

# 应对多种页面类型的爬虫程序





# 目录

## 一、数据抓取

1. 简介

6. Selenium示例

12. 如何解释一个js语句

## 二、数据处理

## 三、其他

18. 如何识别Headless浏览

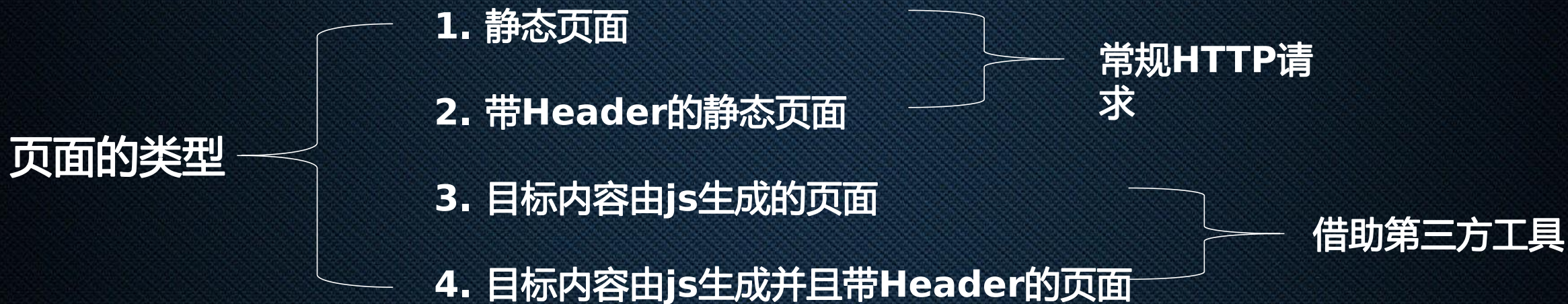
器

19. 如何识别验证码



# 一、数据抓取

## 1. 简介





# 一、数据抓取

# 2. 环境准备

## 模拟服务器 ( Node.js )

## 页面内容

```
var http = require('http')
var fs = require('fs')
var server = http.createServer((req,res) => {
  // 返回页面内容
  fs.readFile('./index.html', 'utf-8', (err,data) => {
    res.setHeader("Content-Type", "text/html");
    res.end(data);
  });
  // 打印请求中的Cookie信息
  console.log(req.headers.cookie)
})
server.listen(9000)
```

```
<!DOCTYPE html>
<html>
<head>
  <title>This is a title</title>
</head>
<body>

</body>
</html>
```



## 一、数据抓取

## 3. 抓取静态页面 ( Go - Colly )



```
func main() {  
    crawler := colly.NewCollector()  
    crawler.OnHTML("title", func(e *colly.HTMLElement) {  
        fmt.Println(e.Text)  
    })  
    crawler.Visit("http://localhost:9000/")  
}  
  
// This is a title
```

## 一、数据抓取

### 4. 带Header的静态页面 (Go - Colly)

```
func main() {  
    crawler := colly.NewCollector()  
    crawler.OnRequest(func(r *colly.Request) {  
        r.Headers.Set("Cookie", "name=smallyu")  
    })  
    crawler.OnHTML("title", func(e *colly.HTMLElement) {  
        fmt.Println(e.Text)  
    })  
    crawler.Visit("http://localhost:9000/")  
}
```



# 一、数据抓取

## 5. HTML改造 ( 加js )



```
<!DOCTYPE html>
<html>
<head>
  <title>This is a title</title>
</head>
<body>
  <div id="box"></div>
  <script>
    document.getElementById("box").innerHTML = "text"
  </script>
</body>
</html>
```



# 一、数据抓取

# 6. 动态数据

针对JavaScript生成的动态数据：

## ui4j ( Java )

- 支持有界面
- ( 据说 ) 支持无界面
- 依赖JavaFx

## Puppeteer ( js )

- 支持无界面
- 依赖Chromium

## Selenium ( 主流平台 )

### 1. 模拟浏览器

程序使用驱动调起本地浏览器，获取浏览器加载渲染后的页面

- ChromeDriver
- FirefoxDriver
- InternetExplorerDriver
- SafariDriver

### 2. 伪浏览器 ( 无头浏览器 )

程序直接对页面进行渲染

- PhantomJS Drive  
r
- HtmlUnitDriver



## 一、数据抓取

## 7. HtmlUnitDriver



```
public static void main(String[] args) {  
    HtmlUnitDriver wd = new HtmlUnitDriver(true);  
    wd.get("http://localhost:9000/");  
  
    System.out.println(wd.findElementById("box").getText());  
}
```



