

“高端语言”？





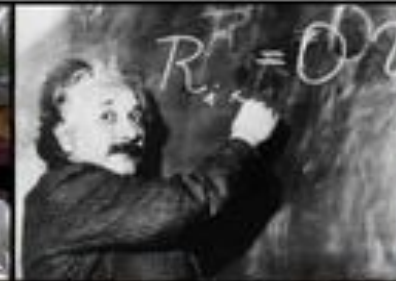











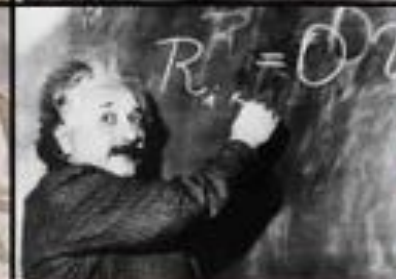





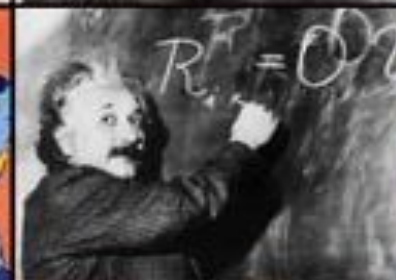









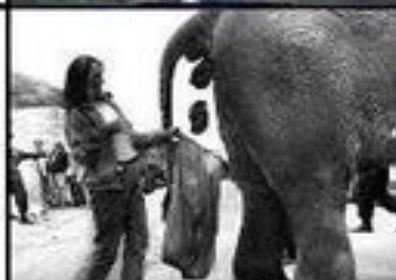



@k1ic

chenkai116@gmail.com

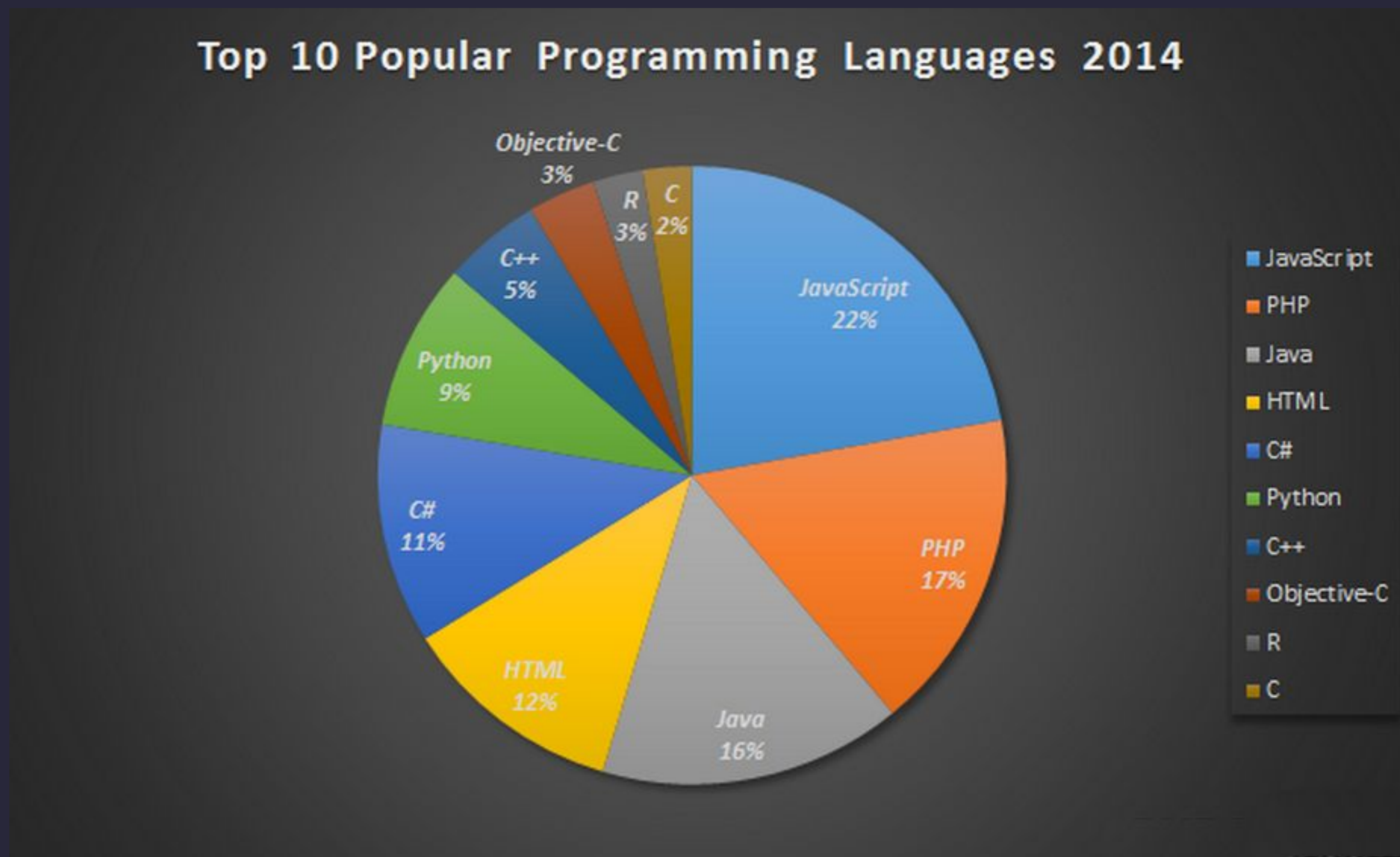
缘起

- 搞Java的不要转C++
- 第一门编程语言选什么
- Ruby、Python、JavaScript初学者抱怨语言太低端

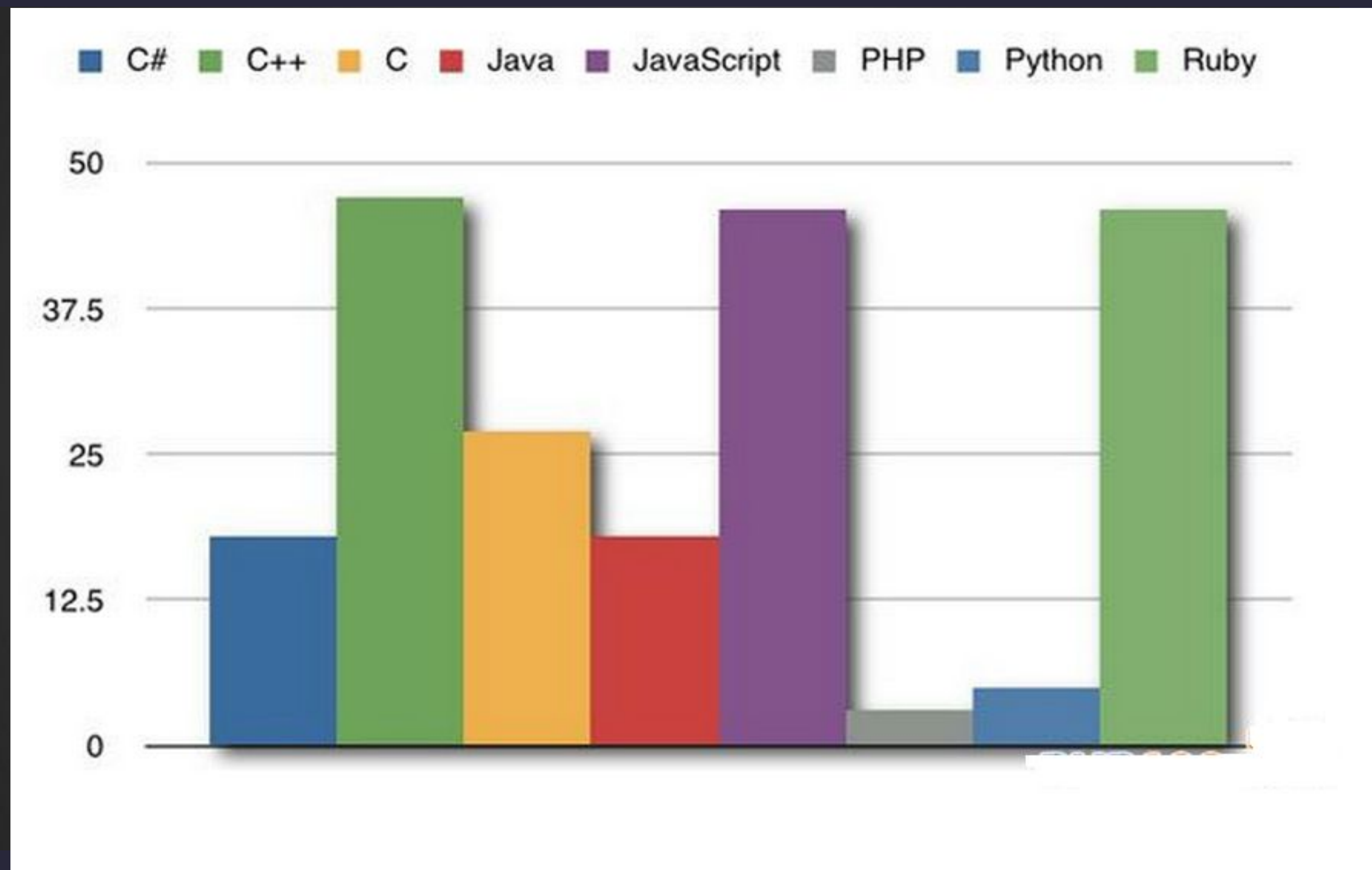
大家都很自负

Java	C	PHP	Ruby	Haskell	Lisp	在下列人的眼里
						Java fans
						C fans
						PHP fans
						Ruby fans
						Haskell fans
						Lisp fans

2014年被讨论最多的语言 JavaScript



2014年被吐槽最多的语言 C++



到底有没有 “高端” 语言？

六种编程语言读数据库

PHP

```
1  <?php
2      $mysql_server_name="localhost";
3      $mysql_username="root";
4      $mysql_password="123456";
5      $mysql_database="test";
6
7      $connection = mysql_connect($mysql_server_name, $mysql_username,$mysql_password);
8      if(!$connection) {
9          echo "connection failed!";
10         return;
11     }
12     mysql_set_charset("gbk",$connection);
13     mysql_select_db($mysql_database, $connection);
14     $sql="select * from test";
15     $result=mysql_query($sql, $connection);
16     while($row = mysql_fetch_array($result)) {
17         echo "|".$row["id"]."|".$row["name"]."|\n";
18     }
19     mysql_close($connection);
20 ?>
21
```


