


钱进培训是哈法地区资深工程师组成的培训机构，通过各位老师的现身说法，帮助各位学员迅速掌握实战知识，为求职打下坚实的基础。电子邮件：[jin.qian.canada@gmail.com](mailto:jinqian.canada@gmail.com) 钱老师报名、答疑微信号：[qianjincanada](https://www.wechat.com/p/qianjincanada)，或扫描以下二维码添加：



钱老师 
加拿大



扫一扫上面的二维码图案，加我微信

 钱进老师



 钱进老师

Java高级班（JavaEE方向）

讲义3

钱进培训立足Halifax地区，面向在校生提供软件开发技能培训和课程辅导。

本课是钱进培训组织的软件开发系列课程的主打课程，主要针对已经学习过CSCI1100（JAVA1）的同学，目标是通过大约10次课的学习，掌握JavaEE开发必备的技能，对现代软件企业的开发方式、常用类库、方法论等在校很难学到的知识点进行全面讲解，做到心中有数，提前具备求职的基本技术素质。

第二次课，课后作业：

1. 继续练习、领会如何在工程中引入第三方的jar文件，到如下网站下载jar文件，新建一个java工程，添加jar文件。

https://commons.apache.org/proper/commons-lang/download_lang.cgi

http://commons.apache.org/proper/commons-io/download_io.cgi

2. 自己在google中搜索StringUtils和FileUtils两个常用工具类的用法

3. 分析access_log（我发给你的文件），找出访问最多的地址

思路： 1. 用FileUtils把文件读取到一个string的变量中，尝试用两种方法做

JAVA软件开发培训

第一种方法： 用 string类的自有函数 indexOf substring等，可以借助 array

第二种方法： 用StringUtils的静态函数，感受一下 StringUtils为什么方便

4. 看一下 <http://thechronicleherald.ca/heraldflyers>

这个地址，想一下如果想把每个flyer的图标都下载到本地，如何实现？

List是一个接口，而ArrayList是一个类,ArrayList继承并实现了List。

所以List不能被构造，但可以为List创建一个引用，相对而言，ArrayList就可以被构造。

```
List list; //正确 list=null;
```

```
List list=new List(); // 是错误的用法
```

List list = new ArrayList();这句创建了一个ArrayList的对象后把上溯到了List。此时它是一个List对象了，有些ArrayList有但是List没有的属性和方法，它就不能再用了。

而ArrayList list=new ArrayList();创建一对象则保留了ArrayList的所有属性。

这是一个例子：

```
import java.util.*;
```

```
public class TestList{
```

```
public static void main(String[] args){
```

```
List list = new ArrayList();
```

```
ArrayList arrayList = new ArrayList();
```

```
list.trimToSize(); //错误，没有该方法。
```

```
arrayList.trimToSize(); //ArrayList里有该方法。
```

```
}
```

```
}
```

(1) 一个类（如：**ClassA**），如果没有声明任何构造函数，那么系统会自动生成一个无参构造函数，此时使用**ClassA myClass= new ClassA（）**，不会报错。但是，如果显式声明了一个有参构造函数，却没有声明一个无参构造函数的话，系统不会自动生成一个无参构造函数，此时，再使用**ClassA myClass = new ClassA（）**就会报错。如果要消除报错，则必须使用有参构造函数，

或者添加一个无参构造函数。所以，一个类的构造函数，一般只有三种状况：无参 或者 无参 + 有参 或者 有参

(2) 在继承关系中，子类的所有构造函数（包括无参构造函数（默认构造函数），有参构造函数等），如果不显式声明调用哪种**super**，那么都会默认调用**super（）**，即都会默认调用父类的无参构造函数（默认构造函数）。而，如果父类此时没有无参构造函数存在的话，就会报错。为了修改报错，只能显式调用父类显式声明的构造函数之一或者在父类中增加无参构造函数。

(3) 在继承关系中，当使用**new**一个类时，程序的执行顺序为：首先执行父类的构造函数方法，再执行自身的构造函数方法，这个执行顺序不可更改，否则报错或不能运行。

(4) 在深度继承关系中，即当存在**C**继承**B**，**B**继承**A**，或者更多层继承时，首先执行最上层的构造函数，再依次顺着继承链传递下去，一直到创建对象的那个类。例如：我们声明**C**对象时，调用的是**C**的构造函数**c**，构造函数**c**调用的是父类**B**的构造函数**b**，构造函数**b**调用的是父类**A**中的构造函数**a**，那么执行结果就是：**a => b => c**。

在语法层次，java 语言对于抽象类和接口分别给出了不同的定义。下面用**Demo** 类来说明他们之间的不同之处。

```
public abstract class Demo {  
    abstract void method1();  
    void method2(){  
        //实现  
    }  
}
```

```
interface Demo {  
    void method1();  
    void method2();  
}
```

abstract class和**interface**是Java语言中对于抽象类定义进行支持的两种机制，正是由于这两种机制的存在，才赋予了Java强大的面向对象能力。**abstract class**和**interface**之间在对于抽象类定义的支持方面具有很大的相似性，甚至可以相互替换

abstract class在Java语言中表示的是一种继承关系，一个类只能使用一次继承关系。但是，一个类却可以实现多个**interface**。这是Java语言的设计者在考虑Java对于多重继承的支持方面的一种折中考虑。

接口和抽象类的概念不一样。接口是对动作的抽象，抽象类是对根源的抽象。

抽象类表示的是，这个对象是什么。接口表示的是，这个对象能做什么。比如，男人，女人，这两个类（如果是类的话.....），他们的抽象类是人。说明，他们都是人。

人可以吃东西，狗也可以吃东西，你可以把“吃东西”定义成一个接口，然后让这些类去实现它。

所以，在高级语言上，一个类只能继承一个类（抽象类）(正如人不可能同时是生物和非生物)，但是可以实现多个接口(吃饭接口、走路接口)。

在面向对象的领域一切都是对象，同时所有的对象都是通过类来描述的，但是并不是所有的类都是来描述对象的。如果一个类没有足够的信息来描述一个具体的对象，而需要其他具体的类来支撑它，那么这样的类我们称它为抽象类。比如 `new Animal()`，我们都知道这个是产生一个动物 `Animal` 对象，但是这个 `Animal` 具体长成什么样子我们并不知道，它没有一个具体动物的概念，所以他就是一个抽象类，需要一个具体的动物，如狗、猫来对它进行特定的描述，我们才知道它长成啥样。

在面向对象领域由于抽象的概念在问题领域没有对应的具体概念，所以用以表征抽象概念的抽象类是不能实例化的。

同时，抽象类体现了数据抽象的思想，是实现多态的一种机制。它定义了一组抽象的方法，至于这组抽象方法的具体表现形式有派生类来实现。同时抽象类提供了继承的概念，它的出发点就是为了继承，否则它没有存在的任何意义。所以说定义的抽象类一定是用来继承的，同时在一个以抽象类为节点的继承关系等级链中，叶子节点一定是具体的实现类。

在使用抽象类时需要注意几点：

- 1、抽象类不能被实例化，实例化的工作应该交由它的子类来完成，它只需要有一个引用即可。
- 2、抽象方法必须由子类来进行重写。
- 3、只要包含一个抽象方法的抽象类，该方法必须要定义成抽象类，不管是否还包含有其他方法。
- 4、抽象类中可以包含具体的方法，当然也可以不包含抽象方法。
- 5、子类中的抽象方法不能与父类的抽象方法同名。

定义一个抽象动物类 `Animal`，提供抽象方法叫 `cry()`，猫、狗都是动物类的子类，由于 `cry()` 为抽象方法，所以 `Cat`、`Dog` 必须要实现 `cry()` 方法。如下：

```
public abstract class Animal {  
    public abstract void cry();  
}  
  
public class Cat extends Animal{  
    @Override  
    public void cry() {  
        System.out.println("猫叫： 喵喵...");  
    }  
}  
  
public class Dog extends Animal{  
    @Override  
    public void cry() {  
        System.out.println("狗叫:汪汪...");  
    }  
}
```

```
public class Test {  
  
    public static void main(String[] args) {  
        Animal a1 = new Cat();  
        Animal a2 = new Dog();  
  
        a1.cry();  
        a2.cry();  
    }  
}
```

Output:

猫叫：喵喵...