## 一、数据抓取

## 8. HtmlUnitDriver (带




```
HtmlUnitDriver wd = new HtmlUnitDriver(true) {
    @Override
    protected WebClient modifyWebClient(WebClient client) {
        URL url = null;
        try {
            url = new URL("http://localhost:9000/");
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
        client.addCookie("name=smallyu", url, null);
        return client;
    }
};
wd.get("http://localhost:9000/");

System.out.println(wd.findElementById("box").getText());
```



## 一、数据抓取

## 9. HTML改造（加异步）

```
  
<body>  
  <div id="box"></div>  
  <script>  
    new Promise(() => {  
      document.getElementById("box").innerHTML = "text"  
    })  
  </script>  
</body>
```



## 一、数据抓取

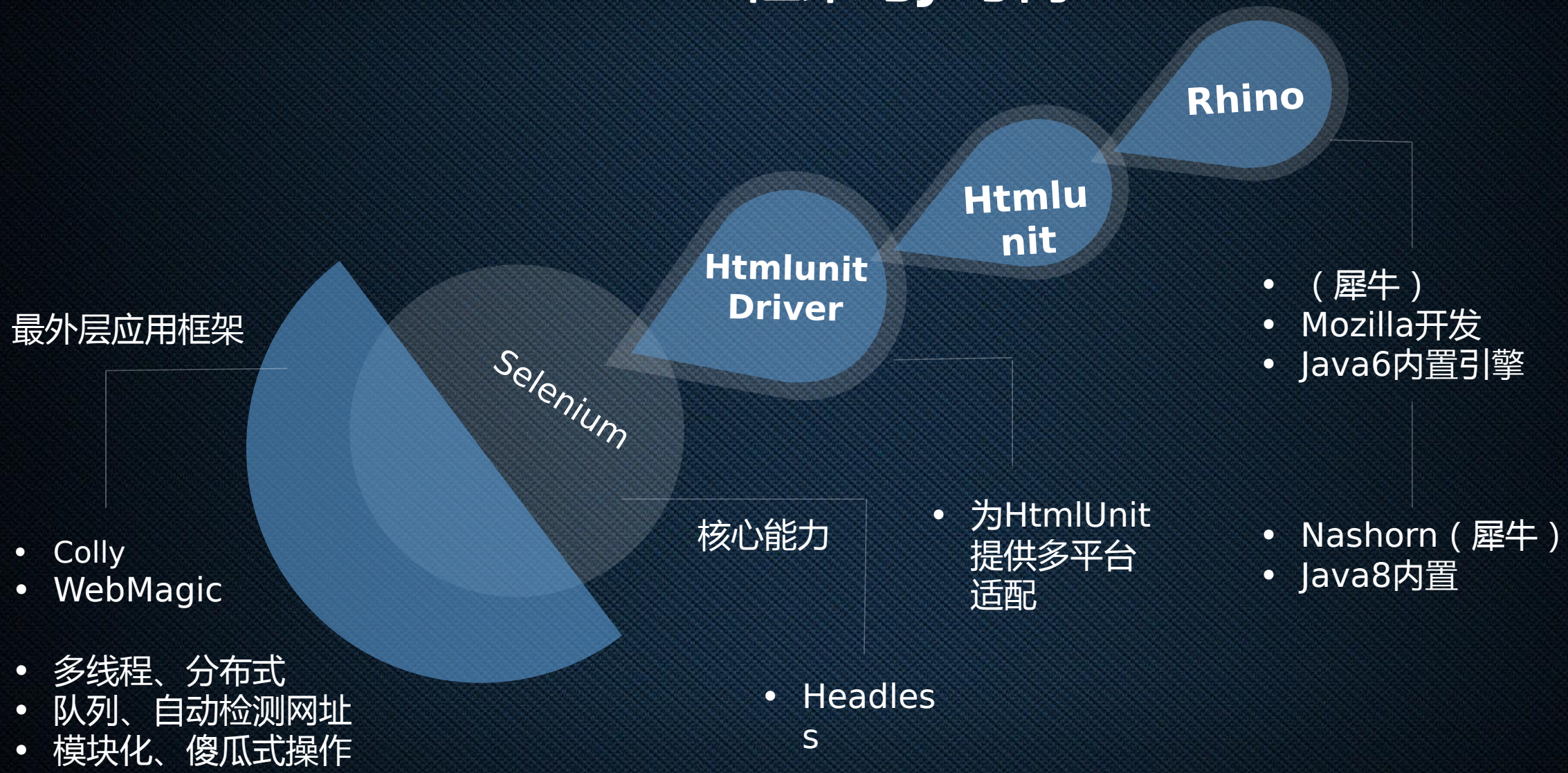
# 10. HtmlUnitDriver不支持

```
WARNING: All illegal access operations will be denied in a future release
2019-09-02 23:49:43 ERROR com.gargoylesoftware.htmlunit.javascript.StrictErrorReporter - error:
  message=[syntax error] sourceName=[script in http://localhost:9000/ from (8, 10) to (12, 11)]
  line=[9] lineSource=[      new Promise() => {}] lineOffset=[16]
Exception in thread "main" org.openqa.selenium.WebDriverException: com.gargoylesoftware.htmlunit
  .ScriptException: syntax error (script in http://localhost:9000/ from (8, 10) to (12, 11)#9)
Build info: version: '3.4.0', revision: 'unknown', time: 'unknown'
System info: host: 'DESKTOP-KEL8SR9', ip: '10.0.75.1', os.name: 'Windows 10', os.arch: 'amd64',
  os.version: '10.0', java.version: '11'
Driver info: driver.version: HtmlUnitDriver
  at org.openqa.selenium.htmlunit.HtmlUnitDriver.get(HtmlUnitDriver.java:688)
  at org.openqa.selenium.htmlunit.HtmlUnitDriver.lambda$8(HtmlUnitDriver.java:657)
  at org.openqa.selenium.htmlunit.HtmlUnitDriver.lambda$0(HtmlUnitDriver.java:414)
  at java.base/java.lang.Thread.run(Thread.java:834)
```