Python

```
1 # coding=utf-8
2 import MySQLdb
3 import sys
4
5 host = 'localhost'
6 user = 'root'
7 password = '123456'
8 db = 'test'
9
10
11 if __name__ == '__main__':
12     connection = MySQLdb.connect(host,user,password,db);
13     try:
14         connection.ping()
15     except:
16         print ('failed to connect MySQL.')
17     sql = 'select * from test'
18     cursor = connection.cursor()
19     cursor.execute(sql)
20     for row in cursor:
21         print ("|" + str(row[0]) + "|" + row[1] + "|")
22     cursor.close()
23     connection.close()
24     sys.exit()
25
```


C

```
1 #include "c_mysql.h"
2
3 #define HOST "localhost"
4 #define USERNAME "root"
5 #define PASSWORD "123456"
6 #define DATABASE "test"
7
8 int main()
9 {
10     char *sql = "select * from test";
11     execute_sql(sql);
12     return 0;
13 }
14
15 void execute_sql(char* sql)
16 {
17     MYSQL connection;
18     MYSQL_RES *result_pointer;
19     MYSQL_ROW result_row;
20     int result, row, column, i, j;
21     mysql_init(&connection);
22     if (NULL == mysql_real_connect(&connection, HOST, USERNAME, PASSWORD, DATABASE, 0, NULL, CLIENT_FOUND_ROWS))
23     {
24         printf("Error:connection failed!\n");
25         return;
26     }
27     mysql_query(&connection, "set names gbk");
28     result = mysql_query(&connection, sql);
29     if (result)
30     {
31         printf("Error:query failed!\n");
32         mysql_close(&connection);
33         return;
34     }
35     result_pointer = mysql_store_result(&connection);
36     if (result_pointer)
37     {
38         row = mysql_num_rows(result_pointer);
39         for (i = 1; i < row + 1; i++)
40         {
41             result_row = mysql_fetch_row(result_pointer);
42             printf("|%s|%s|\n", result_row[0], result_row[1]);
43         }
44     }
45     mysql_close(&connection);
46     system("pause");
47 }
```

```
1 #ifndef C_MYSQL_H_
2 #define C_MYSQL_H_
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <winsock2.h>
7 #include <windows.h>
8 #include <mysql.h>
9
10 void execute_sql(char* sql);
11
12 #endif
13
```


C++

```
1 #include "c++_mysql.h"
2
3 #define HOST "localhost"
4 #define USERNAME "root"
5 #define PASSWORD "123456"
6 #define DATABASE "test"
7
8 int main()
9 {
10     const SQLString sql = "select * from test";
11     execute_sql(sql);
12     return 0;
13 }
14
15 void execute_sql(const SQLString sql)
16 {
17     mysql::MySQL_Driver *driver;
18     Connection *connection;
19     Statement *statement;
20     ResultSet *result_set;
21     driver = mysql::get_mysql_driver_instance();
22     connection = driver->connect("tcp://localhost:3306", "root", "123456");
23     statement = connection->createStatement();
24     statement->execute("use test");
25     statement->execute("set names gbk");
26     result_set = statement->executeQuery(sql);
27     while(result_set->next())
28     {
29         cout << "|" << result_set->getInt("id") << "|" << result_set->getString("name") << "|" << endl;
30     }
31     delete statement;
32     delete connection;
33     system("pause");
34 }
35
```

```
1 #ifndef C__MYSQL_H_
2 #define C__MYSQL_H_
3
4 #include <iostream>
5 #include <mysql_connection.h>
6 #include <mysql_driver.h>
7 #include <statement.h>
8 using namespace sql;
9 using namespace std;
10
11 void execute_sql(const SQLString sql);
12
13 #endif
14
```