狗叫:汪汪...

二、接口

接口是一种比抽象类更加抽象的“类”。

接口是用来建立类与类之间的协议，它所提供的只是一种形式，而没有具体的实现。同时实现该接口的实现类必须要实现该接口的所有方法，通过使用 **implements** 关键字，他表示该类在遵循某个或某组特定的接口，同时也表示着"interface"只是它的外貌，但是现在需要声明它是如何工作的”。

接口是抽象类的延伸，**java** 为了保证数据安全是不能多重继承的，也就是说继承只能存在一个父类，但是接口不同，一个类可以同时实现多个接口，不管这些接口之间有没有关系，所以接口弥补了抽象类不能多重继承的缺陷，但是推荐继承和接口共同使用，因为这样既可以保证数据安全性又可以实现多重继承。

在使用接口过程中需要注意如下几个问题：

- 1、1个 Interface 的所有方法访问权限自动被声明为 `public`。确切的说只能为 `public`，当然你可以显示的声明为 `protected`、`private`，但是编译会出错！
- 2、接口中可以定义“成员变量”，或者说是不可变的常量，因为接口中的“成员变量”会自动变为 `public static final`。可以通过类命名直接访问：`ImplementClass.name`。
- 3、接口中不存在实现的方法。
- 4、实现接口的非抽象类必须要实现该接口的所有方法。
- 5、不能使用 `new` 操作符实例化一个接口，但可以声明一个接口变量，该变量必须引用 (refer to) 一个实现该接口的类的对象。可以使用 `instanceof` 检查一个对象是否实现了某个特定的接口。例如：`if(anObject instanceof Comparable){}`。

抽象类方式中，抽象类可以拥有任意范围的成员数据，同时也可以拥有自己的非抽象方法，但是接口方式中，它仅能够有静态、不能修改的成员数据（但是我们一般是不会在接口中使用成员数据），同时它所有的方法都必须是抽象的。在某种程度上来说，接口是抽象类的特殊化。

对子类而言，它只能继承一个抽象类，但是却可以实现多个接口。

1、抽象层次不同。抽象类是对类抽象，而接口是对行为的抽象。抽象类是对整个类整体进行抽象，包括属性、行为，但是接口却是对类局部（行为）进行抽象。

2、跨域不同。抽象类所跨域的是具有相似特点的类，而接口却可以跨域不同的类。我们知道抽象类是从子类中发现公共部分，然后泛化成抽象类，子类继承该父类即可，但是接口不同。实现它的子类可以不存在任何关系，共同之处。例如猫、狗可以抽象成一个动物类抽象类，具备叫的方法。鸟、飞机可以实现飞 Fly 接口，具备飞的行为，这里我们总不能将鸟、飞机共用一个父类吧！所以说抽象类所体现的是一种继承关系，要想使得继承关系合理，父类和派生类之间必须存在"is-a"关系，即父类和派生类在概念本质上应该是相同的。对于接口则不然，并不要求接口的实现者和接口定义在概念本质上是一致的，仅仅是实现了接口定义的契约而已。

3、设计层次不同。对于抽象类而言，它是自下而上来设计的，我们要先知道子类才能抽象出父类，而接口则不同，它根本就不需要知道子类的存在，只需要定义一个规则即可，至于什么子类、什么时候怎么实现它一概不知。比如我们只有一个猫类在这里，如果你这是就抽象成一个动物类，是不是设计有点儿过度？我们起码要有两个动物类，猫、狗在这里，我们在抽象他们的共同点形成动物抽象类吧！所以说抽象类往往都是通过重构而来的！但是接口就不同，比如说飞，我们根本就不知道会有什么东西来实现这个飞接口，怎么实现也不得而知，我们要做的就是事前定义好飞的行为接口。所以说抽象类是自底向上抽象而来的，接口是自顶向下设计出来的。

总结：接口和抽象类

“接口是个规范”，既然不是一个类去实现，那就是有很多地方有用到，大家需要统一标准。甚至有的编程语言（Object-C）已经不把接口叫 **interface**，直接叫 **protocol**。统一标准的目的是大家都知道这个是做什么的，但是具体不用知道具体怎么做。

我知道 **Comparable** 这个接口是用来比较两个对象的，那么如何去比较呢？数字有数字的比较方法，字符串有字符串的比较方法，学生（自己定义的类）也有自己的比较方法。然后，在另外一个负责对象排序（不一定是数字喔）的代码里面，肯定需要将两个对象比较。这两个对象是什么类型呢？**Object a,b**？肯定不行，**a > b** 这样的语法无法通过编译。**int a,b**？也不行？一开始就说了，不一定是数字。....所以，**Comparable** 就来了。他告诉编译器，**a b** 两个对象都满足 **Comparable** 接口，也就是他们是可以进行比较的。具体怎么比较，这段程序不需要知道。所以，他需要一些具体的实现，**Comparable** 接口有一个方法，叫 **compareTo**。那么这个方法就是用来取代 **<、>** 这样的运算符。

抽象类

抽象类是用来捕捉子类的通用特性的。它不能被实例化，只能被用作子类的超类。抽象类是被用来创建继承层级里子类的模板。

用`abstract`关键字修饰一个类时，这个类叫做抽象类，用`abstract`修饰一个方法时，这个方法叫做抽象方法。

含有抽象方法的类必须被声明为抽象类，抽象类必须被继承，抽象方法必须被重写。

抽象类不能被实例化。

抽象方法只需声明，不需要实现。

接口

接口是抽象方法的集合。如果一个类实现了某个接口，那么它就继承了这个接口的抽象方法。这就像契约模式，如果实现了这个接口，那么就必须确保使用这些方法。接口只是一种形式，接口自身不能做任何事情。

java比较器的两种实现

比较器有两种实现方式：

- 1.让相应的类实现Comparable接口，重写接口中的compareTo(T o)方法。
- 2.由于第一种方法需要修改类的代码，那么第二种方法就另辟蹊径：再定义一个需要作比较类的比较器，让其实现比较器接口Comparator，重写接口中的比较器接口Compare(T o1, T o2)方法在需要使用时，将该需要作比较类与该比较器放在一起即可。

Comparable 是排序接口。

若一个类实现了Comparable接口，就意味着“该类支持排序”。即然实现Comparable接口的类支持排序，假设现在存在“实现Comparable接口的类的对象的List列表(或数组)”，则该List列表(或数组)可以通过 Collections.sort (或 Arrays.sort) 进行排序。