# 一、数据抓取

## 11. 框架与js引擎





# 一、数据抓取

## 12. 如何解释一个JavaScript语句

( 如何实现一个JavaScript引擎 )



# 一、数据抓取

## 13. 期望效果

输入：

```
⬢ ⬢ ⬢  
<div>1</div>  
<script>document.getElementById( ).innerHTML=2</script>
```



解释器处理

输出：



```
⬢ ⬢ ⬢  
<div>2</div>
```



# 一、数据抓取

# 14.1 词法分析

document . getElementByDiv() . innerHTML  
= 2

将语句分解为  
Token:

```
[
  Token{
    value='document',
    type='object'
  },
  Token{
    value='.',
    type='identifier'
  },
  Token{
    value='getElementByDiv()',
    type='method'
  },
  Token{
    value='.',
    type='identifier'
  }
]
```

```
[
  Token{
    value='innerHTML',
    type='field'
  },
  Token{
    value='=',
    type='assignment'
  },
  Token{
    value='2',
    type='value'
  }
]
```

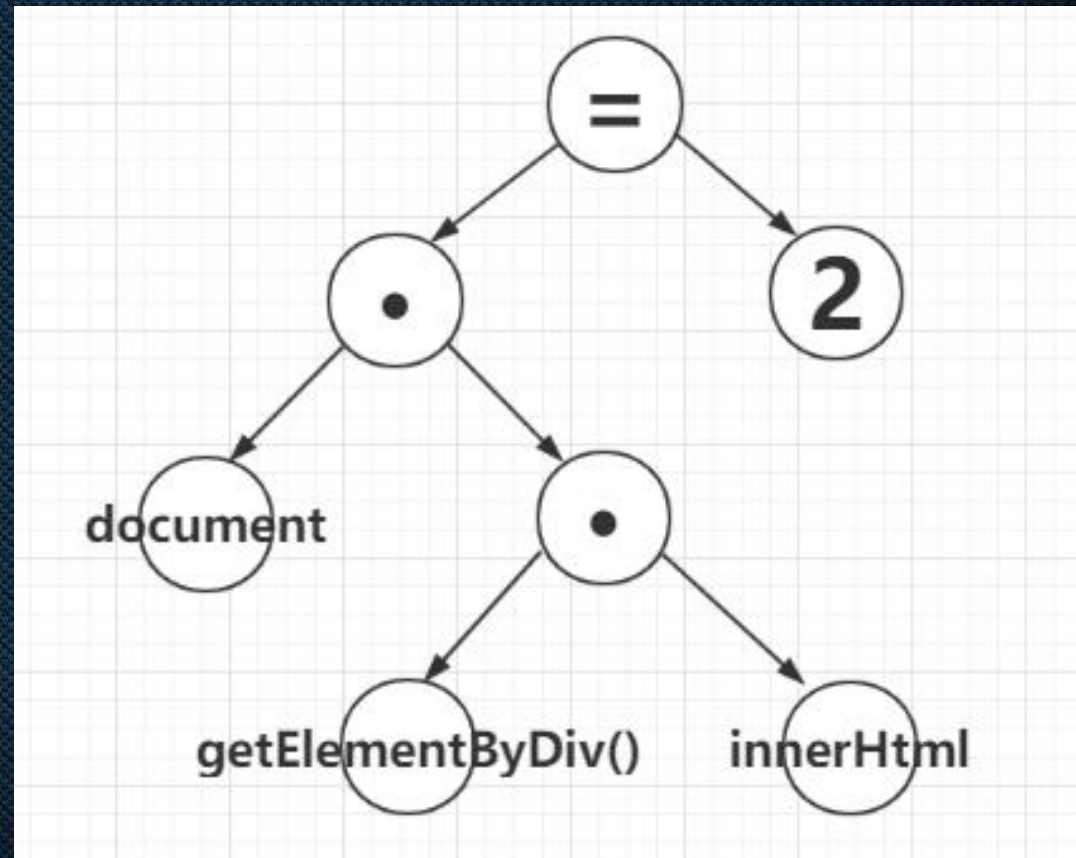


# 一、数据抓取

## 14.2 句法解析

`document` `.` `getElementByDiv()` `.` `innerHTML`  
= 2

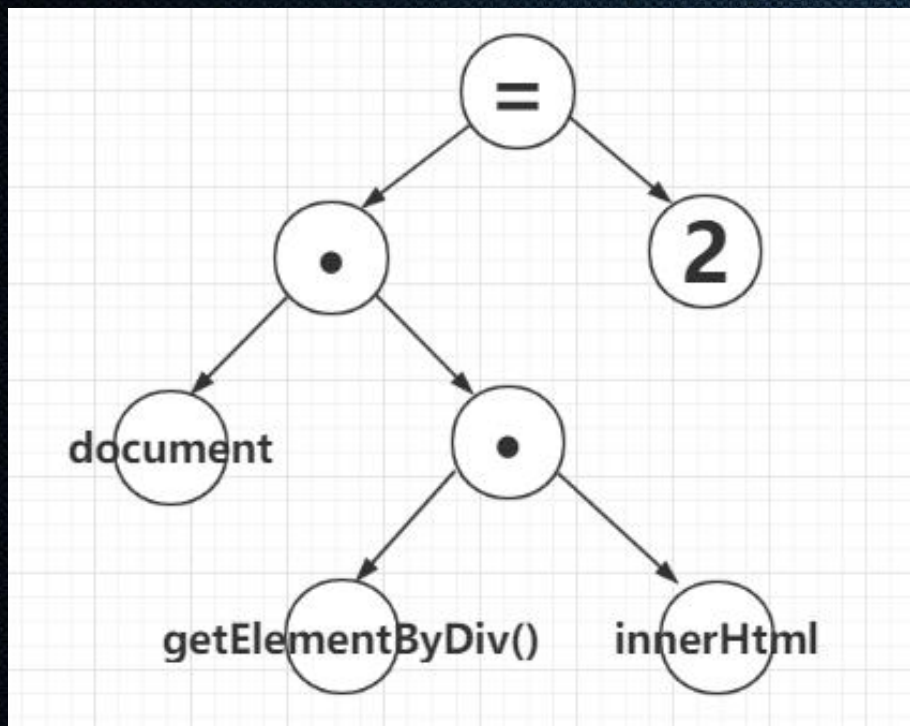
将Token解析为抽象语法树:





# 一、数据抓取

## 14.2 句法解析 - 数据结构



```
AST{
  operation='assignment',
  leftValue='null',
  rightValue='2',
  leftAST=AST{
    operation='identifier',
    leftValue='document',
    rightValue='null',
    leftAST=null,
    rightAST=AST{
      operation='identifier',
      leftValue='getElementByDiv()',
      rightValue='innerHTML',
      leftAST=null,
      rightAST=null
    }
  },
  rightAST=null
}
```



# 一、数据抓取

## 14.3 编译与运行

### 3. 生成字节码

将抽象语法树转为JavaScript引擎可以执行的二进制代码。目前，还没有统一的JavaScript字节码的格式标准

### 4. 解释字节码

读取并执行字节码



### 3. 直接渲染输出DOM



# 一、数据抓取

## 15.1 实体类



```
class Token {  
    String value;  
    String type;  
  
    public Token(String value, String type) {  
        this.value = value;  
        this.type = type;  
    }  
}
```



```
class AST {  
    String operation;  
    String leftValue;  
    String rightValue;  
    AST leftAST;  
    AST rightAST;  
}
```



# 一、数据抓取

## 15.2 主方法



```
public static void main(String[] args) throws ParserConfigurationException {
    var document = "<div>1</div><script>document.getElementById( ).innerHTML=2</script>";
    var docNodes = new ArrayList<String>();
    var scriptNodes = new ArrayList<String>();

    // 节点初始化
    nodesInit(document, docNodes, scriptNodes);
    // 词法分析
    var TokenList = lexer(scriptNodes);
    // 句法分析
    parser(TokenList, AST);
    // DOM渲染
    var ast = AST;
    interpreter(docNodes, ast);

    System.out.println(doc);
}
```



# 一、数据抓取

## 15.3 节点初始化

```
static void nodesInit(String document, ArrayList<String> docNodes, ArrayList<String> scriptNodes) {  
    var p = Pattern.compile("<[^>]+>[<]*<[^>]+>");  
    var m = p.matcher(document);  
    while (m.find()) {  
        if (m.group().startsWith("<script>")) {  
            scriptNodes.add(m.group());  
        } else {  
            docNodes.add(m.group());  
        }  
    }  
}
```



# 一、数据抓取

## 15.4 词法分析

```
static ArrayList<Token> lexer(ArrayList<String> scriptNodes) {  
    // 获取脚本内容  
    var script = scriptNodes.get(0).toString();  
    script = script.substring(8, script.length() - 9);  
  
    // 扫描词义单位  
    var tokens = new ArrayList<String>();  
    var p = Pattern.compile("[^.=]+|\\.|=");  
    var m = p.matcher(script);  
    while (m.find()) tokens.add(m.group());  
  
    // 词法分析  
    var TokenList = new ArrayList<Token>();  
    for (var token : tokens) {  
        switch (token) {  
            case ".": TokenList.add(new Token(token, "identifier")); break;  
            case "document": TokenList.add(new Token(token, "object")); break;  
            case "getElementById()": TokenList.add(new Token(token, "method")); break;  
            case "innerHTML": TokenList.add(new Token(token, "field")); break;  
            case "=": TokenList.add(new Token(token, "assignment")); break;  
            case "2": TokenList.add(new Token(token, "value")); break;  
        }  
    }  
    return TokenList;  
}
```