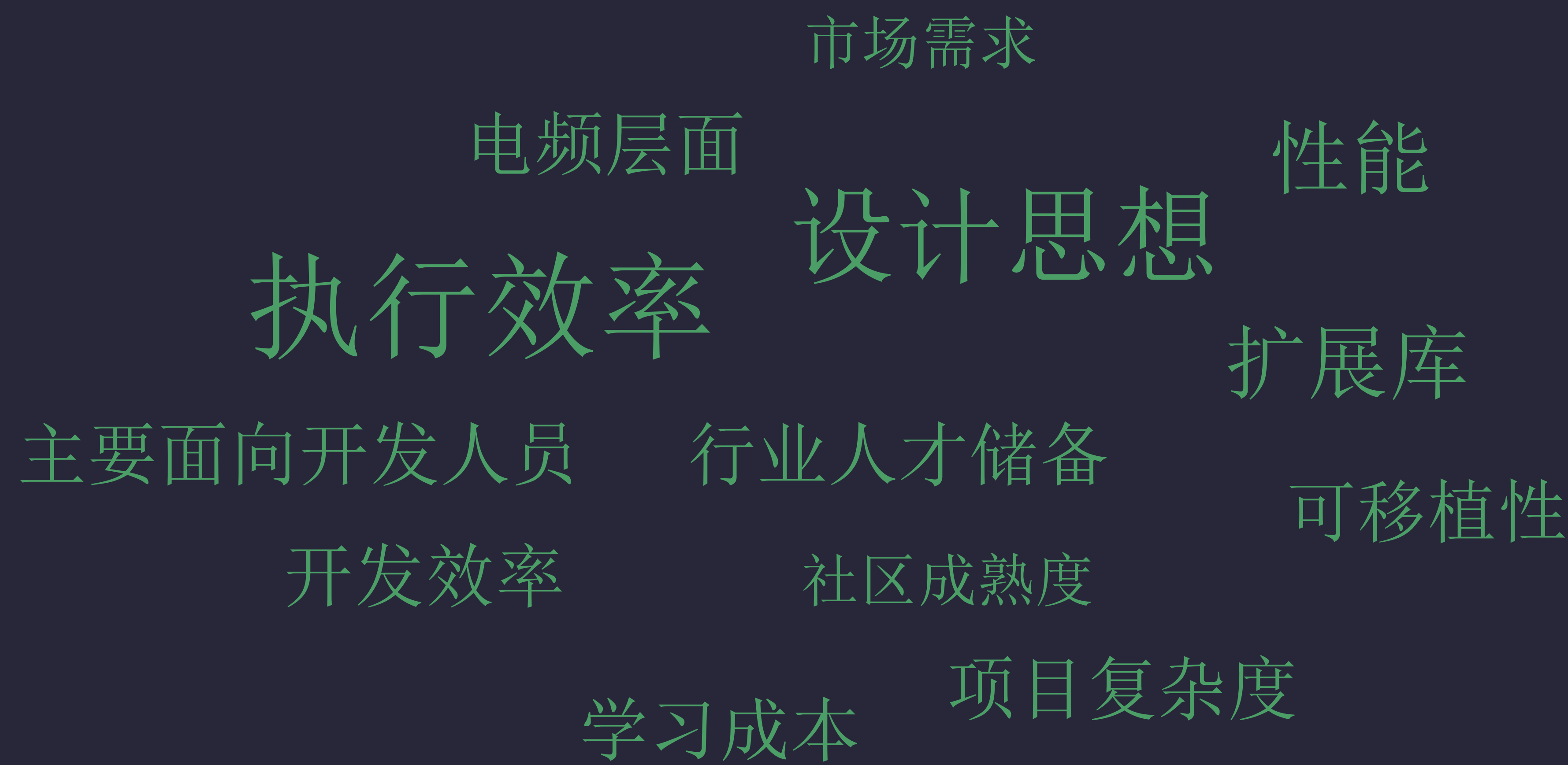

C#

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using MySql.Data.MySqlClient;
6
7 namespace CSMysql
8 {
9     class Program
10     {
11         static void Main(string[] args)
12         {
13             MySqlConnection connection = new MySqlConnection("Database='test';Data Source='localhost';User Id='root';Password='123456';charset='utf8';pooling=true");
14             MySqlCommand command = new MySqlCommand();
15             command.Connection = connection;
16             command.CommandText = "select * from test";
17             try
18             {
19                 command.Connection.Open();
20                 MySqlDataReader reader = command.ExecuteReader();
21                 while (reader.Read())
22                 {
23                     Console.WriteLine("|" + reader.GetInt32("id") + "|" + reader.GetString("name") + "|");
24                 }
25                 Console.ReadLine();
26             }
27             catch (Exception)
28             {
29                 Console.WriteLine("query failed!");
30             }
31             finally
32             {
33                 command.Connection.Close();
34             }
35         }
36     }
37 }
38
```


JAVA