Comparable 定义

Comparable 接口仅仅只包括一个函数，它的定义如下：

```
package java.lang;
import java.util.*;

public interface Comparable<T> {
    public int compareTo(T o);
}
```

说明：

假设我们通过 x.compareTo(y) 来“比较x和y的大小”。若返回“负数”，意味着“x比y小”；返回“零”，意味着“x等于y”；返回“正数”，意味着“x大于y”。

Comparator 是比较器接口。

我们若需要控制某个类的次序，而该类本身不支持排序(即没有实现Comparable接口)；那么，我们可以建立一个“该类的比较器”来进行排序。这个“比较器”只需要实现Comparator接口即可。

也就是说，我们可以通过“实现Comparator类来新建一个比较器”，然后通过该比较器对类进行排序。

Comparator 定义

Comparator 接口仅仅只包括两个函数，它的定义如下：

```
package java.util;

public interface Comparator<T> {
    int compare(T o1, T o2);
    boolean equals(Object obj);
}
```

说明：

(01) 若一个类要实现Comparator接口：它一定要实现compareTo(T o1, T o2) 函数，但可以不实现 equals(Object obj) 函数。

为什么可以不实现 equals(Object obj) 函数呢？因为任何类，默认都是已经实现了 equals(Object obj)的。Java中的一切类都是继承于java.lang.Object，在Object.java中实现了 equals(Object obj)函数；所以，其它所有的类也相当于都实现了该函数。

(02) int compare(T o1, T o2) 是“比较o1和o2的大小”。返回“负数”，意味着“o1比o2小”；返回“零”，意味着“o1等于o2”；返回“正数”，意味着“o1大于o2”。

Comparator 和 Comparable 比较

Comparable是排序接口；若一个类实现了**Comparable**接口，就意味着“该类支持排序”。

而**Comparator**是比较器；我们若需要控制某个类的次序，可以建立一个“该类的比较器”来进行排序。

我们不难发现：**Comparable**相当于“内部比较器”，而**Comparator**相当于“外部比较器”。

```
public class ComparatorTest1 {  
    public ComparatorTest1() {}  
    public static void main(String[] args) {  
        Person[] group = {  
            new Person(10,"小明"),  
            new Person(10,"小智"),  
            new Person(12,"tom"),  
            new Person(11,"美美")  
        };  
        Arrays.sort(group);//默认升序  
        //按顺序打印Person信息  
        for(Person p : group){  
            System.out.println(p.getName() +"今年"+ p.getAge()+"岁~");  
        }  
    }  
}
```

```
class Person implements Comparable<Person>{
```

```
    @Override
```

```
    //比较器中比较大小的依据
```

```
    public int compareTo(Person o) {
```

```
        if(this.age > o.age)
```

```
            return 1;
```

```
        else if(this.age < o.age)
```

```
            return -1;
```

```
        else{
```

```
            return this.name.compareTo(o.name);
```

```
        }
```

```
    }
```

```
}
```



```
public class ComparatorTest2 {  
    public ComparatorTest2() {}  
    public static void main(String[] args) {  
        PersonN[] group = {  
            new PersonN(10,"小明"),  
            new PersonN(10,"小智"),  
            new PersonN(12,"tom"),  
            new PersonN(11,"美美")  
        };  
        PersonNComparator pc = new PersonNComparator();  
        Arrays.sort(group,pc);//默认升序  
        //按顺序打印Person信息  
        for(PersonN p : group){  
            System.out.println(p.getName() +"今年"+ p.getAge()+"岁~");  
        }  
    }  
}
```

//PersonN类的比较器类

```
class PersonNComparator implements Comparator<PersonN>{
```

```
    @Override
```

```
    public int compare(PersonN o1, PersonN o2) {
```

```
        if(o1.getAge() > o2.getAge())
```

```
            return 1;
```

```
        else if(o1.getAge() < o2.getAge())
```

```
            return -1;
```

```
        else{
```

```
            return o1.getName().compareTo(o2.getName());
```

```
        }
```

```
    }
```

```
}
```

1、`java.util.Collection` 是一个集合接口。它提供了对集合对象进行基本操作的通用接口方法。`Collection`接口在Java 类库中有很多具体的实现。`Collection`接口的意义是为各种具体的集合提供了最大化的统一操作方式。

`Collection`

└ `List`

| └ `LinkedList`

| └ `ArrayList`

| `Vector`

| `Stack`

`Set`

2、`java.util.Collections` 是一个包装类。它包含有各种有关集合操作的静态多态方法。此类不能实例化，就像一个工具类，服务于Java的Collection框架。

`Collections.sort`：将集合按照自然顺序或者给定的顺序排序。

`java.util.Arrays` 包含了许多处理数据的实用方法：

`Arrays.asList`：可以从 `Array` 转换成 `List`。可以作为其他集合类型构造器的参数。

`Arrays.sort`：对整个数组或者数组的一部分进行排序。也可以使用此方法用给定的比较器对对象数组进行排序。

想要明白hashCode的作用，你必须要先知道Java中的集合。总的来说，Java中的集合（Collection）有两类，一类是List，再有一类是Set。前者集合内的元素是有序的，元素可以重复；后者元素无序，但元素不可重复。

那么这里就有一个比较严重的问题了：要想保证元素不重复，可两个元素是否重复应该依据什么来判断呢？

这就是Object.equals方法了。但是，如果每增加一个元素就检查一次，那么当元素很多时，后添加到集合中的元素比较的次数就非常多了。也就是说，如果集合中现在已经有1000个元素，那么第1001个元素加入集合时，它就要调用1000次equals方法。这显然会大大降低效率。

于是，Java采用了哈希表的原理。哈希（Hash）实际上是个人名，由于他提出一哈希算法的概念，所以就以他的名字命名了。哈希算法也称为散列算法，是将数据依特定算法直接指定到一个地址上。初学者可以这样理解，hashCode方法实际上返回的就是对象存储的物理地址（实际可能并不是）。这样一来，当集合要添加新的元素时，先调用这个元素的hashCode方法，就一下子能定位到它应该放置的物理位置上。

如果这个位置上没有元素，它就可以直接存储在这个位置上，不用再进行任何比较了；如果这个位置上已经有元素了，就调用它的equals方法与新元素进行比较，相同的话就不存了，不相同就散列其它的地址。所以这里存在一个冲突解决的问题。这样一来实际调用equals方法的次数就大大降低了，几乎只需要一两次。