# 一、数据抓取

## 15.4 句法解析

```
static String flag = "assignment";
static void parser(List<Token> TokenList, AST AST) {
    var TokenListSize = TokenList.size();
    for (var i = 0; i < TokenListSize; i++) {
        var Token = TokenList.get(i);
        // 根据操作符生成AST
        if (Token.value.equals("=") && (flag.equals("assignment"))
            || Token.value.equals(".") && (flag.equals("identifier"))) {
            AST.operation = Token.type;
            // 左侧AST递归
            if (i > 1) {
                flag = "identifier";
                AST.leftAST = new AST();
                parser(TokenList.subList(0, i), AST.leftAST);
            } else {
                AST.leftValue = TokenList.get(i - 1).value;
            }
            // 右侧AST递归
            if (TokenListSize - i - 1 > 1) {
                flag = "identifier";
                AST.rightAST = new AST();
                parser(TokenList.subList(i + 1, TokenListSize), AST.rightAST);
            } else {
                AST.rightValue = TokenList.get(i + 1).value;
            }
            break;
        }
    }
}
```



# 一、数据抓取

## 15.5 渲染输出

```
static String doc = "";
private static void interpreter(ArrayList docNodes, AST ast) {
    if (ast.operation.equals("assignment")) {
        // 等号左边递归
        if (ast.leftAST != null) {
            interpreter(docNodes, ast.leftAST);
        }
        // 等号右侧递归 TODO
    }
    if (ast.operation.equals("identifier")) {
        // 点号左侧递归
        if (ast.leftAST == null) {
            if (ast.leftValue.equals("document")) {
                doc = docNodes.get(0).toString();
            }
            if (ast.leftValue.equals("getElementByDiv()")) {
                doc = docNodes.get(0).toString();
            }
        }
    } else {
        // 我们的示例不会进到这个分支
    }
    // 点号右侧递归
    if (ast.rightAST == null) {
        if (ast.rightValue.equals("innerHTML")) {
            doc = doc.replaceAll(">.*<", ">" + AST.rightValue + "<");
        }
    } else {
        interpreter(docNodes, ast.rightAST);
    }
}
}
```



# 一、数据抓取

## 15.6 日志

```
Run: Interpreter x [Settings] [Close]
```

```
D:\Application\Java\jdk11\bin\java.exe -Didea.launcher.port=24118 -Didea.launcher.bin
.path=D:\Application\IntelliJ IDEA 2018.1.3\bin -Dfile.encoding=UTF-8 -classpath
"D:\tradingarea\untitled\out\production\untitled;D:\Application\IntelliJ IDEA 2018.1
.3\lib\idea_rt.jar" com.intellij.rt.execution.application.AppMainV2 Interpreter
[Token{value=' document', type=' object'}, Token{value='.', type=' identifier'},
Token{value=' getElementByDiv()', type=' method'}, Token{value='.', type=' identifier'},
Token{value=' innerHTML', type=' field'}, Token{value='=', type=' assignment'},
Token{value=' 2', type=' value'}]
AST{operation=' assignment', leftValue=' null', rightValue=' 2',
<leftAST=AST{operation=' identifier', leftValue=' document', rightValue=' null',
<leftAST=null, rightAST=AST{operation=' identifier', leftValue=' getElementByDiv()',
<rightValue=' innerHTML', leftAST=null, rightAST=null}}, rightAST=null}
<div>2</div>

Process finished with exit code 0
```



## 二、数据处理

## 16. 简介

10000 条数据

1. 抓取数据
2. 储存数据
3. 分析整理
4. 储存数据
5. 数据可视化



10000^10000 条数据

1. 摄入 (Insight)
2. 储存数据
3. 分析整理
4. 储存数据
5. 数据可视化

————— HDFS

MapReduce / Spark (Flink)

