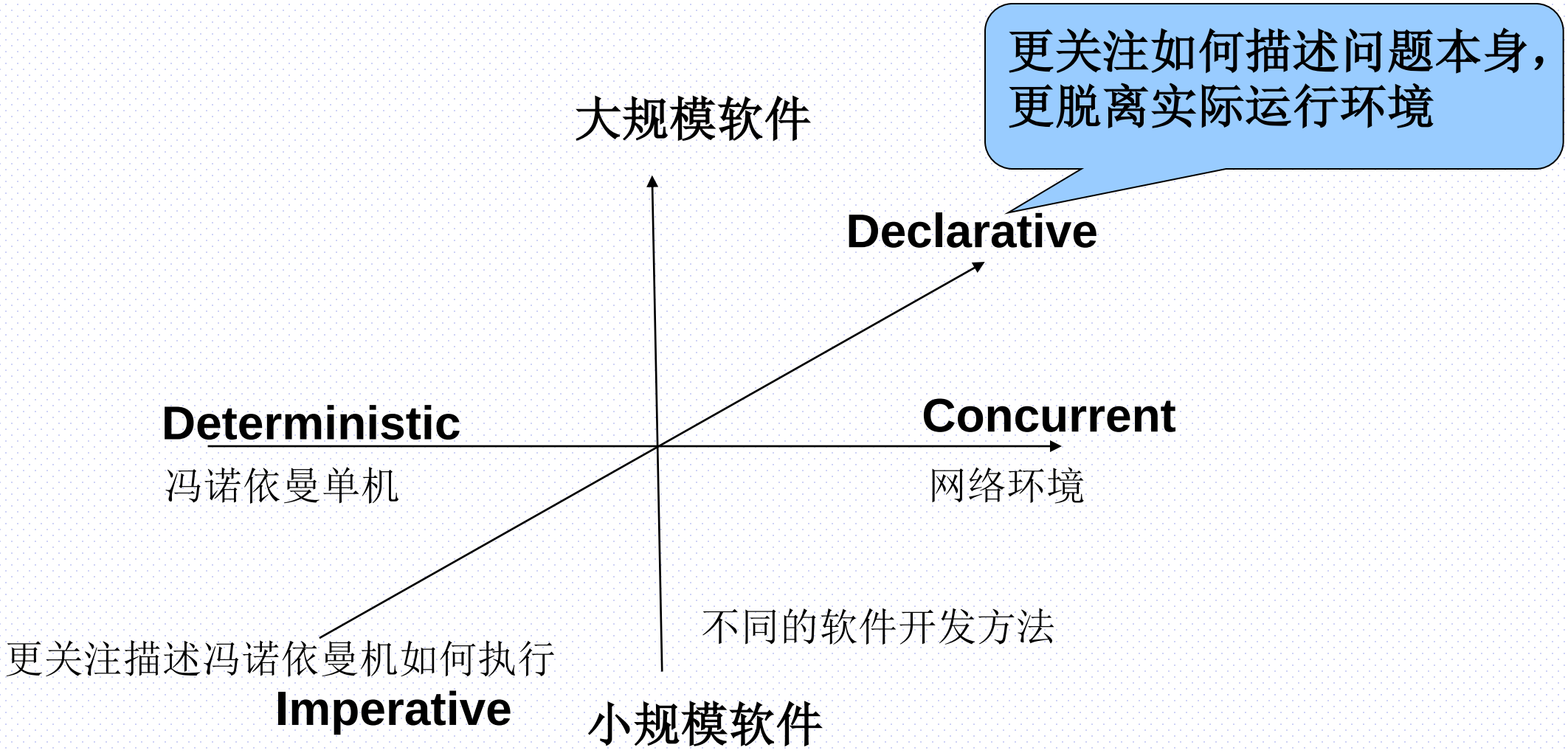
描述性程序设计语言

DECLARATIVE LANGUAGES

DECLARATIVE LANGUAGES



DECLARATIVE LANGUAGES

- 走出去：某一个环境下编写的程序要能在其他环境下执行。
- 请进来：其他环境下编制的程序要能为我的环境所使用。

走出去

- 平台无关（编程语言）
- 放置标示
- 脚本语言
- 服务计算语言
- 本体描述语言

走出去

- 源程序经过编译生成的目标码是与本地机紧密相关的。表现在：
  - 基本类型定义与及其字长相关；
  - 各机指令集不一，操作效果相当，实现过程有微小的差别；
  - 同一语言在相同编译和执行环境下，同一程序执行速度不一（硬件差异，优化次数）；

平台无关

- 可移植性(portability)-为减少开发费用和开发时间。
  - 程序员可移植：用户会了某种语言到任何能实现该语言的环境上即可编制程序而不需其它知识；
  - 程序可移植：在该语言的某个环境上编制的程序拿到有该语言实现的另一个环境上可以照样运行，程序的计算，语义不变。
- 采用增加可移植手段
  - 设预定义环境（包）——Ada，支持环境(APSE)；
  - 分出头文件 - 宏 - 编译文件——C；

平台无关

- 网络计算的兴起使可移植性上升为平台无关性。
- 局域网时代，需要在局域网内实现信息共享，有协作计算需求。
  - 网络协议栈实现了数据的平台无关；
  - 以文件共享形式实现；
  - 客户/服务器模式，程序可以不共享；
  - 只要局域网内使用相同或相互兼容的平台，也可以实现程序的共享。

平台无关

- Internet时代，对资源共享提出了新的要求：
  - 局域网的规模和结构是可控制的；但是，没有任何个人和机构可以控制Internet的构成。
  - WWW(World Wide Web)是Internet上最广泛的信息发布/浏览方式。与FTP等以往的方式相比较，WWW具有较强的动态性和交互性，需要完成复杂的应用（如电子商务等）。因此，共享资源不但包括数据，也应包括程序。这就需要一种平台无关的语言。
  - 激烈的竞争迫使软件生产者不断降低开发成本，缩短开发周期。平台无关语言使开发者不需要为一个软件开发多个操作系统的版本，减少了开发费用和时间。

平台无关

- 平台无关语言（编程语言）的实现：
  - 传送源代码
    - 将源代码传送到目标机，先经过编译，生成目标机代码，再执行。前提是必须有不同平台的编译器，这种情况往往在编译时缺乏源代码原有的环境信息（全程量），且只能先编译存入目标码再运行。局域网时代服务器已经做过了。要实时运行只能是解释（也要求有完整的信息）型语言。
  - 传送目标代码
    - 只有在相同或相互兼容的平台之间才可以实现。如在Windows NT/9x组成的局域网中，可以在一台机器上调用执行其它机器上的应用程序。
  - 传送中间代码
    - 中间代码由源代码经过编译生成。中间代码经过优化。中间代码传送到目标机上由解释器解释执行。Java语言使用的是这种方式。

JAVA对平台无关性的支持方法?

- 保留了高级语言的主要机制
  - 与C++基本相同的字符集、标识符、关键字、运算符、特殊符号
  - 比C++更加强化类型。增加了布尔类型，只保留数组类型；使用真正的类型转换（C++中的类型转换是“伪”的，目的是为了通过编译器的检查；Java中的类型转换在运行时真正发生，如果不能转换，系统会抛出异常）
  - 保留声明、作用域、变量、表达式、语句、三种结构化控制；
  - 保留并增强C++的异常；
  - 与C++相似的面向对象机制：类作为特殊类型、构造函数、初始化序列、实例变量、this/super、方法的覆盖与重载、成员可见性。

- 取消了C++中的部分机制：
  - 指针。以对象引用代替指针；
  - 取消头文件和预处理器（宏与机器相关）；
  - C++中原有冗余：struct、union都被类代替；
  - 不支持模板，动态束定功能可实现；
  - 取消typedef；
  - 不支持运算符重载；

平台无关

- 编译解释执行。

  Java语言实现平台无关的关键是使用了“编译-解释”执行方式。

  Java源代码经过编译，称为Java字节代码(byte-code)。

  Java虚拟机(VM)是字节代码的解释器。JavaVM是用软件构造的一个虚拟计算机，它由虚拟的寄存器、内存、堆栈等；字节代码就是这台虚拟计算机的指令。

  所有操作系统上的JavaVM执行一致的指令，这样，就屏蔽了各个平台之间的差异。

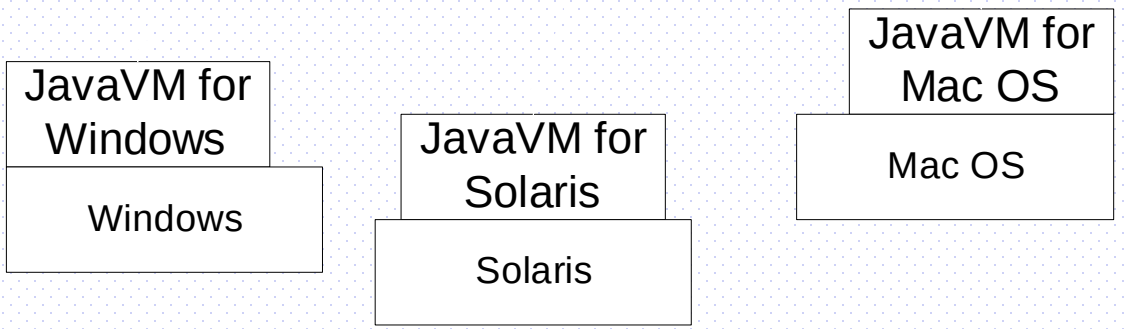
JAVA的跨平台

- 对下跨OS
  - 在不同的操作系统上有不同的Java虚拟机，向上有一致的接口（虚拟机的指令——字节代码），向下针对不同的操作系统有不同的实现方式。
- 对上跨语言
  - Clojure——Lisp
  - Jython——Python
  - Jruby
  - Scala
  - Koltin
  - Groovy
  - ~~J#~~