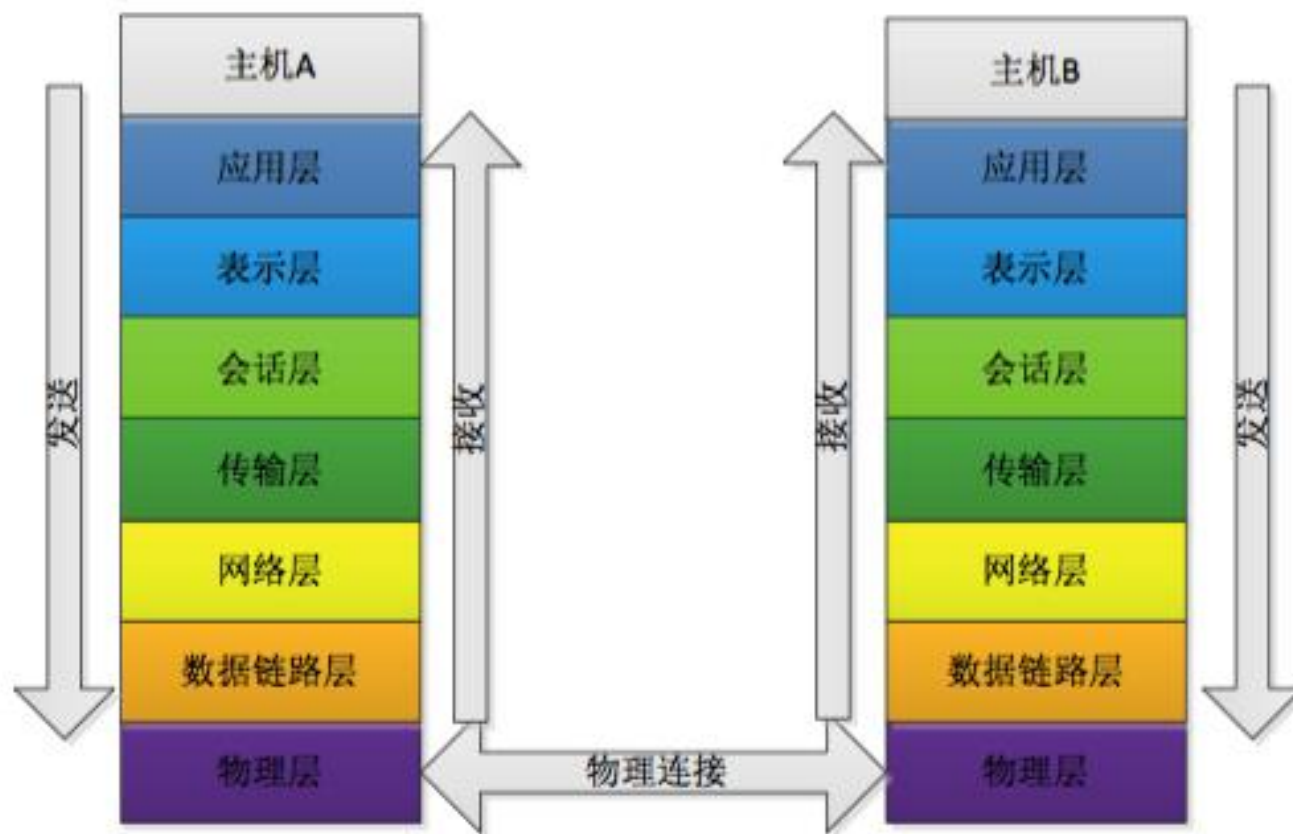
所以，Java对于equals方法和hashCode方法是这样规定的：

1、如果两个对象相同，那么它们的hashCode值一定要相同；2、如果两个对象的hashCode相同，它们并不一定相同 上面说的对象相同指的是用equals方法比较。

`hash code`(散列码，也可以叫哈希码值)是对象产生的一个整型值。其生成没有规律的。二者散列码可以获取对象中的信息，转成那个对象的“相对唯一”的整型值。所有对象都有一个散列码，`hashCode()`是根类 `Object` 的一个方法。

计算机网络技术 基础

为了使不同计算机厂家生产的计算机能够相互通信，以便在更大的范围内建立计算机网络，国际标准化组织（ISO）在1978年提出了“开放系统互联参考模型”，即著名的OSI/RM模型（Open System Interconnection/Reference Model）。它将计算机网络体系结构的通信协议划分为七层，自下而上依次为：物理层（Physics Layer）、数据链路层（Data Link Layer）、网络层（Network Layer）、传输层（Transport Layer）、会话层（Session Layer）、表示层（Presentation Layer）、应用层（Application Layer）。其中第四层完成数据传送服务，上面三层面向用户。



物理层：

物理层负责最后将信息编码成电流脉冲或其它信号用于网上传输；

eg：RJ45等将数据转化成0和1；

规定通信设备的机械的、电气的、功能的和过程的特性，用以建立、维护和拆除物理链路连接。具体地讲，机械特性规定了网络连接时所需接插件的规格尺寸、引脚数量和排列情况等；电气特性规定了在物理连接上传输bit流时线路上信号电平的大小、阻抗匹配、传输速率 距离限制等；功能特性是指对各个信号先分配确切的信号含义，即定义了DTE和DCE之间各个线路的功能；规程特性定义了利用信号线进行bit流传输的一组 操作规程，是指在物理连接的建立、维护、交换信息是，DTE和DCE双放在各电路上的动作系列。在这一层，数据的单位称为比特（bit）。属于物理层定义的典型规范代表包括：EIA/TIA RS-232、EIA/TIA RS-449、V.35、RJ-45等。

数据链路层：

数据链路层通过物理网络链路 供数据传输。不同的数据链路层定义了不同的网络和协议特征,其中包括物理编址、网络拓扑结构、错误校验、数据帧序列以及流控；

可以简单的理解为：规定了0和1的分包形式，确定了网络数据包的形式；

在物理层提供比特流服务的基础上，建立相邻结点之间的数据链路，通过差错控制提供数据帧（Frame）在信道上无差错的传输，并进行各电路上的动作系列。数据链路层在不可靠的物理介质上提供可靠的传输。该层的作用包括：物理地址寻址、数据的成帧、流量控制、数据的检错、重发等。在这一层，数据的单位称为帧（frame）。数据链路层协议的代表包括：SDLC、HDLC、PPP、STP、帧中继等。

网络层

网络层负责在源和终点之间建立连接;

可以理解为，此处需要确定计算机的位置，怎么确定？IPv4，IPv6！

在计算机网络中进行通信的两个计算机之间可能会经过很多个数据链路，也可能还要经过很多通信子网。网络层的任务就是选择合适的网间路由和交换结点，确保数据及时传送。网络层将数据链路层提供的帧组成数据包，包中封装有网络层包头，其中含有逻辑地址信息--源站点和目的站点地址的网络地址。如果你在谈论一个IP地址，那么你是在处理第3层的问题，这是“数据包”问题，而不是第2层的“帧”。IP是第3层问题的一部分，此外还有一些路由协议和地址解析协议（ARP）。有关路由的一切事情都在这第3层处理。地址解析和路由是3层的重要目的。网络层还可以实现拥塞控制、网际互连等功能。在这一层，数据的单位称为数据包（packet）。网络层协议的代表包括：IP、IPX、RIP、OSPF等。

传输层

传输层向高层 提供可靠的端到端的网络数据流服务。

可以理解为：每一个应用程序都会在网卡注册一个端口号，该层就是端口与端口的通信！常用的（TCP / IP）协议；

第4层的数据单元也称作数据包（**packets**）。但是，当你谈论TCP等具体的协议时又有特殊的叫法，TCP的数据单元称为段（**segments**）而UDP协议的数据单元称为“数据报（**datagrams**）”。这个层负责获取全部信息，因此，它必须跟踪数据单元碎片、乱序到达的数据包和其它在传输过程中可能发生的危险。第4层为上层提供端到端（最终用户到最终用户）的透明的、可靠的数据传输服务。所谓透明的传输是指在通信过程中 传输层对上层屏蔽了通信传输系统的具体细节。传输层协议的代表包括：TCP、UDP、SPX等。