```
1 package cn.zxl.jmysql;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.ResultSet;
6 import java.sql.Statement;
7
8 public class JMySQL {
9
10     private static final String DRIVER = "com.mysql.jdbc.Driver";
11     private static final String URL = "jdbc:mysql://localhost/test";
12     private static final String USERNAME = "root";
13     private static final String PASSWORD = "123456";
14     private static final String SQL = "select * from test";
15
16     public static void main( String[] args ) {
17         Connection connection = null;
18         Statement statement = null;
19         ResultSet resultSet = null;
20         try {
21             Class.forName(DRIVER);
22             connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
23             statement = connection.createStatement();
24             resultSet = statement.executeQuery(SQL);
25             while (resultSet.next()) {
26                 System.out.println("|" + resultSet.getString("id") + "|" + resultSet.getString("name") + "|");
27             }
28         } catch (Exception e) {
29             System.out.println("query failed!");
30         } finally {
31             try {
32                 resultSet.close();
33                 statement.close();
34                 connection.close();
35             } catch (Exception e) {
36                 throw new RuntimeException(e);
37             }
38         }
39     }
40
41 }
42
```

编程语言面面观



执行效率

执行效率 之 执行机制

解释执行 vs 编译执行

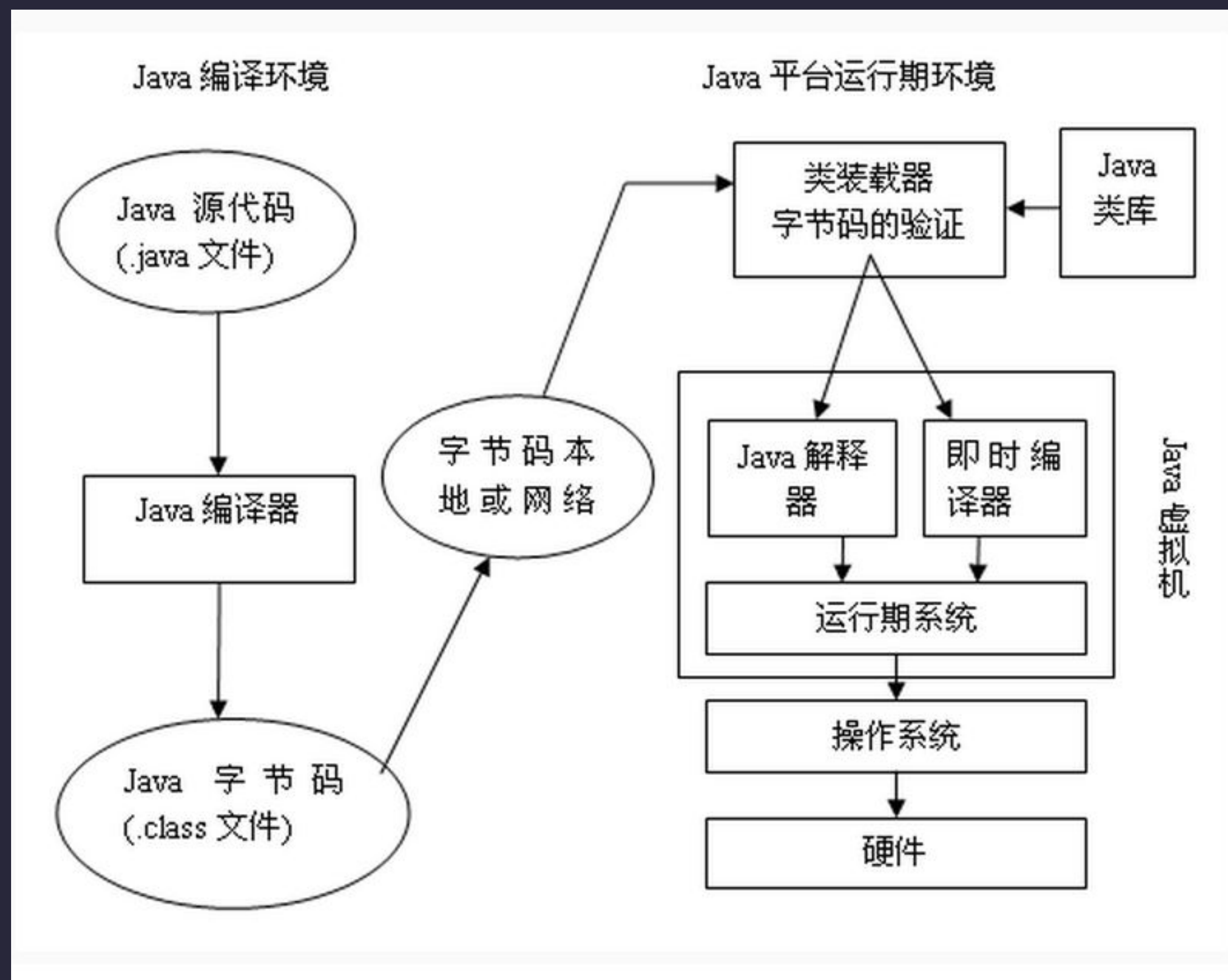
解释执行

PHP - 一次编译，一次执行

- 调用`zend_compile_file()`函数完成词法分析、语法分析
（输入`php`源文件，输出`op_array`即`op code`）
- 调用`zend_execute()`函数，执行`op code`（`op`命令共`150`种）
`zend`引擎命令分发方式：`call`（默认）、`switch`、`goto`
效率：`goto` > `switch` > `call`

编译执行

JAVA - 一次编译，永久执行

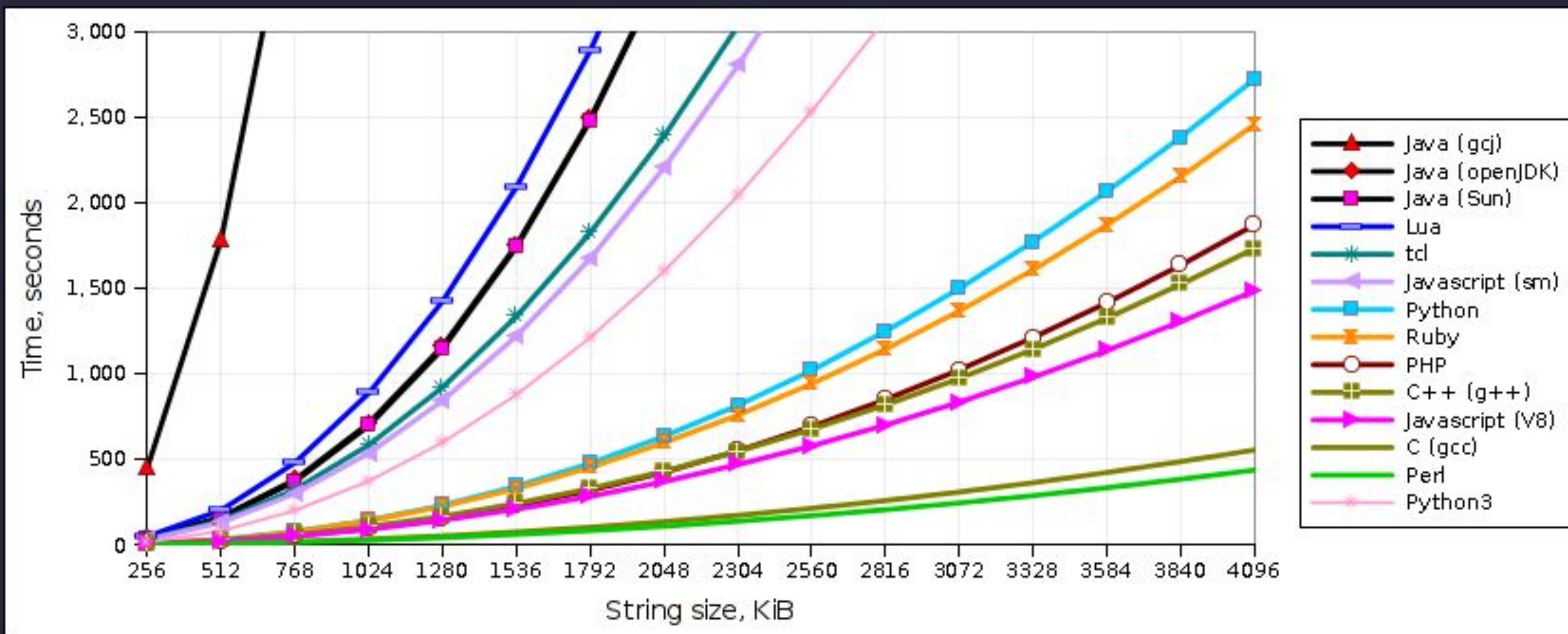


对比（同一个请求运行100次）

- 对于java，只要一次编译即可重复执行，所以其损耗为：
1次编译 + 100次执行
- 对于php，其每次都要编译+执行，所以其损耗为：
100次编译 + 100次执行

执行效率 之 文本处理

各种语言处理文本的速度



由于执行时间易度量，象C一样 “快速” 的语言
被认为是 “高端” 的编程语言

但这是片面的

C和更底层语言，在计算机硬件发展迟缓且系统资源受限的时代，是不可或缺的

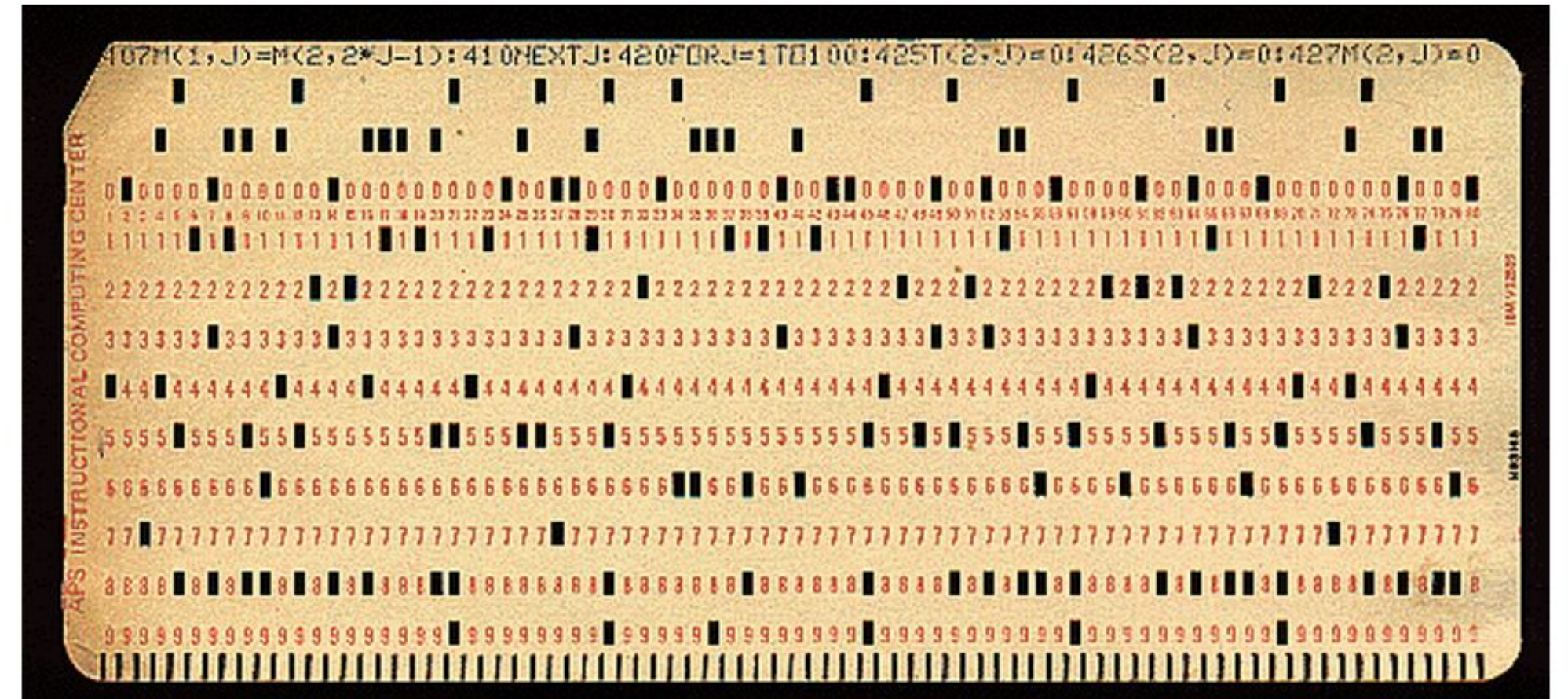
然而，当下相对于我们的需求，计算能力是充足且廉价的。软件开发的瓶颈在于开发人员的时间和精力

更少的代码做更多的事情，以提升开发效率

电频层面

操作数字传感器和电动机