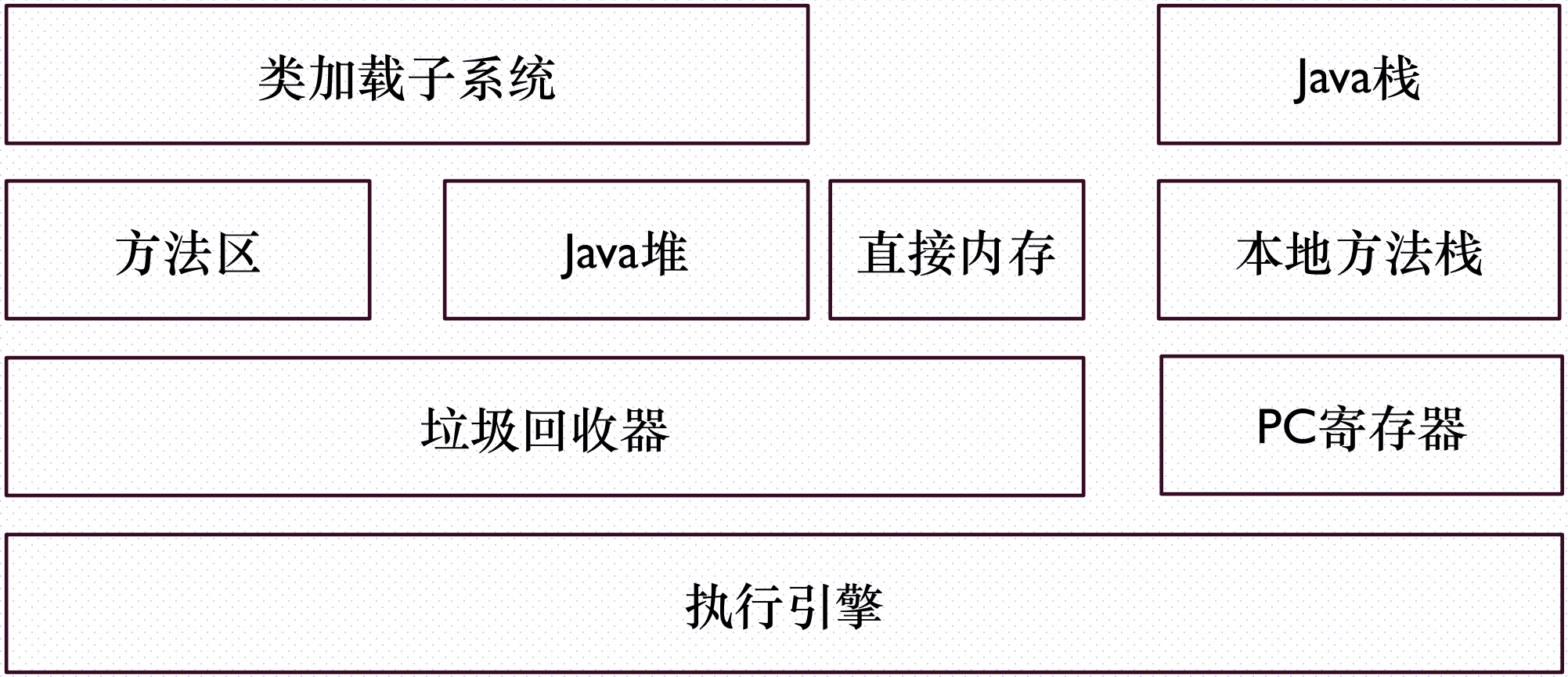
Utf-8

Java字节代码

8位字节二进制流
操作码：256个
操作数：运行时合成

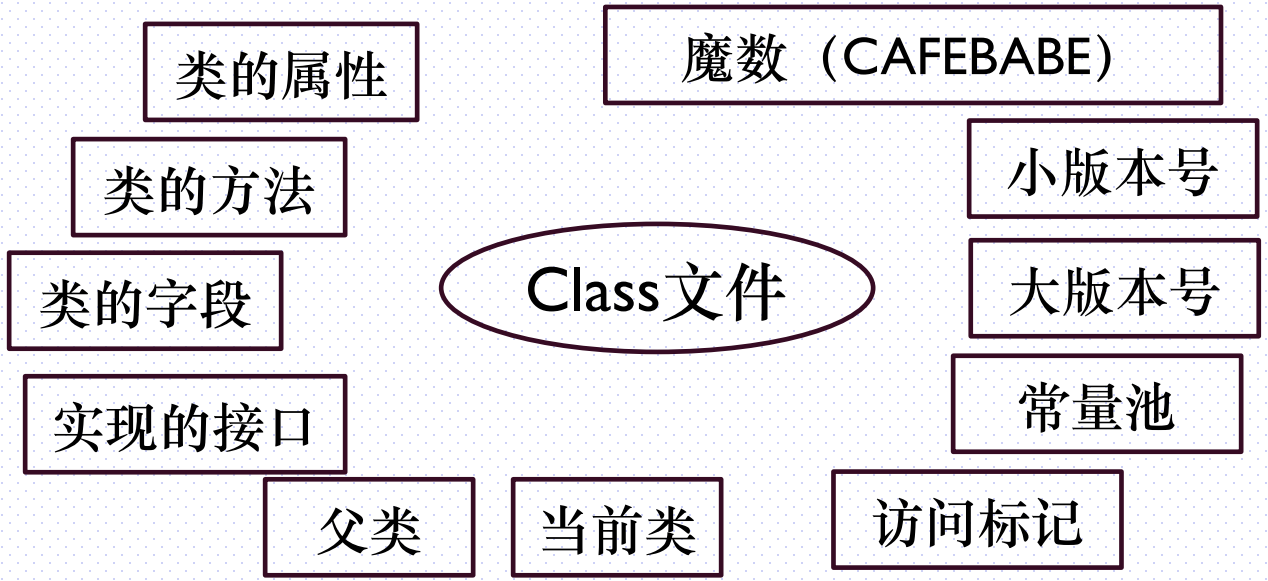


虚拟机的基本架构



CLASS文件

- 对于Java虚拟机，Class文件是虚拟机的一个重要接口。
- 任何语言→Class文件→可在Java虚拟机上执行
- Class文件的基本结构与解析：




```
ClassFile{
  u4         magic;
  u2         minor_version;
  u2         major_version;
  u2         constant_pool_count;
  cp_info constant_pool[constant_pool_count-1];
  u2         access_flags;
  u2         this_class;
  u2         super_class;
  u2         interfaces_count;
  u2         interfaces[interfaces_count];
……
}
```

操作字节码

- ASM体系结构
- ASM是一款Java字节码的操作库
- <<interface>> Opcodes
- ClassReader
- ClassVisitor
  - ClassWriter
- FieldVisitor
  - FieldWriter
- MethodVisitor
  - MethodWriter