会话层

会话层建立、管理和终止表示层与实体之间的通信会话；

建立一个连接（自动的手机信息、自动的网络寻址）；

在会话层及以上的高层次中，数据传送的单位不再另外命名，而是统称为报文。会话层不参与具体的传输，它提供包括访问验证和会话管理在内的建立和维护应用之间通信的机制。如服务器验证用户登录便是由会话层完成的。

表示层:

表示层 供多种功能用于应用层数据编码和转化,以确保以一个系统应用层发送的信息 可以被另一个系统应用层识别;

可以理解为: 解决不同系统之间的通信, eg: Linux下的QQ和Windows下的QQ可以通信;

这一层主要解决拥护信息的语法表示问题。它将欲交换的数据从适合于某一用户的抽象语法, 转换为适合于OSI系统内部使用的传送语法。即提供格式化的表示和转换数据服务。数据的压缩和解压缩, 加密和解密等工作都由表示层负责。

应用层:

OSI 的应用层协议包括文件的传输、访问及管理协议(FTAM),以及文件虚拟终端协议(VIP)和公用管理系统信息(CMIP)等;

规定数据的传输协议;

应用层为操作系统或网络应用程序提供访问网络服务的接口。应用层协议的代表包括: Telnet、FTP、HTTP、SNMP等。

常见的应用层协议：

协议	端口	说明
HTTP	80	超文本传输协议
HTTPS	443	HTTP+SSL,HTTP的安全版
FTP	20,21,990	文件传输协议
POP3	110	邮局协议
SMTP	25	简单邮件传输协议
telnet	23	远程终端协议

开放式系统互联（OSI）模型与TCP/IP协议有什么区别？

开放式系统互联模型是一个参考标准，解释协议相互之间应该如何相互作用。TCP/IP协议是美国国防部发明的，是让互联网成为了目前这个样子的标准之一。开放式系统互联模型中没有清楚地描绘TCP/IP协议，但是在解释TCP/IP协议时很容易想到开放式系统互联模型。两者的主要区别如下：

TCP/IP协议中的应用层处理开放式系统互联模型中的第五层、第六层和第七层的功能。

TCP/IP协议中的传输层并不能总是保证在传输层可靠地传输数据包，而开放式系统互联模型可以做到。TCP/IP协议还提供一项名为UDP（用户数据报协议）的选择。UDP不能保证可靠的数据包传输。

TCP/UDP协议

TCP(Transmission Control Protocol)和**UDP(User Datagram Protocol)**协议属于传输层协议。其中**TCP**提供IP环境下的数据可靠传输，它提供的服务包括数据流传送、可靠性、有效流控、全双工操作和多路复用。通过面向连接、端到端和可靠的数据包发送。通俗说，它是事先为所发送的数据开辟出连接好的通道，然后再进行数据发送；而**UDP**则不为IP提供可靠性、流控或差错恢复功能。一般来说，**TCP**对应的是可靠性要求高的应用，而**UDP**对应的则是可靠性要求低、传输经济的应用。

TCP支持的应用协议主要有：**Telnet**、**FTP**、**SMTP**等；**UDP**支持的应用层协议主要有：**NFS**（网络文件系统）、**SNMP**（简单网络管理协议）、**DNS**（主域名称系统）、**TFTP**（通用文件传输协议）等。

TCP/IP协议与低层的数据链路层和物理层无关，这也是**TCP/IP**的重要特点。

IP协议(Internet Protocol)又称互联网协议，是支持网间互连的数据报协议，它与TCP协议（传输控制协议）一起构成了TCP/IP协议族的核心。它提供网间连接的完善功能，包括IP数据报规定互连网络范围内的IP地址格式。

Internet 上，为了实现连接到互联网上的结点之间的通信，必须为每个结点（入网的计算机）分配一个地址，并且应当保证这个地址是全网唯一的，这便是IP地址。

目前的IP地址（IPv4：IP第4版本）由32个二进制位表示，每8位二进制数为一个整数，中间由小数点间隔，如159.226.41.98，整个IP地址空间有4组8位二进制数，由表示主机所在的网络的地址（类似部队的编号）以及主机在该网络中的标识（如同士兵在该部队的编号）共同组成。

HTTP协议

HTTP 协议是互联网的基础协议，也是网页开发的必备知识。HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

http协议规定了客户端和服务端之间的数据传输格式.

http优点:

简单快速:

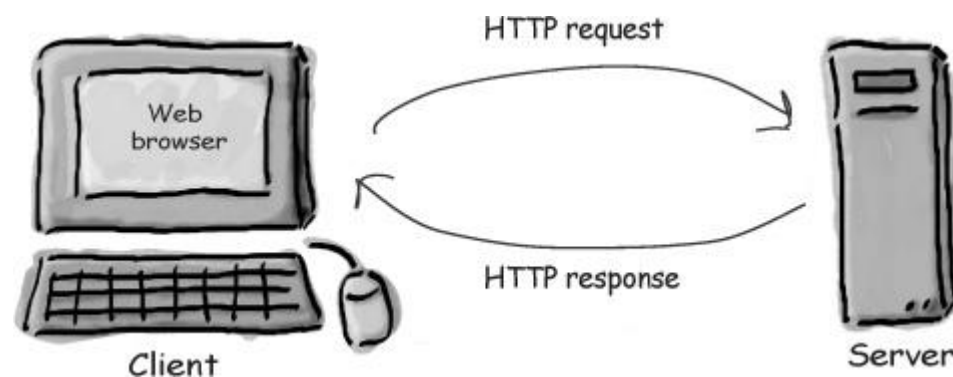
http协议简单,通信速度很快;

灵活:

http协议允许传输任意类型的数据;

短连接:

http协议限制每次连接只处理一个请求,服务器对客户端的请求作出响应后,马上断开连接.这种方式可以节省传输时间.



主要特点

- 1、简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。
- 2、灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。
- 3.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
- 4.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

HTTP之URL

HTTP使用统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。URL是一种特殊类型的URI，包含了用于查找某个资源的足够的信息，URL,全称是UniformResourceLocator, 中文叫统一资源定位符,是互联网上用来标识某一处资源的地址。以下面这个URL为例，介绍下普通URL的各部分组成：