```
00 2007 # jump past sensor and constant locations
01 0000 # read right sensor value here
02 0000 # read left sensor value here
03 0000 # write right motor power level here
04 0000 # write left motor power level here
05 0000 # store motor-off constant here
06 0100 # store motor-on constant here
07 4011 # load right sensor value into register 1
08 4022 # load left sensor value into register 2
09 1123 # subtract 1 from 2 and store result in 3
10 4105 # load motor-off constant into register 1
11 4206 # load motor-on constant into register 2
12 3316 # if the left sensor is greater than the
13 5203 # right then turn the right motor on
14 5104 # and turn the left motor off
15 2007 # and then jump to beginning of the loop
16 5204 # else turn the left motor on
17 5103 # and turn the right motor off
18 2007 # and then jump to beginning of the loop
19 0000 # this location is not used by the program
```



编程语言用更具可操作性的方式，将我们的想法转化为机器语言，差异仅为转换方式的不同

实现相同功能，不同语言的差别仅在于 “电频转换” 过程，结果无差别

设计思想

设计思想 之 设计目标

- C: 接近机器语言, 良好移植性
- C++: 高执行效率, 直接操作硬件而不引入额外开销
- PHP: Web开发、网页胶水? (Swoole)
- sh: OS跟HardWare交互的命令集
- js: 供浏览器执行, 增加网页动态特性

多样化需求 ➡ 多样化设计目标

设计思想 之 语法特色

pipe *vs* chaining

pipe