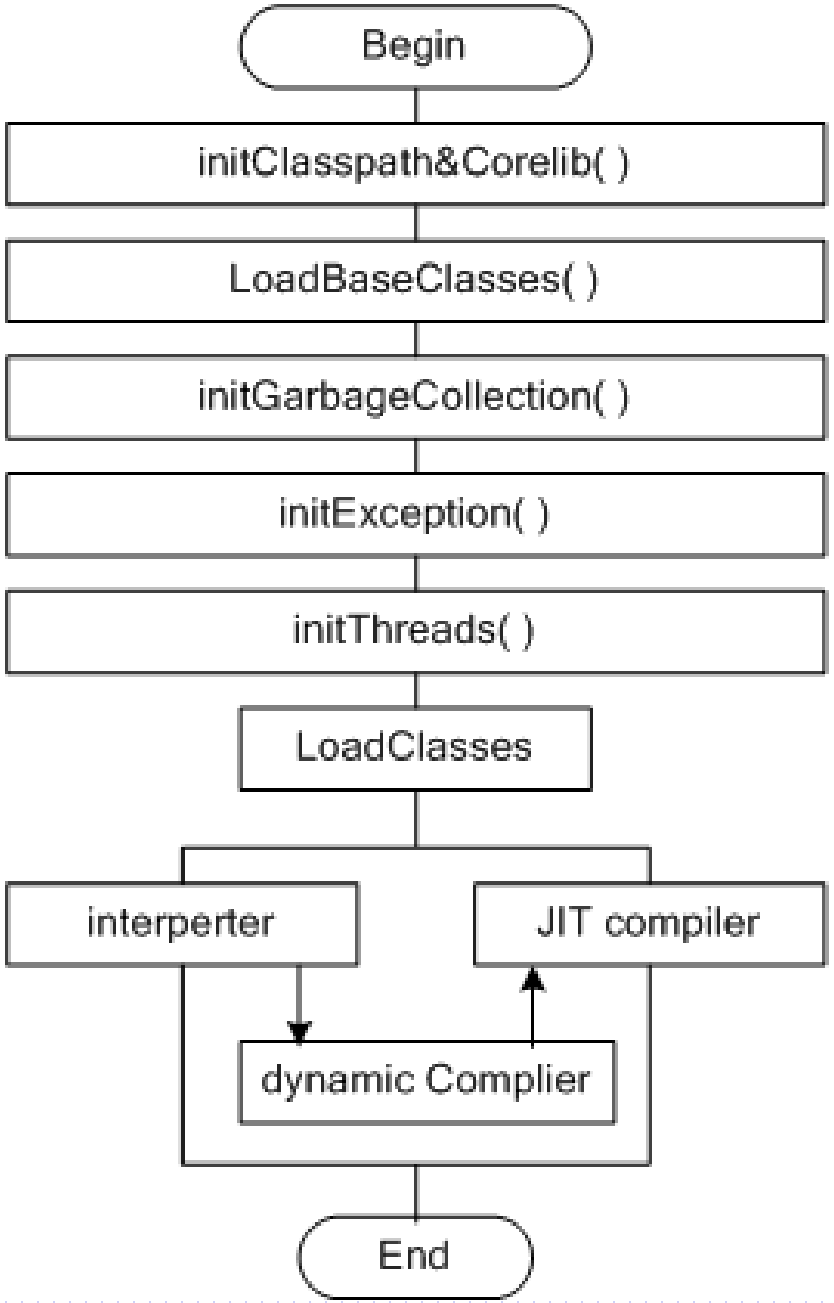
Java虚机要实现Java的面向对象程序:

- 类装载条件：创建类实例、调用类静态方法、使用类/接口静态字段、使用反射类方法、初始化子类前先初始化父类、启动main类
- 1）加载类：获取类的二进制流、解析类（方法区的数据结构）、创建java.lang.Class类的实例表示该类;
  - 自顶向下尝试加载类（加载类时）*，自底向上检查是否加载（使用类时）
- 2.1）验证类：格式检查、语义检查、字节码验证、符号引用验证;
- 2.2）准备：分配空间、设初值（0或空值）
- 2.3）解析类：将类、接口、字段和方法符号引用转为直接引用;
- 3）初始化：
- 线程运行;
- 处理异常;
- 作无用单元自动收集。

字节码执行

- 当Java源码被编译成Class文件后，虚拟机就会将Class文件内的方法字节码载入系统并加以执行。
- Java字节码对于虚拟机，如同汇编对应计算机。8bit长，约200+条指令
- 常用指令：
  - 出入栈指令
  - 类型转换指令
  - 运算指令
  - 比较控制指令
  - 对象操作指令
  - 函数调用与返回指令
  - 同步控制

- 虚机执行过程如下



装入本地核心类库java.lang,java.io
类目录和 路径并初始化

装入系统类和基类, ObjectClass,
SystemClass,ClassClass,intclass

无用单元收集器的初始化

异常处理初始化

线程类初始化

装入用户程序编译好的类文件

解释执行/及时编译执行

动态编译后执行

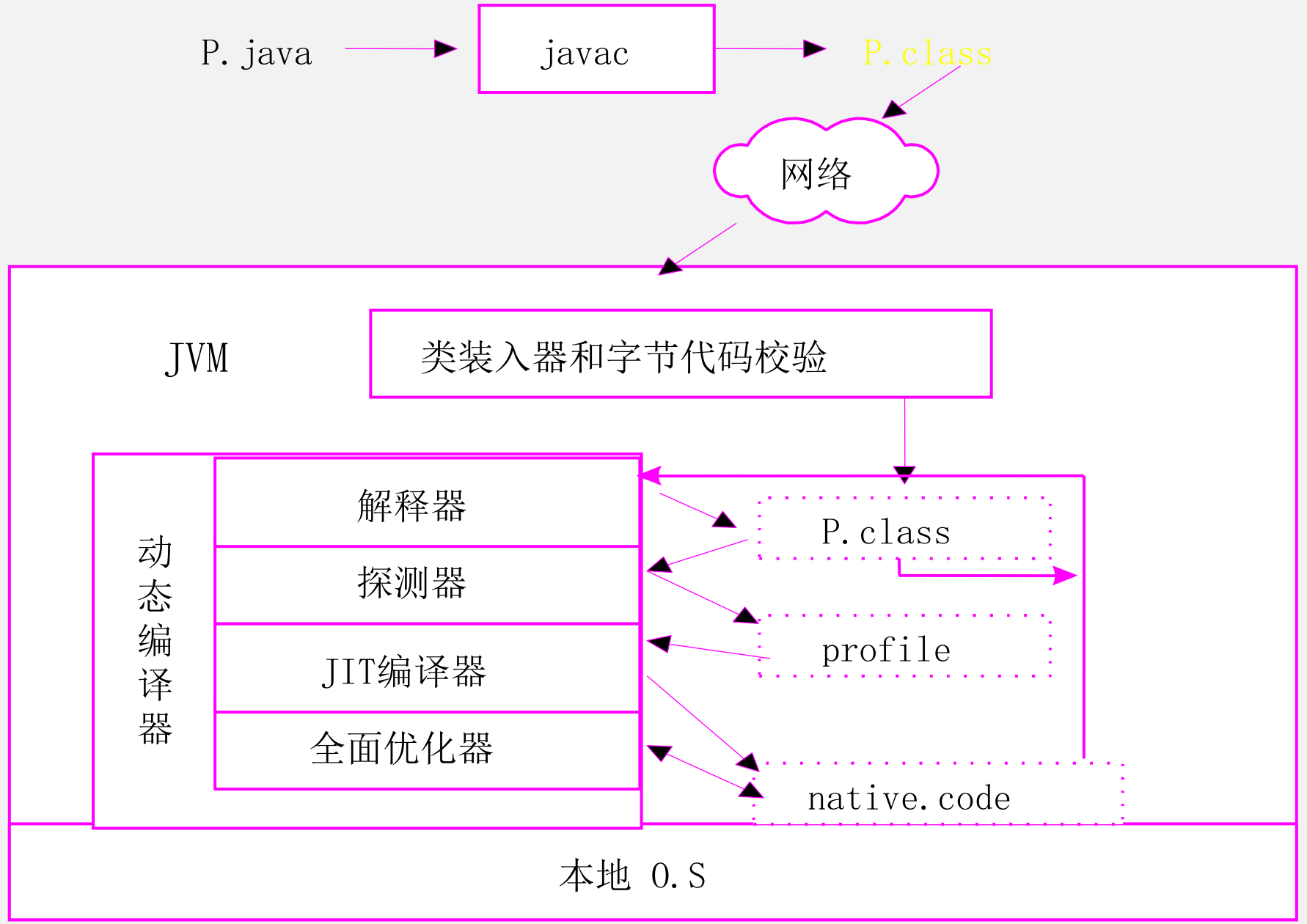
虚机技术进展

- 第一代JVM直接解释执行字节代码文件
- 第二代JVM及时编译方案，它的执行效率大写提高，约为解释执行的6-10倍。
- 第三代JVM采用近年发展出的动态编译技术。
  - 第三代虚拟机只编译运行时间长的热点，混用解释和编译后的机器码，效率最高。

即时编译(JIT: JUST-IN-TIME COMPILE)

- 使用即时编译是为了提高Java程序的执行效率，是对Java解释器的改进。
- 即时编译在虚拟机解释执行Java字节代码时发生。当虚拟机第一次调用某个方法时，不再直接解释这个方法的字节代码，而是用即时编译器将这个方法的字节代码编译成为本机目标代码，然后执行这些目标代码。
- 即时编译不是将整个类的字节代码进行一次性编译，而是只编译正在执行的函数，“边执行，边编译”。
  - 设置热点编译阈值
  - C1编译器（client模式）和C2编译器（server模式），5级编译策略
  - OSR栈上替换
  - 方法内联