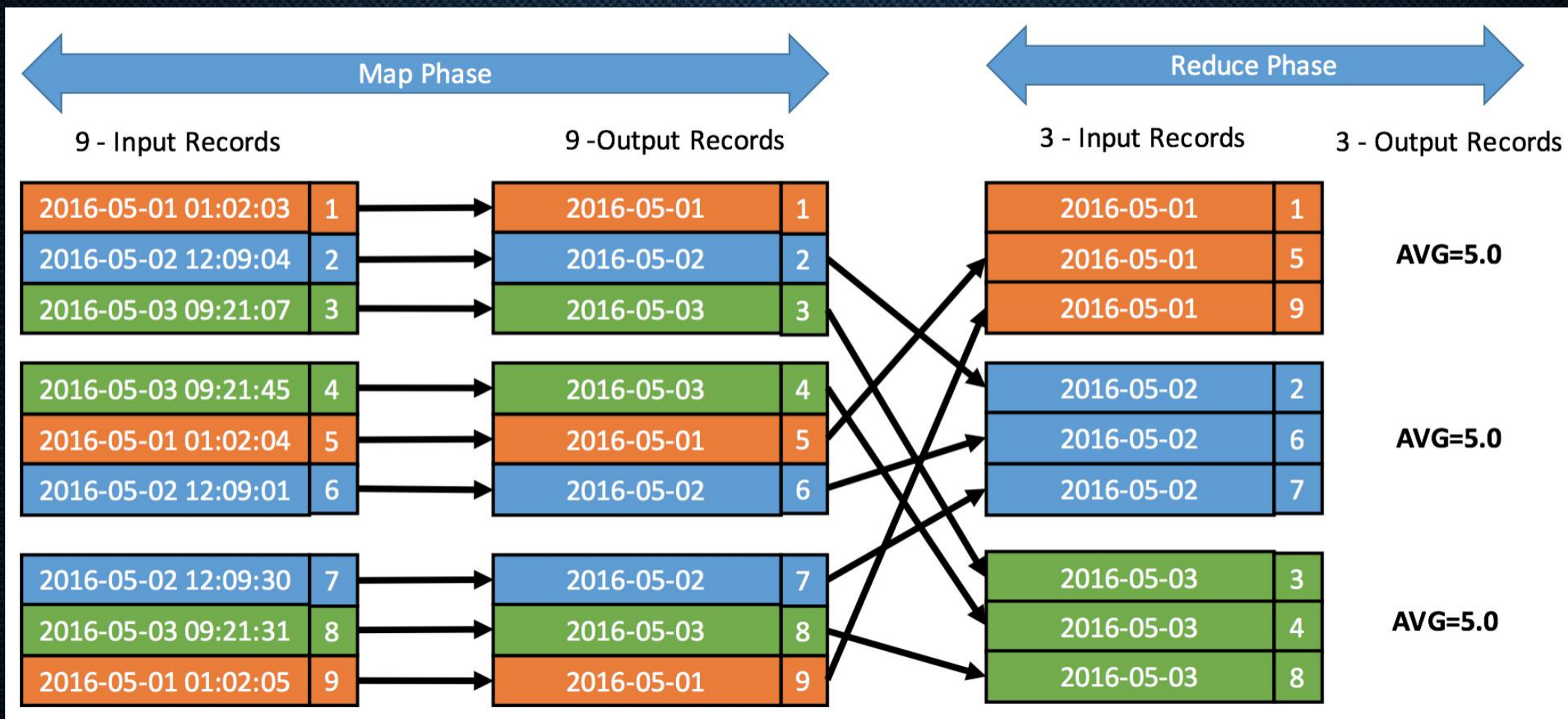
————— 存回HDFS

Yarn



## 二、数据处理

# 17. MapReduce





### 三、其他

## 18. 如何区分浏览器与Headless浏览器

#### 1. 检测 User-Agent ( 容易被伪装 )

```
if (/HeadlessChrome/.test(window.navigator.userAgent))  
{  
    console.log("Chrome headless detected");  
}
```

#### 2. 是否包含浏览器插件 ( 容易误判 )

```
if(navigator.plugins.length === 0) {  
    console.log("It may be Chrome headless");  
}
```



## 三、其他

# 18. 如何区分浏览器与Headless浏览器

3. 借助「WebGL API 获得的参数信息」检测
4. 通过「判断浏览器是否支持某些功能」进行检测
5. 借助「加载失败的图片」检测
- .....



### 三、其他

## 18. 如何区分浏览器与Headless浏览器

针对Puppeteer的一种方法：

Puppeteer定义请求：

```
page.setExtraHTTPHeaders({ 'Accept-Language': 'en-US,en;q=0.9' })
```

实际上发送的内容：

```
accept-language: en-US,en;q=0.9
```

原文：<https://news.ycombinator.com/item?id=20480915>



## 三、其他

### 19. 遗留问题：如何识别验证码



# 附：基于Java的爬虫框架WebCollector ( PDF )



基于Java的爬虫框  
架WebCollector.pdf



感谢大家

---