```
1 /* elixir pipe "|>" */
2 list_data = Store.get_host(host)
3 map = to_map(list)
4 formatted_output = tabularize(map)
5 IO.puts(formatted_output)
6
7 host
8 |> Store.get_host
9 |> to_map
10 |> tabularize
11 |> IO.puts
12
13 =====
14
15 /* shell pipe "|" */
16 log_name=${res_log_pre}"_push_msg_create_raw.log";
17 raw_data="cat "${log_names}" | grep push/message/create | grep FS_INFO_API | grep -v \ "^$" > "${raw_
18
19 log_name=${res_log_pre}"_push_msg_create_raw.log";
20 raw_data="cat "${log_names}"
21 | grep push/message/create
22 | grep FS_INFO_API
23 | grep -v \ "^$" > "${raw_log_name}";
24
```


chaining

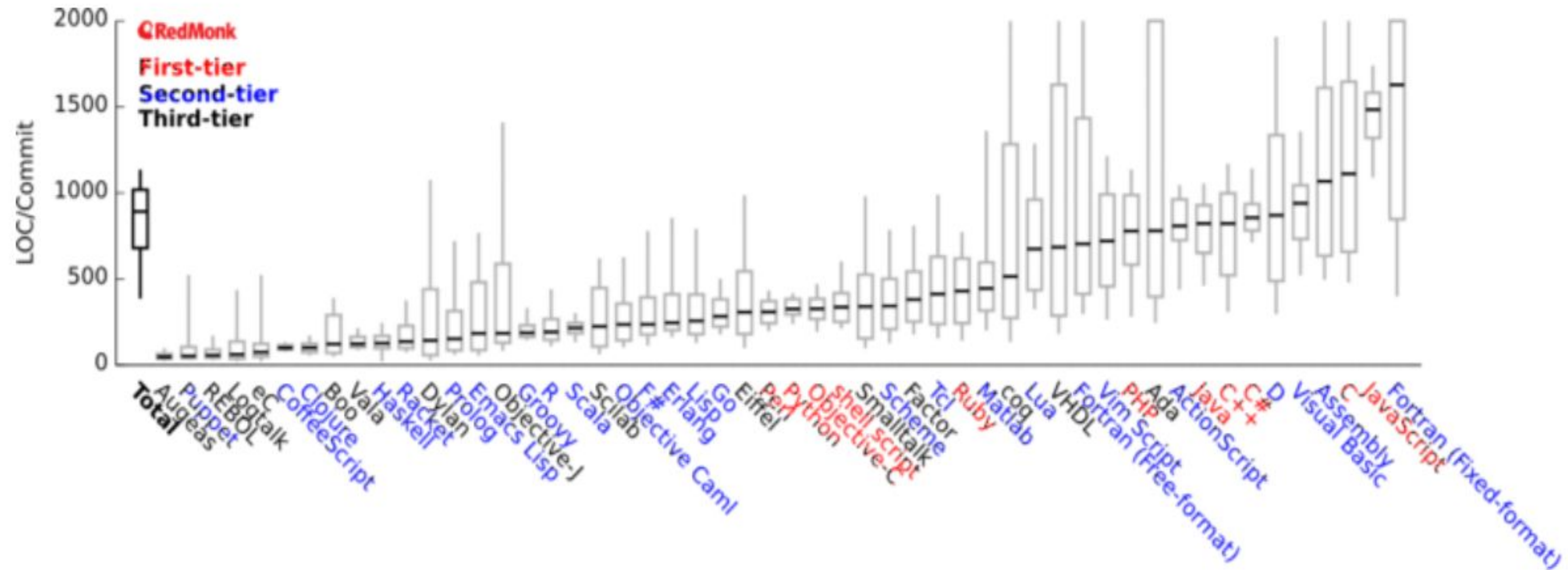
```
27
28 /* jQuery chaining "." */
29 $.ajax("test.html").done(function(){ alert("success");} ).fail(function(){ alert("error"); } ).done(f
30
31 $.ajax("test.html")
32     .done(function(){ alert("success");} )
33     .fail(function(){ alert("error"); } )
34     .done(function(){ alert("second callback");} );
35
```

表述方式类似，但设计思想不同

- pipe针对数据，chaining针对对象
- pipe返回数据，chaining返回对象
- 部分chaining操作有顺序依赖

复杂度

编程语言代码复杂度统计



C++的语法较PHP更严谨

但是C++的代码复杂度、开发周期、可读性均比PHP差

学习门槛低 == 低端？

学习门槛低 == 低端？

学习曲线陡峭 == 高端？

多元化



“高端语言” 是个伪命题

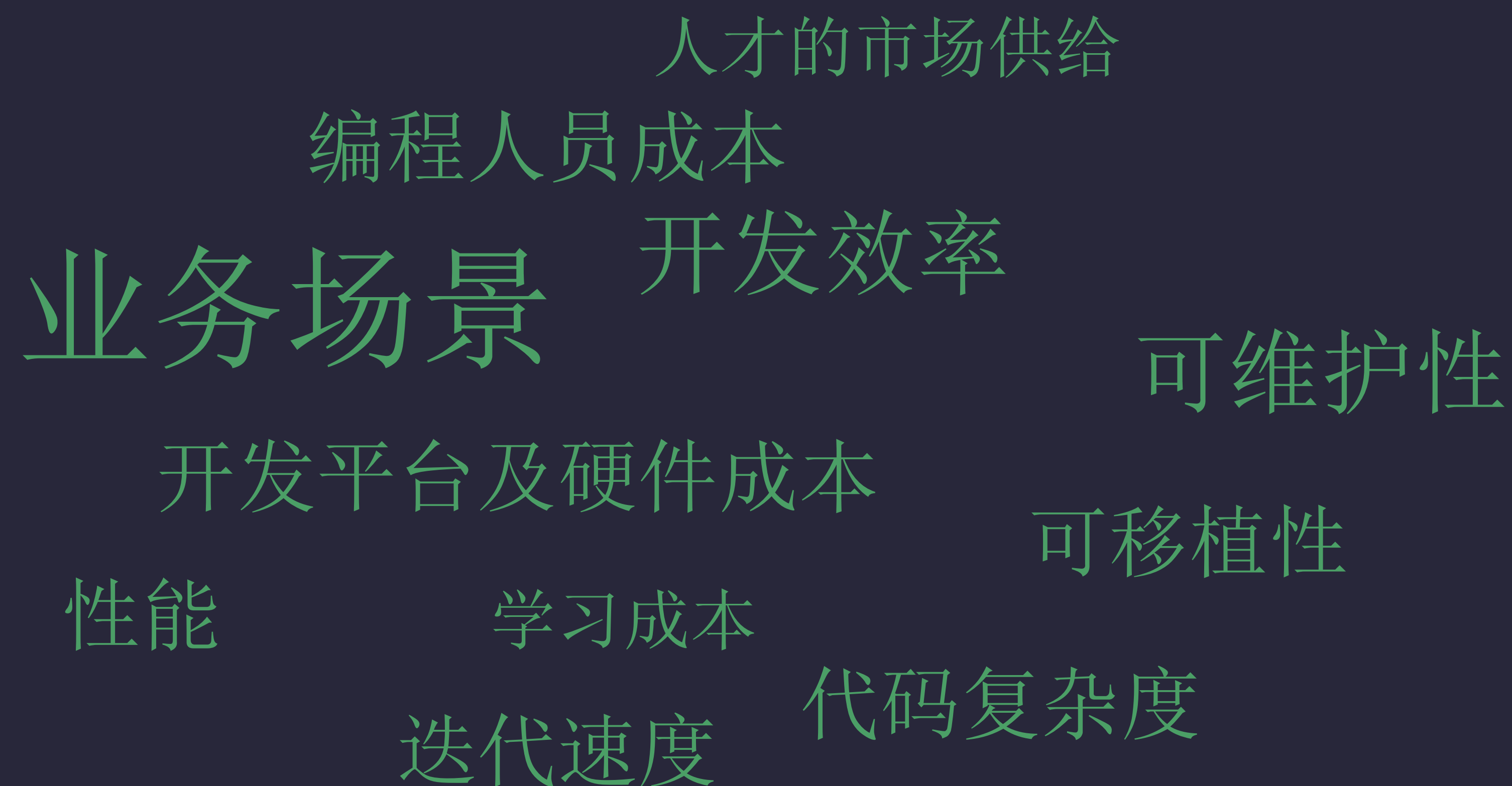
问题的答案取决于你的需要

- 统计学家和数据分析师需要像R语言那样擅长统计计算和作图的语言，做统计分析
- Web开发者需要像PHP或Python那样注重开发效率的语言，满足快速增长业务需求

让项目决定编程语言

- 需要的多样性，决定编程语言多样性
- 选择能够为你提供最快捷、最有效帮助的语言

技术选型



H5的兴起 --> 效率、成本

前端工程师走俏 --> 成本

前后端分离(MV*) --> 成本

Android、IOS App开发热门 --> 商业模式

后端程序员三座高峰

大数据、高并发、实时更新

行业走势

- 深度精细化分工
- 语言学习门槛进一步降低
- 简单、重复劳动（套模板）被消灭

PHP程序员的核心竞争力

- 计算机底层知识
- 解决问题的能力
- 工程化思维、结构化思维、产品意识、商业意识

- 理论基础（进程、网络、内存、文件系统）
- 语法背后的思想，设计背后的初衷（解决问题的思路）
- 工程化思维、结构化思维、产品意识、商业意识

Q&A

THE END