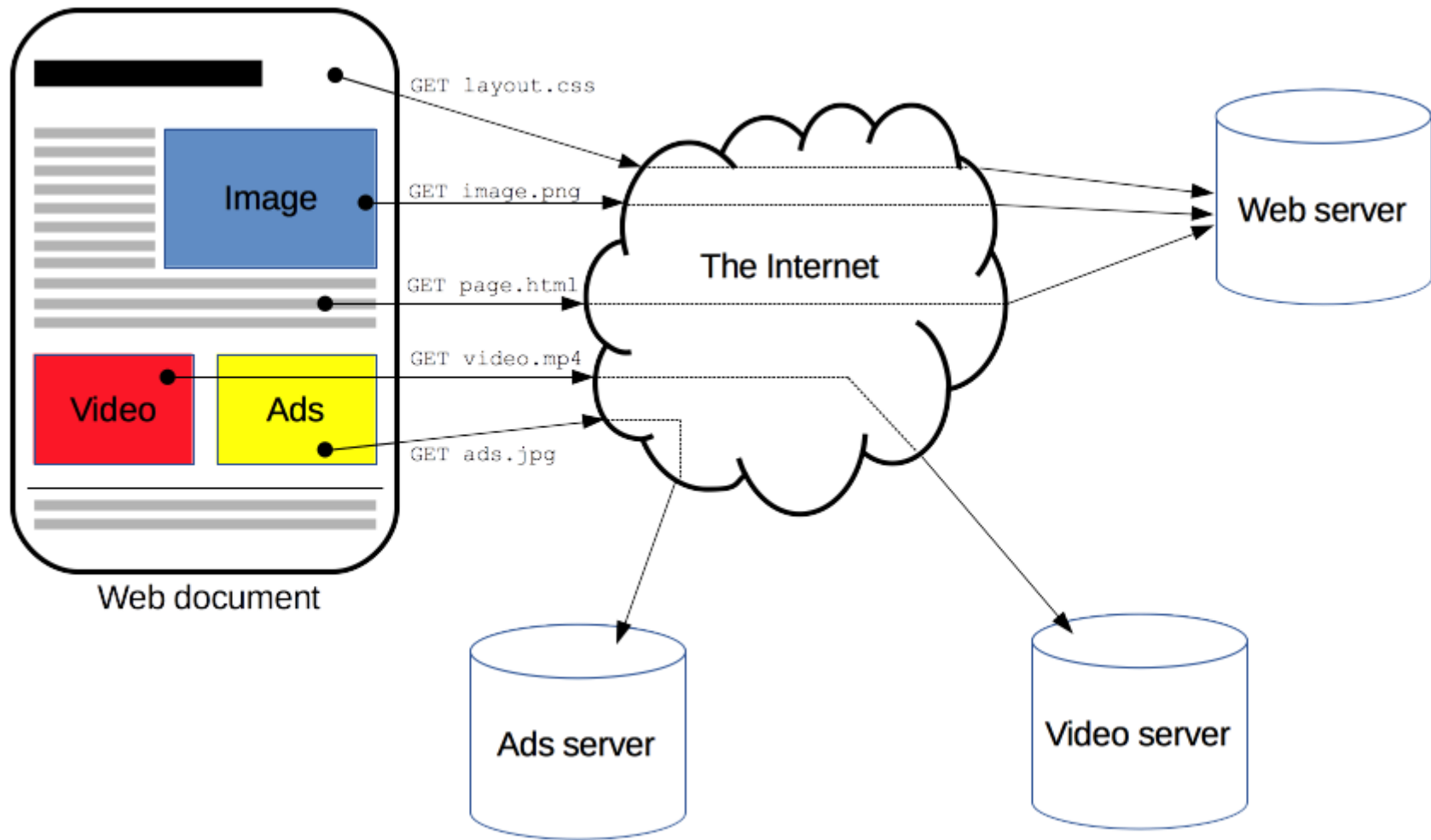
`http://www.google.com:8080/news/index.asp?boardID=5&ID=24618&page=1#name`

从上面的URL可以看出，一个完整的URL包括以下几部分：

- 1.协议部分：该URL的协议部分为“http：”，这代表网页使用的是HTTP协议。在Internet中可以使用多种协议，如HTTP，FTP等等本例中使用的是HTTP协议。在"HTTP"后面的“//”为分隔符
- 2.域名部分：该URL的域名部分为“www.google.com”。一个URL中，也可以使用IP地址作为域名使用
- 3.端口部分：跟在域名后面的是端口，域名和端口之间使用“:”作为分隔符。端口不是一个URL必须的部分，如果省略端口部分，将采用默认端口
- 4.虚拟目录部分：从域名后的第一个“/”开始到最后一个“/”为止，是虚拟目录部分。虚拟目录也不是一个URL必须的部分。本例中的虚拟目录是“/news/”
- 5.文件名部分：从域名后的最后一个“/”开始到“？”为止，是文件名部分，如果没有“?”,则是从域名后的最后一个“/”开始到“#”为止，是文件部分，如果没有“？”和“#”，那么从域名后的最后一个“/”开始到结束，都是文件名部分。本例中的文件名是“index.asp”。文件名部分也不是一个URL必须的部分，如果省略该部分，则使用默认的文件名
- 6.锚部分：从“#”开始到最后，都是锚部分。本例中的锚部分是“name”。锚部分也不是一个URL必须的部分
- 7.参数部分：从“？”开始到“#”为止之间的部分为参数部分，又称搜索部分、查询部分。本例中的参数部分为“boardID=5&ID=24618&page=1”。参数可以允许有多个参数，参数与参数之间用“&”作为分隔符。

两台计算机在使用HTTP通信在一条线路上的必须是一端为客户端，一端为服务器；
HTTP协议规定请求从客户端发出，最后服务器端响应该请求并返回；
HTTP是不保存状态，即无状态协议，于是为了实现保持状态功能引入了Cookie技术；

HTTP是一种能够获取如HTML这样网络资源的协议。它是Web上数据交换的基础，是一种client-server协议，也就是说请求通常是由像浏览器这样的接受方发起的。一个完整的web文档是由不同的子文档重新组建而成的，像是文本、布局描述、图片、视频、脚本等等。



http协议的使用

请求:客户端向服务器索要数据.

http协议规定:一个完整的http请求包含'请求行','请求头','请求体'三个部分;

请求行: 包含了请求方法,请求资源路径,http协议版本. "GET /resources/images/ HTTP/1.1"

请求头:包含了对客户端的环境描述,客户端请求的主机地址等信息.

Accept: text/html (客户端所能接收的数据类型)

Accept-Language: zh-cn (客户端的语言环境)

Accept-Encoding: gzip(客户端支持的数据压缩格式)

Host: m.baidu.com(客户端想访问的服务器主机地址)

User-Agent: Mozilla/5.0(Macintosh;Intel Mac OS X10.10 rv:37.0) Gecko/20100101Firefox/37.0(
客户端的类型,客户端的软件环境)

请求体:客户端发给服务器的具体数据,比如文件/图片等

GET / HTTP/1.0

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5)

Accept: */*

响应:服务器返回客户端想要的

数据
http协议规定:一个完整的http响应包含'状态行','响应头','实体内容'三个部分;

状态行:包含了http协议版本,状态码,状态英文名称.

"HTTP/1.1 200 OK"

响应头:包含了对服务器的描述,对返回数据的描述.

Content-Encoding: gzip(服务器支持的数据压缩格式) Content-Length: 1528(返回数据的长度)

Content-Type:application/xhtml+xml;charset=utf-8(返回数据的类型)

Date: Mon,15Jun201509:06:46GMT(响应的时间) Server: apache (服务器类型)

实体内容:服务器返回给客户端的具体数据(图片/html/文件...).

HTTP/1.0 200 OK

Content-Type: text/plain

Content-Length: 137582

Expires: Thu, 05 Dec 1997 16:00:00 GMT

Last-Modified: Wed, 5 August 1996 15:55:28 GMT

Server: Apache 0.84

<html>

<body>Hello World</body>

</html>

关于字符的编码，1.0版规定，头信息必须是 ASCII 码，后面的数据可以是任何格式。因此，服务器回应的时候，必须告诉客户端，数据是什么格式，这就是Content-Type字段的作用。

下面是一些常见的Content-Type字段的值。

text/plain

text/html

text/css

image/jpeg

image/png

image/svg+xml

audio/mp4

video/mp4

application/javascript

application/pdf

application/zip

application/atom+xml

这些数据类型总称为**MIME type**，每个值包括一级类型和二级类型，之间用斜杠分隔。除了预定义的类型，厂商也可以自定义类型。

application/vnd.debian.binary-package

上面的类型表明，发送的是**Debian**系统的二进制数据包。

MIME type还可以在尾部使用分号，添加参数。

Content-Type: text/html; charset=utf-8

上面的类型表明，发送的是网页，而且编码是**UTF-8**。

客户端请求的时候，可以使用**Accept**字段声明自己可以接受哪些数据格式。

Accept: */*

http协议定义了很多方法对应不同的资源操作,其中最常用的是GET和POST方法。

在请求URL后面以?的形式跟上发给服务器的参数,参数以"参数名"="参数值"的形式拼接,多个参数之间用&分隔;

GET的本质是从服务器得到数据,效率更高.并且GET请求可以被缓存.