- 第三代JVM采用动态编译技术。



走出去

- 平台无关
- 放置标示
- 脚本语言
- 服务计算语言
- 本体描述语言

放置标识

- 标记语言/置标语言（Markup Language ）不是程序设计语言，是电子文档发展以后，以计算机或网上自动处理电子文档而设计的语言。它们一般解释执行，平台无关。
- 标记语言是一种将文本（Text）以及文本相关的其他信息结合起来，使用标记（markup）进行标识，展现出关于文档结构和数据处理细节的计算机文字编码。
- 电子文档三类标准：信息的基本表达，信息结构描述，电子信息再现处理。

置标语言

- **表示性的置标语言（Presentational markup）**是在编码过程中，标记文档的结构信息。
- **过程性置标语言（Procedural markup）** 一般都专门于文字的表达，但通常对于文本编辑者可见,并且能够被软件依其出现顺序依次解读。//打印机PostScript
- **描述性置标（Descriptive markup），**所描述的是文件的内容或结构，而不是文件的显示外观或样式，制定SGML的基本思想就是把文档的内容与样式分开，XML、SGML都是典型的通用置标语言。

标准通用置标语言SGML

- SGML（Standard Generalized Markup Language）是ISO（国际标准化组织，International Standards Organization）在1986年推出的一个用来创建标识语言的语言标准。
- SGML是一种用来定义标识的语言，它提供了一种将数据内容与显示分离开来的数据表示方法，使得数据独立于机器平台和处理程序。
- SGML也被称作元语言（metalanguage），后来的许多标识语言都采用了SGML的标识方法，例如HTML、XML和XHTML。

SGML

- **SGML**把文档看作是对象集合，对象即元素。
- 每个元素是标记括着的内容，内容又可嵌入元素，这就形成树形分层体系结构。也是文档的逻辑结构。
- SGML 不涉及内容的语义。

(1) SGML文档结构

- SGML的文档结构由有类型的元素(实例对象)组成、元素是由命名的起始标记<tag_Name>和结束标记</tag_name>括着的正文单元。正文单元也叫内容。

例 SGML的文档实例

这是一个文档实例，就元素<anthology>而言，它给出了内容模型:

```
<anthology>//选集
<poem><title> The SICK ROSE </title>  //William Blake
  <stanza>//节
    <line> O Rose thon art sick. </line>
    <line> The invisible worm, </line>
    <line> That flies in the night </line>
    <line> In the howling storm: </line>
  </stanza>
  <stanza>
      <line> Has found out thy bed </line>
      <line> Of crimson joy: <line>
      <line> Does thy life destroy <line>
    </stanza>
</poem>
<!--more poems go here-->  //SGML的注释
</anthology>
```

SGML

(2) 定义SGML文档结构

在创建SGML文档实例时，首先要给出形式规格说明以指明该文档结构，即文档类型定义，简称DTD (Document Type Definition) 。

DTD是对正文的解释。为了分析可以从不同角度定义DTD。

例子中出现的各元素其形式声明如:

```
<!ELEMENT anthology - - (poem)>
<!ELEMENT poem - - (title? stanza+)>
<!ELEMENT title - O (#PCDATA)>
<!ELEMENT Line  O O (#PCDATA)>
```

SGML

(3) 并发结构

同样一个文档结构，可从另一观点定义它，设按页的诗集 <p.anth>:

```
<!DOCTYPE p.anth [
<!ELEMENT p.anth - - (page+) >
<!ELEMENT page   - - ((title?, line+)+)>
<!ELEMENT (title | line ) - O (#PCDATA) >
]>
```

因为它最低层行题名和行都是一样的(#PCDATA)，它们是并发定义的，可以写到一起:

```
<(anthology) anthology>      //A类型的A
<(p.anth) p.anth>            //B类型的B
<(p.anth) page>              //按B类型的DTD定义page元素
```

```
<!--   other titles and lines on this page here -->
        <(anthology)poem><title> The SICK ROSE
        <(anthology)stanza>
        <line> O Rose thou art sick.
        <line> The invisible worm.
    </(p.anth) page>                    //page结束
    <(p.anth) page>                     //另起一页
        <line> That flies in the night
        <line> In the howling storm:
        <(anthology) stanza>            //按A类型的DTD 定义stanza元素
        <line> Has found out thy hed
        <line> Of crimson joy:
        <line> And his dark crecret loe
        <line> Does tyh life destroy.
      </(anthology) poem>
    <!-- rest of material on this page here -->
    </(p.anth) page>            //B类型的page结束
    </(p.anth) p.anth>          //B类型的B结束
    </(anthology) anthology>  //A类型的A结束
```

这样定义的电子文档，当处理器只识别其中一个类型(有其定义)，则自动跳过另一个类的标记。若二者都识别则由人指定其一。

SGML

(4) 属性

- SGML的每个元素都可以定义属性或属性表。
- 属性并不影响元素的内容，只是为显示，处理提供方便。
- 在文档实例中要给出属性值，如:

  <poem id = PI status =“draft”>...</poem>

  <poem>元素定义了两个属性一为标识id一为状态status，它们均在起始标记之中，只看有无‘属性名=属性值’。

SGML

(5) 实体

- 为了跨文档间的可移植性，SGML提供了实体(Entity)概念。
- 实体是置标文档的某一部分，可大可小，小到一字符串，大到整个正文文件。如有声明：

  〈!ENTITY tei “Text Encoding Initiative”〉

  〈!ENTITY ChapTwo SYSTEM “sgmlmkup.txt”〉

  - 第一句声明了名为tei的实体，也叫内部实体，其值(即内容)是引号中的那段正文，故也称正文实体。
  - 第二个声明的实体是ChapTwo，指明它是SYSTEM实体，本文档以外的外部实体，其值是名为sgmlmkup.txt的正文文件的全部内容。

SGML

(6) 标记节

标记节(Marked Section)是为了将一个主文档做成多个版块，即在一段正文中做出条件正文域标记，其形式是:

<! [关键字[被标记正文段]]>

(7) 文档样式

SGML只处理文档结构、内容，没有为文档样式(style)制定标准。

SGML

SGML十分庞大，不同业务需求取其不同子集。关键的问题是‘定制’实现。至少，当前学习使用全集，既耗费资金又浪费精力。但作为学术研究是极有价值的。

但SGML复杂度太高，不适合网络的日常应用，加上开发成本高、不被主流浏览器所支持等原因，使得SGML在Web上的推广受到阻碍。HTML和XML都是SGML的子集。

HTML

- 超文本标记语言，即HTML（Hypertext Markup Language），是用于描述网页文档的一种标记语言。
- HTML是在SGML定义下的一个描述性语言，或可说 HTML是 SGML的一个应用程式， HTML不是程式语言，它只是标示语言。
- HTML被用来结构化信息——例如标题、段落和列表等等，也可用来在一定程度上描述文档的外观和语义。

HTML

HTML是SGML应用程序，它的文档模型也是树状结构模型。标记的约定和置标方式和SGML一样，只是文档类型定义DTD已由系统定义为HTML，用户不必学习SGML复杂的语法，直接用标记定义实例文档。

标记分以下四类:

- 文档结构定义
- 字体字型定义
- 版面布局定义
- 链接定义

它的最大特点是超文本和多媒体。

HTML

- HTML写的Web主页成了网上主要传递信息手段。

(I) 文档结构标记

<HTML>一个HTML文档<IHTML>

<TITLE> 题名</TITLE>

<!--注释-->

<HI>下一级标题</HI>　　//可以嵌套6级，至N6

....

<P> ....　　//另起一段，无结束标记

.....　　//另起一行，无结束标记

HTML

(2) 文档页面格式标记

分逻辑字符模式和物理字符模式。因为是逐词标记都是成对的。

逻辑的:

〈DFN〉定义的单词　　//一般斜体

〈EM〉　强调的单词　　//一般斜体

〈CITE〉书或电影主题名　//一般斜体

〈CODE〉程序代码段　　//打印字体

〈KBD〉键盘输入

〈SAMP〉计算机状态信息

〈STRONG〉特别强调词　　//黑体

<VAR>　代替变量的实例　//斜体

物理的:

〈B〉黑体　〈I〉斜体　<TI> 打印机字体

HTML

(3) 版面布局标记

各种列表　标记列表及表项:

<UL>　//无号列表　　<OL>　//有序号列表

<LI>　//表项　　<LI>　//表项1

<LI>　//表项　　<LI>　//表项2

</UL>　//至此结束　　</OL>　//至此结束

<DL>　//定义列表

<DT> //第一条目

<DD> //与<DT>交替出现，对<DT>的测试

<DT> //第二条目

<DD> //对<DT>的测试

</DL> //结束

XML

- 可扩展标记语言 (Extensible Markup Language, XML) ，用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。
- XML是标准通用标记语言 (SGML) 的子集，非常适合 Web 传输。XML提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。
- XML是一套**定义语义标记**的规则，这些标记将文档分成许多部件并对这些部件加以标识。
- 它也是元标记语言，即定义了用于定义其他与特定领域有关的、语义的、结构化的标记语言的句法语言。

