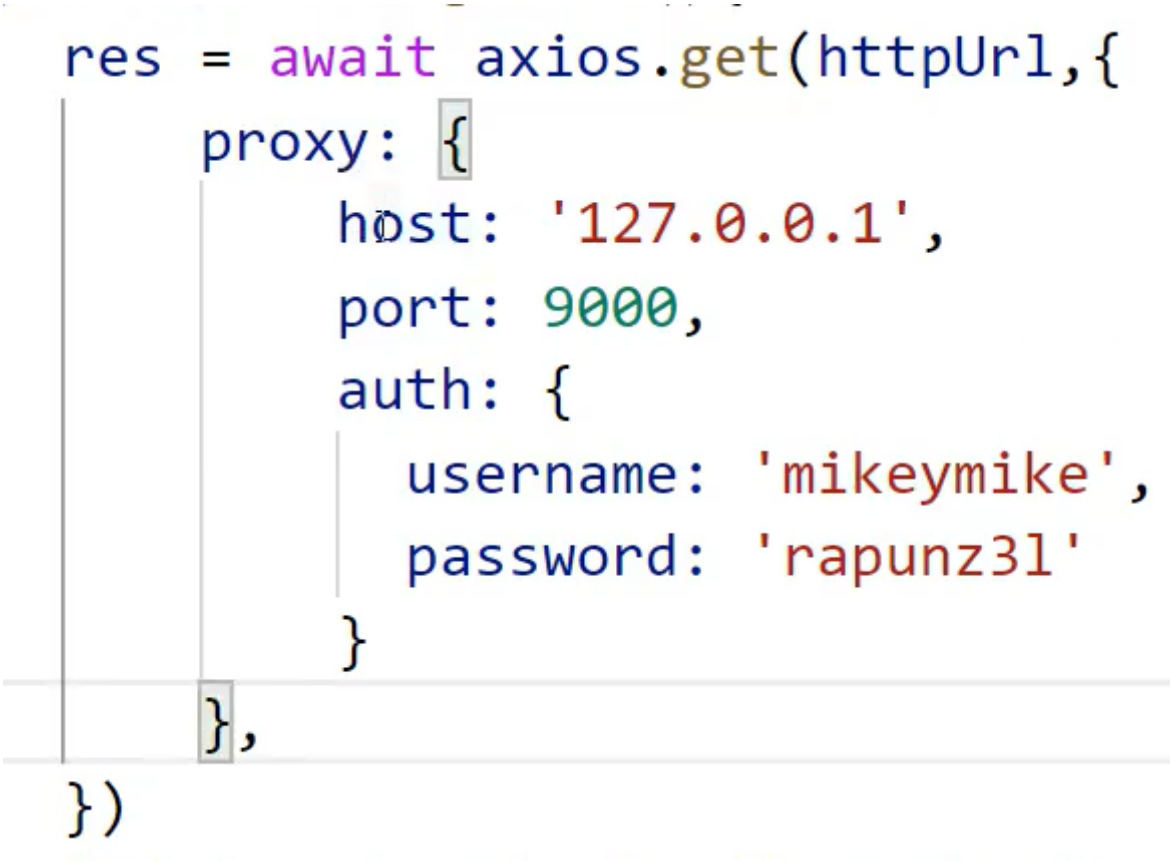
得到结果 放到本地文件

```
let ws = fs.createWriteStream('img/ / ')
axios.get(imgUrl,{responseType:'stream'}).then(function(){
    res.data.pipe(ws)//导入到写入流

})
```

```
axios.get后面可以直接.then
/(.*?)\d/igs //匹配文本
```

反爬虫



可以为axios设置代理

```
async function download(url,title) {
    let res = await axios.get(url,{responseType:"stream"})
    let wx = fs.createWriteStream('./mp3/' + 'title' + '.mp3')
    res.data.pipe(ws)
    res.data.on('close',function() {
        ws.close()
    })
}
```

追加

```
fs.writeFile("music.txt",content,{flag:'a'},function() {
    console.log("写入完成")
})
```

Puppeteer

无头浏览器

倾向于抓取SPA

```
let puppeteer = require('puppeteer')
async function test() {
    let option = {
        defaultViewport:{
            width:1400,
            height:800
        },                              //默认视图窗口
        headless:false,
        slowMo:1000, //放慢毫秒数
    }
    let browser = await puppeteer.launch(option) //可以配置为无界面浏览器 性能更高
    let page = await browser.newPage()
    await page.goto("https://www.baidu.com")
    await page.screeshot({path:"screenshot.png"})
    page.$$(selector) //获取选择的内容
    await page.$$eval("#menu li a",(elements)=>{
        elements.forEach((item,i)=>{
            console.log(item.innerHTML)
        })
    })
    page.on('console',function(...arg){
        console.log(args)
    })
}
test()
```

对Dom元素进行操作

```
 elementHandles = await page.$$("menu li a")
 elementHandles[2].click()
 inputEle = await page.$(".searchl .formhue")
 await inputEle.focus()
 await page.keyboard.type("小丑") //输入小丑
 let btnEle = await page.$(".....")
 await btnEle.click()
```

page.waitForSelector(