注意:GET的长度是有限制的,不同的浏览器有不同的长度限制,一般在2~8K之间;

POST的本质是向服务器发送数据,也可以获得服务器处理之后的结果,效率不如GET.POST请求不可以被缓存,每次刷新之后都需要重新提交表单.

发送给服务器的参数全部放在'请求体'中;

理论上,POST传递的数据量没有限制.

注意:所有涉及到用户隐私的数据(密码/银行卡号等...)都要用POST的方式传递.

在一个网络中。传输数据需要面临三个问题:

- 1.客户端如何知道所求内容的位置?
- 2.当客户端知道所求内容的位置后, 如何获取所求内容?
- 3.所求内容以何种形式组织以便被客户端所识别?

对于WEB来说, 回答上面三种问题分别采用三种不同的技术, 分别为:统一资源定位符(URIs),超文本传输协议(HTTP)和超文本标记语言(HTML)。

学习web前端开发基础技术需要掌握：HTML、CSS、JavaScript语言。

1. HTML是网页内容的载体。内容就是网页制作者放在页面上想要让用户浏览的信息，可以包含文字、图片、视频等。

2. CSS样式是表现(外观控制)。就像网页的外衣。比如，标题字体、颜色变化，或为标题加入背景图片、边框等。所有这些用来改变内容外观的东西称之为表现。

3. JavaScript是用来实现网页上的特效效果。如：鼠标滑过弹出下拉菜单。或鼠标滑过表格的背景颜色改变。还有焦点新闻（新闻图片）的轮换。可以这么理解，有动画的，有交互的一般都是用JavaScript来实现的。

HTML文件是什么？

HTML表示超文本标记语言（Hyper Text Markup Language）。

HTML文件是一个包含标记的文本文件。

这些标记告诉浏览器怎样显示这个页面。

HTML文件必须有htm或者html扩展名。

HTML文件可以用一个简单的文本编辑器创建。

HTML是用于创建网页的语言。我们通过使用HTML标记标签创建html文档来创建网页。

HTML代表超文本标记语言。HTML是一种标记语言，它具有标记标签的集合。

HTML(HyperText MarkUp Language)超文本标记语言,通过使用标记来描述文档结构和表现形式的一种语言,由浏览器进行解析,然后把结果显示在网页上. 它是网页构成的基础,你见到的所有网页都离不开HTML,所以学习HTML是基础中的基础.

HTML标签是由尖括号（如<html>， <body>）包围的字词。标签通常成对出现，例如<html>和</html>。

一对中的第一个标签是开始标签;第二个标签是结束标签。在上面的示例中，<html>是开始标签，而</html>是结束标签。

我们还可以将开始标签称为起始标签，结束标签称为闭合标签。

HTML 是由各种各样的标签组成，学习 HTML 就是学习使用这些标签。

```
1  <html>
2      <head>
3          <title>网页标题</title>  <!--这是注释，在网页中不显示-->
4      </head>
5      <body>
6          <h1>内容标题（HTML标签演示页面）</h1>
7          <hr />
8          <p>这里是<b>具体</b>的内容，P标签表示是段落</p>
9          <a href='http://www.adminwang.com'>链接演示</a>
10     </body>
11 </html>
12
```

头部

<body> 主体部

HTML 网页

HTML标签是HTML文档的基本元素,它一般是成对出现的,即有开始标签和对应的结束标签构成. 如<p></p> <body></body> <head></head>等,但有些是特殊的单标签,如
 <hr />等. Html HTML语言是弱类型语言,标签不区分大小写<P>和<p>显示结果是相同的,不过标准推荐使用小写.

| 标签是HTML中最基本单位,也是最重要组成部分。

| HTML标签由开始标签和结束标签组成。

| 某些HTML元素没有结束标签，比如

| 标签是大小写无关的,<body>跟<BODY>表示意思是一样的，标准推荐使用小写.

| 所有的标签之间可以嵌套。例：<head> <title>标签嵌套演示</title></head>

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage.
<b>This text is bold</b>
</body>
</html>
```

HTML文档中，第一个标签是<html>。这个标签告诉浏览器这是HTML文档的开始。HTML文档的最后一个标签是</html>，这个标签告诉浏览器这是HTML文档的终止。

在<head>和</head>标签之间文本的是头信息。在浏览器窗口中，头信息是不被显示的。

在<title>和</title>标签之间的文本是文档标题，它被显示在浏览器窗口的标题栏。

在<body>和</body>标签之间的文本是正文，会被显示在浏览器中。

在和标签之间的文本会以加粗字体显示。

HTML 标题

HTML 标题（Heading）是通过<h1> - <h6> 标签来定义的.

<h1>这是一个标题</h1>

<h2>这是一个标题</h2>

<h3>这是一个标题</h3>

HTML 段落

HTML 段落是通过标签 <p> 来定义的.

<p>这是一个段落。</p>

<p>这是另外一个段落。</p>

HTML 链接

HTML 链接是通过标签 <a> 来定义的.

实例

这是一个链接

提示:在 href 属性中指定链接的地址。

HTML 图像

HTML 图像是通过标签 来定义的.

HTML文档可以包含的内容

通过不同的标签，HTML文档可以包含不同的内容，比如文本，链接，图片，列表，表格，表单，框架等。

文本

HTML对文本的支持是最丰富的，你可以设置不同级别的标题，分段和换行，可以指定文本的语义和外观，可以说明文本是引用自其它的地方，等等等等。

链接

链接用来指出内容与另一个页面或当前页面某个地方有关。

图片

图片用于使页面更加美观，或提供更多的信息。

列表

列表用于说明一系列条目是彼此相关的。

表格

表格是按行与列将数据组织在一起的形式。也有不少人使用表格进行页面布局。

表单

表单通常由文本输入框，按钮，多选框，单选框，下拉列表等组成，使HTML页面更有交互性。

框架

框架使页面里能包含其它的页面。

HTML文档可以用任何文本编辑器（比如记事本，UltraEdit等）创建，编辑，因为它的内容在本质也只是一些文本。

在HTML文本中，用尖括号括起来的部分称为标签。如果想在正文里使用尖括号（或者大与号小与号，总之是同一个东西），必须使用字符转义，也就是说转换字符的原有意义。<应该使用<代替，>则使用>，至于&符号本身,则应该使用&替代（不得不说的是有很多HTML文档没有遵循这个规则，常用的浏览器也都能够分析出&到底是一个转义的开始，还是一个符号，但是这样做是不推荐的）。

标签本质上是对它所包含的内容的说明，可能会有属性，来给出更多的信息。比如（图片）标签有src属性（用于指明图片的地址），width和height属性（用于说明图片的宽度和高度）。HTML里能使用哪些标签，这些标签分别可以拥有哪些属性，这些都是有规定的，知道了这里说的基本知识之后，学习HTML其实也就是学习这些标签了。本文后面会对常用的HTML标签做出简短的介绍。

标签通常有开始部分和结束部分（也被称为开始标签和结束标签），它们一起限定了这个标签所包含的内容。属性只能在开始标签中指定，属性值可以用单引号或双引号括起来。结束标签都以/加上标签名来表示。有时候，有些标签并不包含其它内容（只包括自己的属性，甚至连属性都没有），这种情况下，可以写成类似这样：。注意最后的一个空格和一个反斜杠，它说明这个标签已经结束，不需要单独的结束标签了。

大部分HTML标签都可以添加属性,常见的属性有宽度,高度,颜色,背景,字体等等

| HTML属性一般都出现在HTML标签中, 是HTML标签的一部分。

| 标签可以有属性,它包含了额外的信息.属性的值一定要在双引号中。

| 标签可以拥有多个属性。 标签可以拥有多个属性。

| 属性由属性名和值成对出现。

语法格式

<标签名属性名1="属性值" 属性名2="属性值" ... ">.....</标签名>