XML

- XML文档的三种形式
  - 第一种文档形式先形式化地定义文档的类型、元素、属性、实体的标记，再以这些标记定义文档实例。
  - 第二种形式是不作DTD的形式定义，但遵照文档模型直接置标写出文档实例。文档的标记必须是良定义(well-formed)的。
  - 第三种形式也不作完整的DTD形式定义，但可以引用已有的DTD定义。称之为合法(valid)的XML文档。

- 指出XML文档是否独立是非常重要的。
- 为此，所有XML文档的第一行必须是处理指令。

例　一个出版商清单

```
<?xml version = "1.0" standalone = "yes"? >          //处理指令
<!DOCTYPE document [                                  //文档类型定义
 <!ELEMENT document ANY>
 <!ELEMENT title (#PCDATA)>
 <!ELEMENT publisher (name, email, homepage, address, voice, fax)+>
 <!ELEMENT name  (#PCDATA)>
 <!ELEMENT email (#PCDATA)>
 <!ELEMENT homepage (#PCDATA)>
 <!ELEMENT address (#PCDATA)>
 <!ELEMENT fax (#PCDATA)>
]>
 <document>                                           //以下文档实例
 <title>
          publishers of the Music of New York women composers
 </title>
 <publisher>
 <name> ACA-American composers Alliance </name>
    <email> info @ composers.com </email>
    <homepage> http: //www.composers.com/</homepage>
    <address> 170 west 74th st.NY NY10023 </address>
    <voice> 212-362-8900 </voice>
    <fax> 212-874-8605 </fax>
 </publisher>
 <publisher>
    <name> Alfred Publishing </name>
    <email></email>          //***
    <homepage / >            //**
    <address>15535 Morrison South Oaks CA 91403 </address>
    <voice></voice>
    <fax/>
 </publisher>
</document>
```

- 第三种形式的XML文档例子:

  例 一个合法的XML文档

```
<?xml version="1.0"? >
<! DOCTYPE advert SYSTEM "http://www.foo.org/ad.dtd">
<advert>
    <headline>...<plcl>...</headline>
    <text>...</text>
</advert>
```

XML

- XSL可扩展样式语言

  XML文档通过XSL可扩展样式语言描述的样式表(它本身也是XML的文档), 执行后产生一个HTML文档, 即XSL把XML文档翻译为HTML文档显现。

XML 特点

- XML与Access,Oracle和SQL Server等数据库不同，数据库提供了更强有力的数据存储和分析能力，例如：数据索引、排序、查找、相关一致性等，XML仅仅是展示数据。
- XML与HTML的设计区别是：XML是用来存储数据的，重在数据本身。而HTML是用来定义数据的外观，重在数据的显示模式。
- XML的简单使其易于在任何应用程序中读写数据，这使XML很快成为数据交换的唯一公共语言
- XML去掉了一些繁杂的功能，保留了SGML的结构化功能，使得网站设计者可以定义自己的文档类型，XML同时也推出一种新型文档类型，使得开发者也可以不必定义文档类型。

- XML有两个先驱--SGML和HTML，它既具有SGML的强大功能和可扩展性，同时又具有HTML的简单性。

  1. 良好的可扩展性。XML允许各个不同的行业根据自己独特的需要制定自己的一套标记。

  2. 内容与形式的分离。

  3. 遵循严格的语法要求。XML不但要求标记配对、嵌套，而且还要求严格遵守DTD的规定。

  4. 便于不同系统之间信息的传输。

  5. 具有较好的保值性。XML的保值性来自它的先驱之一--SGML语言。

- 单纯的XML是用来描述数据的，如果没有搭配适当的样式表，在Web浏览器中浏览XML文件时，只能看到XML文件的树形结构，这本身意义不大，所以需要借助一些相关技术。

1. CSS：层叠样式表(Cascading Style Sheets)，是一种用来表现HTML或XML等文件样式的语言。
2. DTD和XML Schema：XSD，用于可替代文档类型定义，一份XML schema文件描述了可扩展标记语言文档的结构。
3. XML DOM： XML Document Object Model ，定义了访问和处理 XML 文档的标准方法。用于获取、更改、添加或删除 XML 元素的标准。
4. XML XSLT：**可扩展样式表转换语言**，是一种样式转换标记语言，可以将XML数据档转换为另外的XML或其它格式，如HTML网页，纯文字。
5. XLink、XPionter和Xpath：创建超链接、定位数据、确定XML文档中某部分位置

HTML5的起源

- HTML5 是 W3C(World Wide Web Consortium) 与 WHATWG (Web Hypertext Application Technology Working Group)合作的结果。
- WHATWG 致力于 web 表单和应用程序，而 W3C 专注于 XHTML 2.0。
- 在 2006 年，双方决定进行合作，来创建一个新版本的 HTML。
- HTML 5 的第一份正式草案已于2008年1月22日公布。
- HTML 5有两大特点：
  - 强化了Web 网页的表现性能。
  - 追加了本地数据库等Web 应用的功能。

HTML5建立的规则

- 新特性应该基于 HTML、CSS、DOM 以及 JavaScript。
- 减少对外部插件的需求（比如 Flash）
- 更优秀的错误处理
- 更多取代脚本的标记
- HTML5 应该独立于设备
- 开发进程应对公众透明

HTML5新特性

- 用于绘画的 canvas 元素
- 用于媒介回放的 video 和 audio 元素
- 对本地离线存储的更好的支持
- 新的特殊内容元素，比如 article、footer、header、nav、section
- 新的表单控件，比如 calendar、date、time、email、url、search

HTML5改进特征

- HTML5提供了一些新的元素和属性，例如<nav>（网站导航块）和<footer>。这种标签将有利于搜索引擎的索引整理，同时更好的帮助小屏幕装置和视障人士使用。
- 除此之外，还为其他浏览要素提供了新的功能，如<audio>和<video>标记。

- audio 和 video 元素可以包含额外的标记，用来描述音频和视频的内容。这对搜索引擎也有帮助。

```
<audio src="kennedyinauguraladdrees.mp3">
  <p>
  Vice President Johnson, Mr. Speaker, Mr. Chief Justice,
  President Eisenhower, Vice President Nixon, President Truman,
  Reverend Clergy, fellow citizens:
  </p>

  <p>
  We observe today not a victory of party, but a celebration of
  freedom -- symbolizing an end, as well as a beginning --
 signifying renewal, as well as change. For I have sworn before
  you and Almighty God the same solemn oath our forebears
  prescribed nearly a century and three-quarters ago.
  </p>

  <p>
  The world is very different now. For man holds in his mortal
  hands the power to abolish all forms of human poverty and all
  forms of human life. And yet the same revolutionary beliefs for
  which our forebears fought are still at issue around the globe --
  the belief that the rights of man come not from the
  generosity of the state, but from the hand of God.
  </p>

  <p>
  ...
  </p>

  </audio>
```

处理图片

- 网站使用了HTML5的Canvas元素来创建一个图层列表，通过从图片库选择图片添加后，会新建一个新的图层，每一层都可以通过操纵将图片进行：移动、缩放、旋转、改变图层上下级、删除层、改变透明度、改变混合模式、启用或禁用的图片阴影。
- 网址：radikalfx.com/files/collage/demo.html

地理定位

- 这个web应用程序会自动检测您所在的位置，然后使用谷歌地图返回附近的就业信息。 不过需要注意的是，它要求分享您浏览器所处的地址信息。
- 网址： studio.html5rocks.com/#Geolocation

走出去

- 平台无关
- 放置标示
- 脚本语言
- 服务计算语言
- 本体描述语言

脚本语言

- 脚本语言（英语：Scripting language）是为了缩短传统的“编写、编译、链接、运行”（edit-compile-link-run）过程而创建的计算机编程语言。
  - 早期的脚本语言经常被称为批处理语言或工作控制语言；
  - 一个脚本通常是解释运行而非编译；
  - 脚本语言通常都有简单、易学、易用的特性，目的就是希望能让程序员快速完成程序的编写工作；
  - 而宏语言则可视为脚本语言的分支，两者也有实质上的相同之处。

脚本语言特点

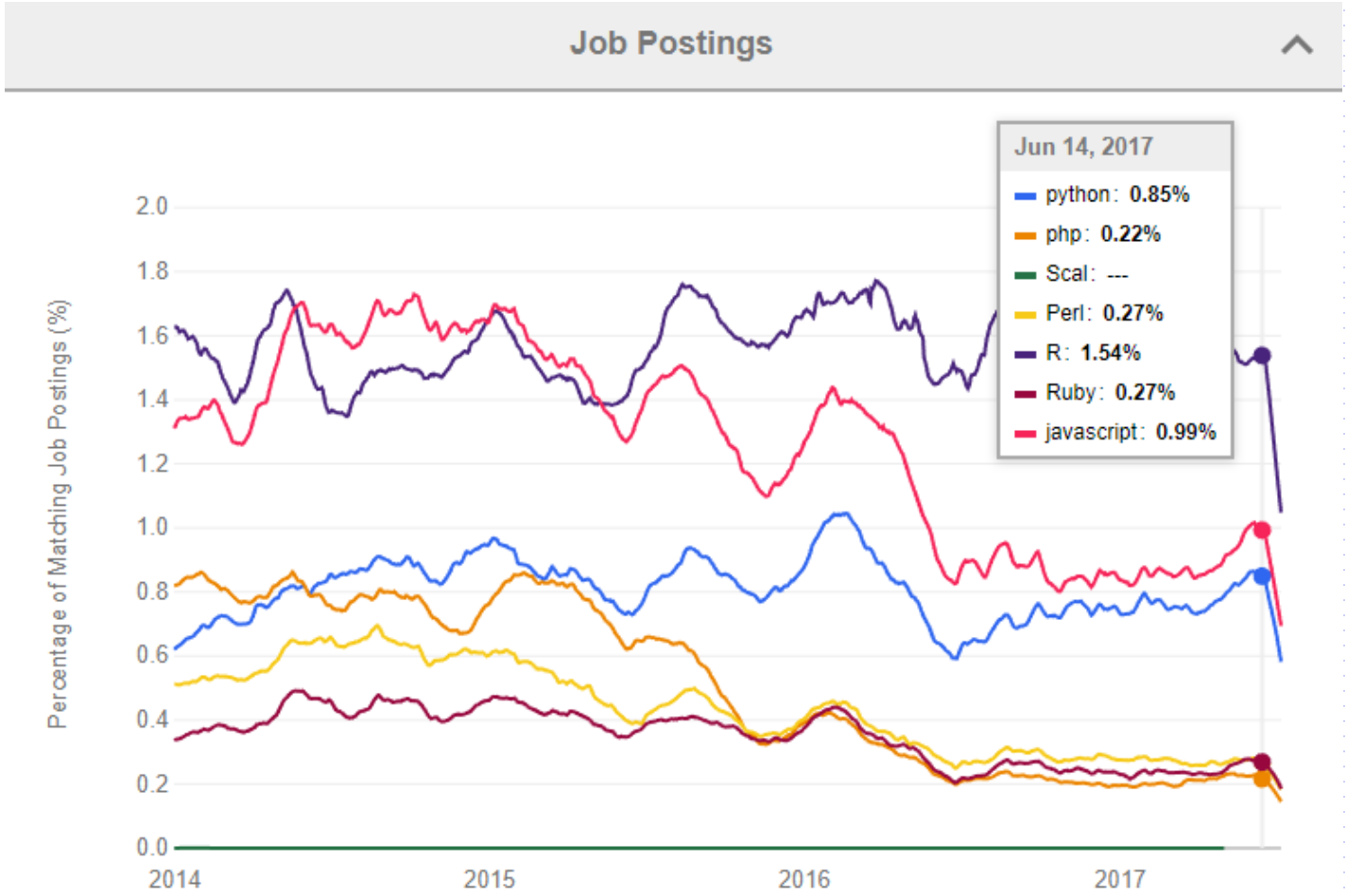
- 良好的快速开发
- 高效率的执行
- 解释而非编译执行
- 和其它语言编写的程序组件之间通信功能很强大

脚本语言的应用

- 脚本程序设计在Web程序设计中占有很重要的地位：
  - 无论是客户端动态页面设计，还是动态网站设计中的服务端编程，使用最多的就是脚本语言；
  - 脚本语言分为客户端脚本语言和服务端脚本语言，前者在客户端执行，后者在服务端执行。

常用的脚本语言

- C Shell
- JavaScript
- Nuva
- Perl
- Python
- R
- Ruby
- Tcl
- VBScript
- CSS



JAVASCRIPT语言

- JavaScript概述：
  - 是一种基于对象和事件驱动并且具有安全性能的脚本语言；它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，用来给HTML网页增加动态功能。
- JavaScript特点：
  - 脚本编程语言
    - JavaScript是一种脚本语言，它采用小程序段的方式实现编程；
  - 基于对象
  - 简单些
  - 安全性
  - 动态性
  - 可移植性

JAVASCRIPT能做什么?

- 使HTML页具有动态文本
- 对各种事件做出响应
- 能够读写HTML元素
- 能够用于验证数据
- 能够用于检测用户的浏览器
- 能够用于建立cookies
- … …

JAVASCRIPT与JAVA的区别

- 基于对象和面向对象
- 解释和编译
- 强变量和弱变量
  - JavaScript采用弱类型，即变量在使用前不需作声明，而是解释器在运行时检查其数据类型；
  - Java采用强类型变量检查，即所有变量在编译之前必须作声明；
- Java程序可单独运行，但JavaScript程序只能嵌入HTML中，不能单独运行

JAVASCRIPT的应用

- 网站开发
  - 网站前/后端开发
- 移动开发
  - Web APP
  - 混合式应用开发
- 桌面开发
- 插件开发

PERL语言

- Perl(Practical Extraction and Report Language)语言
  - 一种功能丰富的计算机程序语言，运行在超过100种计算机平台上，适用广泛，从大型机到便携设备，从快速原型创建到大规模可扩展开发；
- Perl语言特点：
  - 动态性
  - 解释型
  - 跨平台
  - 支持面向对象
  - 内部集成了正则表达式的功能

PERL语言的应用

- 被应用于：
  - 图形编程、系统管理、网络编程、金融、生物以及其他领域；
- 善于完成以下工作：
  - Web编程
  - 数据库处理
  - 文本、XML处理
  - 系统管理

走出去

- 平台无关（编程语言）
- 放置标示
- 脚本语言
- 服务计算语言
- 本体描述语言

服务计算

- 服务计算，泛指以服务及其组合为基础构造应用的新开发范型相关的方法、技术、规范、理论和支撑环境；
- 服务计算中的服务
  - 通常是在软件中实现的一项业务功能，可以被重用，人们不需要知道服务内部实现就可以理解服务用途并使用服务来构建应用系统
  - 是自描述、平台未知的计算单元，可支持快速、低成本的对分布式应用的组合
  - 可以执行多种功能，从简单的请求到复杂的业务流程

软件 ⇨? 服务

基于Web的软件 ⇨? Web服务

服务组合产生的背景

- 基于互联网的应用开发中已经出现了大量的Web服务，Web正发展成为一个集成信息资源的分布式的计算资源；
- 但单独的Web服务已经很难满足用户不断增长以及越来越复杂的应用需求，人们常常需要组合多个Web服务以完成一个较复杂的任务，这就是Web服务的组合问题

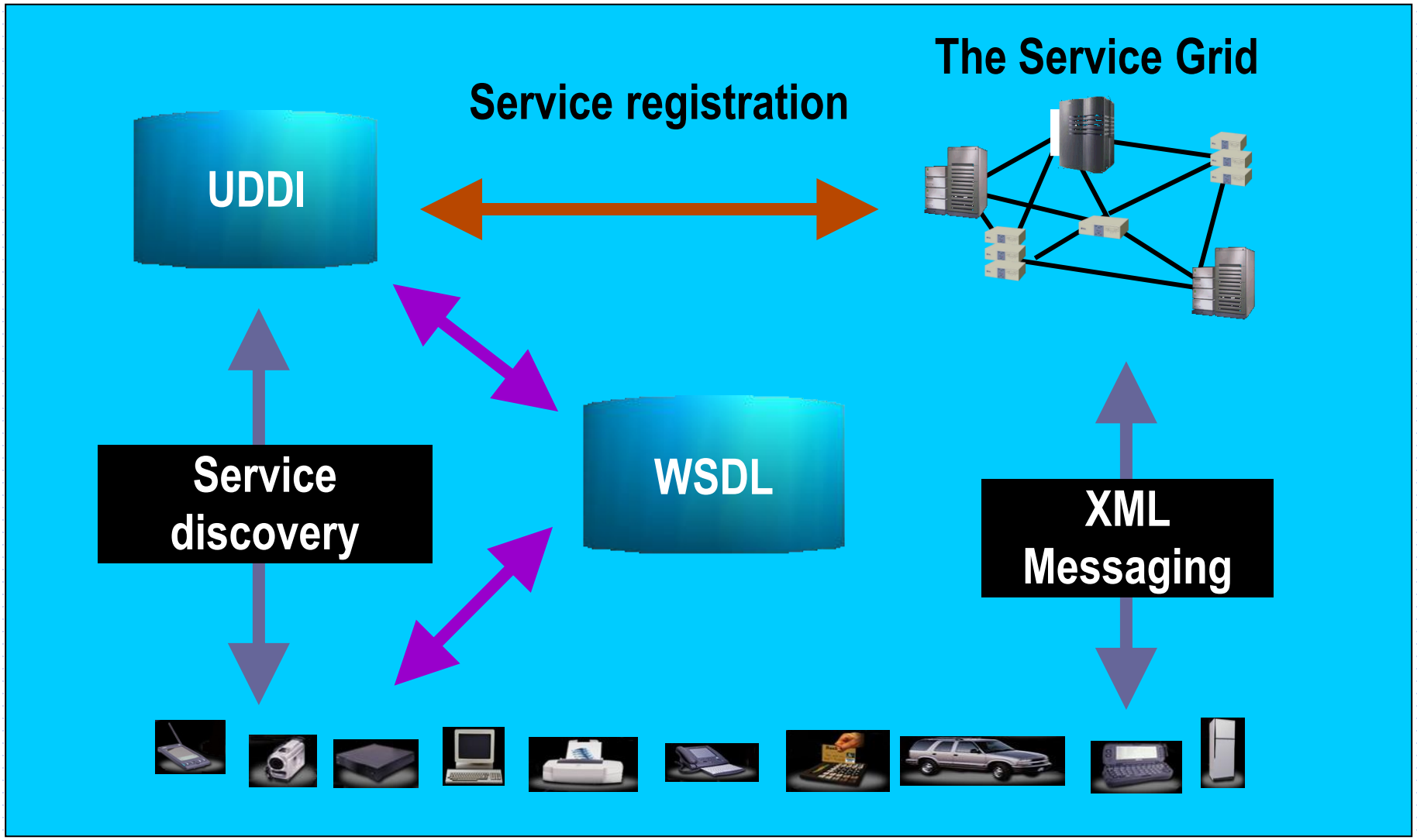
WEB服务实例

- 例子：假设开发人员需要搭建一个商务网站，这个网站需要一个验证客户合法身份的功能。为了实现这个功能，下面分别描述了可以采用的办法：

1. 由开发人员自己编写安全验证所需的全部代码。这样做显然不现实，一个安全验证程序涉及到诸多专业知识，并需要相当长的时间才能够完成。
2. 购买这段程序(通常是一个ActiveX组件)。在收到组件之后，首先将组件注册在自己的机器上，然后根据组件类型库产生接口文件。在实际编程中就可以使用这个接口文件来访问组件服务。很明显，这种方式在目前使用得最为广泛。
3. Web 服务：只需要在自己的程序中通过访问某个服务的URL地址，得到一份XML描述，并使用这个描述文件产生一个接口文件。然后，在实际编程中，只需要通过这个接口文件来访问服务就可以了。

一定要注意，这个服务可不是运行在我们机器上的，是运行在因特网上URL地址所指向的地方。

Web Services (general view)



WEB服务的描述、发布和组合

- 服务的描述和发布：WSDL，UDDI
  - 网络服务描述语言（Web Services Description Language ）是Web Service的描述语言，它包含一系列描述某个web service的定义。
  - UDDI是一种用于描述、发现、集成Web Service的技术，它是Web Service协议栈的一个重要部分。
- 服务组合：OWL-S
  - Web1.0是静态的WWW，其包括URI、HTML、HTTP等；
  - Web服务是Web2.0的一个代表；
  - OWL-S就是Web服务和语义Web的结合，解决Web服务描述和发现以及业务组合的语义表示。

WSDL

- 随着通讯协议和消息格式在WEB中的标准化，以某种格式化的方法描述通讯变得越来越重要，并且其实现的可能性也越来越大。
- WSDL通过定义一套XML的语法来描述网络服务的方式满足了这种需求。
- WSDL把网络服务定义成一个能交换消息的通讯端点集（communication collection）。

WSDL

- 一个WSDL文档在定义网络服务的时候使用如下的元素：
  - 类型——使用某种的类型系统（比如XSD）定义数据类型的容器
  - 消息——通讯数据抽象的有类型的定义
  - 操作——服务支持的动作的抽象描述
  - 端口类型——一个操作的抽象集合，该操作由一个或多个端点支持
  - 绑定—— 针对一个特定端口类型的具体的协议规范和数据格式规范
  - 端口—— 一个单一的端点，定义成一个绑定和一个网络地址的联接
  - 服务—— 相关的端点的集合

WSDL

- 为了描述消息的结构，需要具有丰富类型的系统，WSDL意识到了这种需求，因此它支持XML的schema规范作为它的规范的类型系统。
- 但是仅使用一种类型语言来描述现在和将来的所有消息格式显然是不可能的，因此WSDL可以扩展使用其他的类型定义语言。

- 例子 通过HTTP实现的SOAP 1.1 Request/Response

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
        xmlns:tns="http://example.com/stockquote.wsdl"
        xmlns:xsd1="http://example.com/stockquote.xsd"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

    <types>
        <schema targetNamespace="http://example.com/stockquote.xsd"
                xmlns="http://www.w3.org/1999/XMLSchema">
            <element name="TradePriceRequest">
                <complexType>
                    <all>
                        <element name="tickerSymbol" type="string"/>
                    </all>
                </complexType>
            </element>
            <element name="TradePriceResult">
                <complexType>
                    <all>
                        <element name="price" type="float"/>
                    </all>
                </complexType>
            </element>
        </schema>
    </types>
```

```
<message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePriceResult"/>
</message>

<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>

<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use="literal" namespace="http://example.com/stockquote.xsd"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="literal" namespace="http://example.com/stockquote.xsd"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
</binding>
```

WSDL

```
<service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort"
binding="tns:StockQuoteBinding">
        <soap:address
location="http://example.com/stockquote"/>
    </port>
</service>

</definitions>
```

UDDI

- 描述之后发布：UDDI（Universal Description Discovery and Integration）
- UDDI：统一描述、发现和集成协议，是一个广泛的，开放的行业计划。

服务组合

- 服务组合：Web 服务组合可以以两种方式进行：静态方式和动态方式。

  ①在静态组合中，被组合的服务在设计时由设计者选择。

  如果被组合的过程具有固定的特性，即如果在商业过程中，商业合伙人及其服务组件几乎不变动，那么静态组合就能满足这一需要。

  静态服务组合的两个最著名的平台是微软的Biztalk和Bea/Oracle 的WebLogic。

  ②在动态组合中，被组合的服务是在运行时选择的。

服务组合

- Web 服务组合是当前的一个热门研究领域，学术界和业界的研究群体提出了很多服务组合语言。
- IBM的Web 服务流语言（web service flow language，WSFL）和微软的XLANG 是两个最早的语言，用于定义Web 服务组合的标准。它们都扩展了W3C 的Web 服务描述语言（web service describe language，WSDL）。
- WSFL 是一个基于XML 的语言，它描述了复杂的服务组合，既支持服务的静态配置，又支持在Web服务注册中心动态查找服务。
- 微软的服务组合语言XLANG 对WSDL 进行行为标准的扩展，为服务组合提供了一个模型，但XLANG 只支持动态服务组合。

服务组合

- Web服务商业过程执行语言（BPEL4WS）是后来提出的一个标准，它综合了WSFL 和XLANG。BPEL4WS 试图把WSFL的有向图过程表述和XLANG 的基于结构化构建的过程进行合并，构成Web 服务的一个统一标准。
- 另一个服务组合语言是DARPA 代理标记语言（DAML-S）。DAML-S 为服务提供者提供了一个标记语言核心集，用于以无歧义，计算机可解释的方式来描述它们的服务属性和性能。Web 服务的DAML-S 标记方便了服务任务的自动化，包括Web服务的自动查找、执行、互操作、组合和执行监测。

服务组合

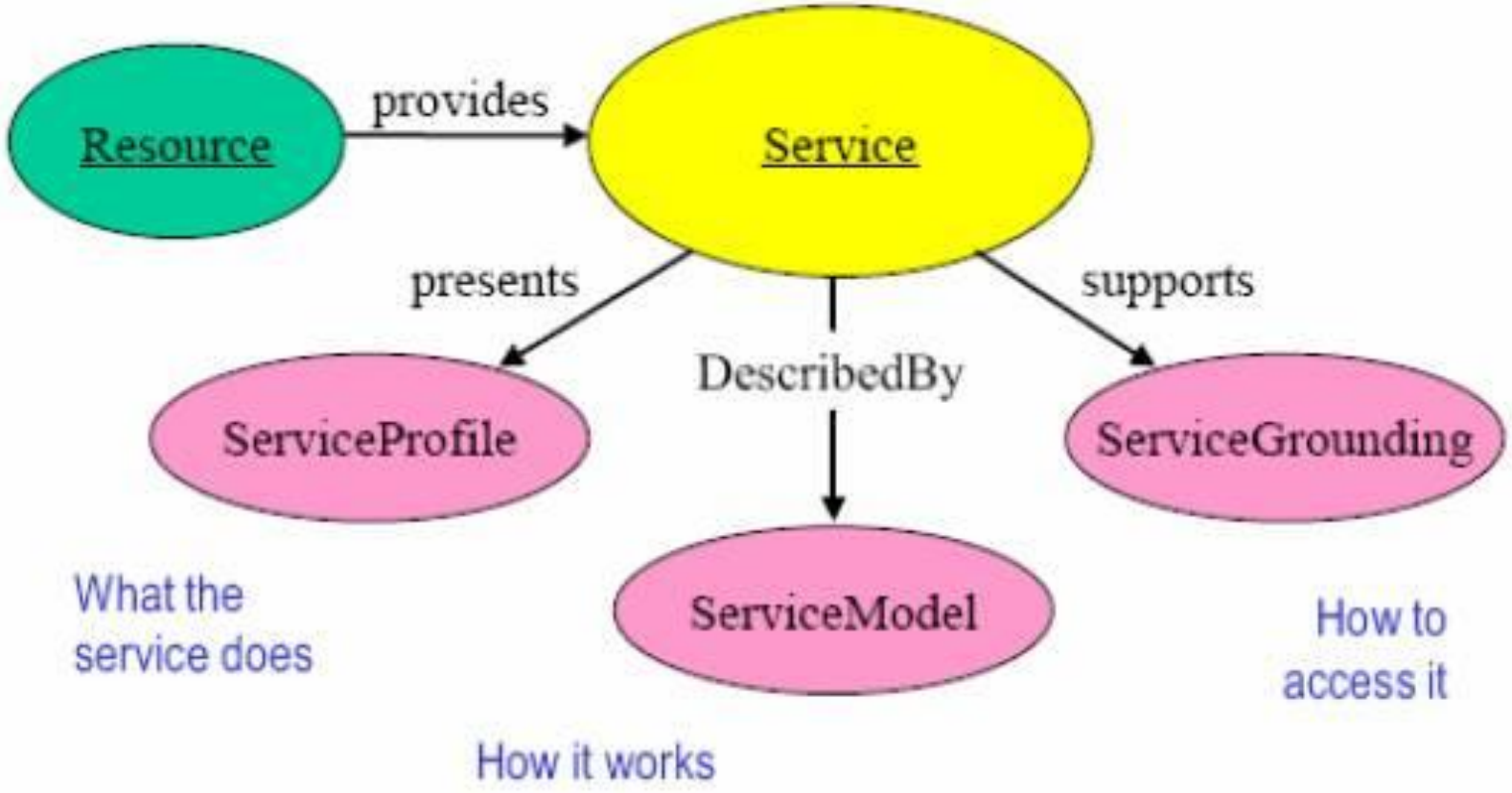
- 服务描述和发现后的一个问题：如何正确理解这些信息以保证集成进来的服务正如我所期望?
- 比如前面介绍的例子：假设开发人员需要搭建一个商务网站，这个网站需要一个验证客户合法身份的功能。
  - 计算机语义理解的问题。

服务组合

- 张三的个子比李四高，李四的个子比王五高->
  - 张三的个子是否比王五高?
  - 张三是否比王五高?
  - 张三是否比王五个子高?
  - 张三是否比王五个高?

服务组合

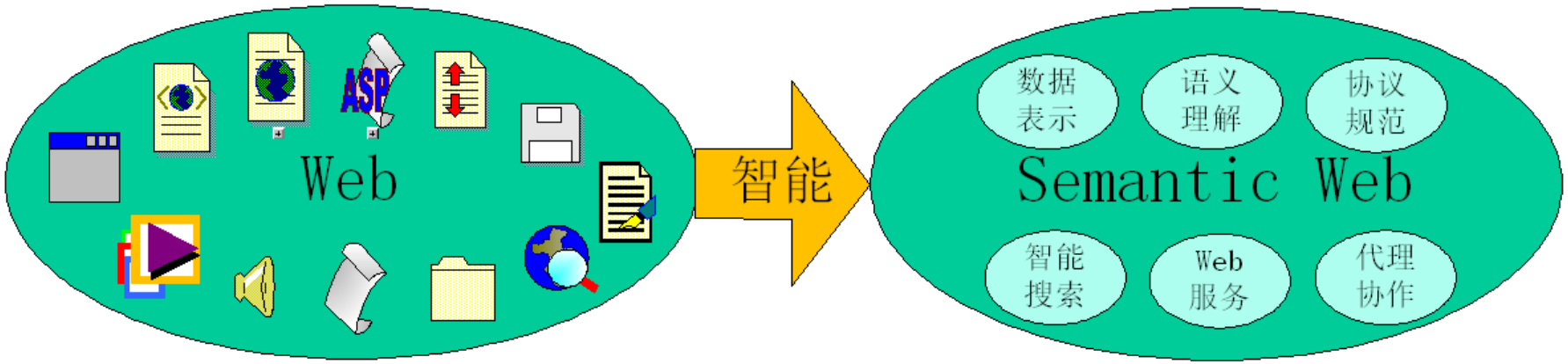
- 本体技术：针对语义理解提出的方法。
- OWL-S介绍：OWL Web Ontology Language for Services，OWL (Web Ontology Language)
- 使用OWL-S Editor进行讲解


Resource
provides
Service
presents
DescribedBy
supports
ServiceProfile
ServiceGrounding
ServiceModel
What the service does
How it works
How to access it

走出去

- 平台无关（编程语言）
- 放置标示
- 脚本语言
- 服务计算语言
- 本体描述语言

WEB的发展——语义网



- Web 语言
  - HTML
    - 数据和表示共存一体
    - 信息量巨大
    - 元数据单一且固定
    - 信息易于表示和发布
    - 不易于进行信息检索
  - XML
    - 数据和表示分离
    - 允许自定义元数据
    - 元数据具有人可以理解的语义
    - 机器不能理解元数据语义
    - 仍然不易于检索

RDF——语义WEB的基础

- RDF是Resource Description Framework的缩写，即资源描述框架；
  - 用于描述万维网上的资源及其类型，为网上资源描述提供了一种通用框架和实现集成的元数据解决方案。

WEB语言——RDF

- RDF基本概念：
  - 资源(Resource)
  - 属性(Property)
  - 陈述(Statement)
- RDF的特点：
  - 基于XML语法，使表示出来的XML/RDF文档具有语义理解的结构基础；
  - 促进了统一词汇表的使用；
  - 允许简单的逻辑推理；
- RDF的基本思想：
  - 用Web标识符（*URIs*）来标识事物，用简单的属性（property-特征, 性质, 关系）及属性值来描述资源。

Web语言——RDF

- RDF三元组有向图示意：

主体（subject）：声明被描述的事物
谓词（predicate）：这个事物的属性
客体（object）：这个属性的值

RDF文档实例

```
<?xml version="1.0"?>
<RDF>
  <Description about="http://www.w3school.com.cn/RDF">
    <author>David</author>
    <homepage>http://www.w3school.com.cn</homepage>
  </Description>
</RDF>
```

RDF文档：

```
<?xml version="1.0"?>
<Class rdf:ID="Resource"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
        xmlns="uri">
    <property>value</property>
    <property>value</property>
    ...
</Class>
```

RDF的扩充——RDFS(RDF SCHEMA)

- RDFS 可以看成是领域模型表达成RDF的形式化语言，就是说领域模型中的各类实体关系，都用RDF三元组来表达，写成RDF模式的序列化形式；
- RDFS引入更多的 “资源”来定义资源和资源之间的关系；

RDF的扩充——RDFS(RDF SCHEMA)

- rdf:type只能定义实例的类型，例如《红楼梦》是一本小说：
- *[1] ex:红楼梦 rdf:type ex:小说*
- 其中ex表示定义“红楼梦”和“小说”的命名域。
- 如果要定义“小说”（类名）是一种“文学作品”（类名）， 就没有相应的rdf资源元素，
- W3C扩展了一个rdfs:subClassOf，和rdfs:superClassOf，可以这样定义：
- *[2] ex:小说 rdfs:subClassOf ex:文学作品*
- *[3] ex:文学作品 rdfs:superClassOf ex:小说*

RDFS例子

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base= "http://www.animals.fake/animals#">

<rdf:Description rdf:ID="animal">
 <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="horse">
<rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>
</rdf:RDF>
```

RDFS例子

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base= "http://www.animals.fake/animals#">

<rdfs:Class rdf:ID="animal" />

<rdfs:Class rdf:ID="horse">
<rdfs:subClassOf rdf:resource="#animal"/>
</rdfs:Class>
</rdf:RDF>
```

RDFS

- 尽管RDFS的适用面和能力非常强大，但如果要表达更为丰富的语义和推理关系，还需要从规则表达（如OWL和SKOS）和词表（如SKOS、FOAF、 DC等等）两方面进行扩展。
- 任何元数据方案以及本体模式，都是组成语义网标准规范体系中的成员，都是对语义网的贡献。

- 总结：
  - 在描述性语言提供更加抽象的描述时，面对Internet环境下对“请进来，走出去”的要求，产生了一系列技术。
  - “走出去”目前比较成功和实用。
  - “请进来”还有待发展和完善。
    - Docker，Kubernetes(K8s)

THE END