```
1 <a href="www.adminwang.com">
2   演示链接
3 </a>
```

这是一个HTML链接
元素

```
5 <p>
6   <a href="www.adminwang.com">
7     演示链接2
8   </a>
9 </p>
```

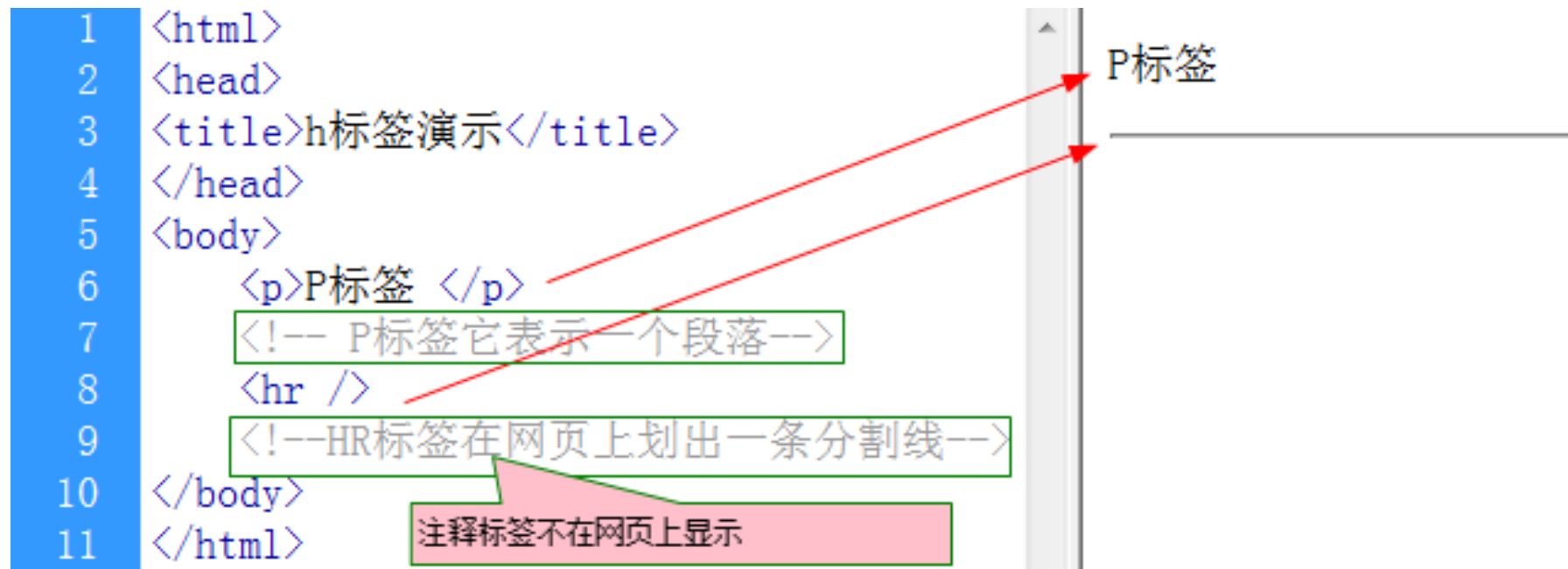
这是一个HTML段落
元素, 它包含了一个
html链接元素。

在实际开发中需要在要在一些代码段做HTML注释,这样做的好处很多,比如:方便查找,方便比对,方便项目组里的其它程序员了解你的代码,而且可以方便以后你对自己代码的理解与修改等等

语法格式:

<!--这里写注释内容 -->

注释: 开始括号之后(左边的括号)需要紧跟一个叹号, 结束括号之前(右边的括号)不需要。



某些标签包含的内容中还可以有新的标签，新的标签名甚至可能还可以与包含它的标签的名称相同（哪些标签可以包含标签，可以包含哪些标签也是有规定的）。比如：

点击查看效果

```
<div>
```

```
  <div>分类目录...</div>
```

```
  <div>当前分类内容列表...</div>
```

```
</div>
```

在这种情况下，最后出现的标签应该最先结束。

HTML文档里所有的空白符（空格，**Tab**，换行，回车）会被浏览器忽略，唯一的例外是空格，对空格的处理方式是所有连续的空格被当成一个空格，不管有一个，还是两个，还是100个。之所以有这样的规则是因为忽略空白符能让使用HTML的作者以他觉得最方便的格式来排列内容，比如可以在每个标签开始后增加缩进，标签结束后减少缩进。由于英语文本中空格用得很普遍（用于分隔单词），所以对空格做了这样的特殊处理。如果要显示连续的空格（比如为了缩进），应该用** **来代表空格。

常用标签介绍

文本

最常用的标签可能是了，它用于改变字体，字号，文字颜色。

点击查看效果

6

4

红色的5

黑体的字

加粗，下划线，斜体字也是常用的文字效果，它们分别用,<u>,<i>表示：

点击查看效果

Bold

<i>italic</i>

<u>underline</u>

HTML文档在浏览器里通常是从左到右，从上到下地显示的，到了窗口右边就自动换行。为了实现分栏的效果，很多人使用表格（<table>）进行页面排版（虽然HTML里提供表格的本意不是为了排版）。

<table>标签里通常会包含几个<tr>标签，<tr>代表表格里的一行。<tr>标签又会包含<td>标签，每个<td>代表一个单元格。

点击查看效果

```
<table>
```

```
  <tr>
```

```
    <td>2000</td><td>悉尼</td>
```

```
  </tr>
```

```
  <tr>
```

```
    <td>2004</td><td>雅典</td>
```

```
  </tr>
```

```
  <tr>
```

```
    <td>2008</td><td>北京</td>
```

```
  </tr>
```

```
</table>
```


HTML 是由各种各样的标签组成，学习 HTML 就是学习使用这些标签。

作业


<http://www.halifax.ca/newcomers/WelcomingNewcomers.php#Holidays>

抓取出ns省的节日

HashMap 基于value的排序

钱进培训是哈法地区资深工程师组成的培训机构，通过各位老师的现身说法，帮助各位学员迅速掌握实战知识，为求职打下坚实的基础。电子邮件：jin.qian.canada@gmail.com钱老师报名、答疑微信号：qianjincanada，或扫描以下二维码添加：



钱老师 
加拿大



扫一扫上面的二维码图案，加我微信

 钱进老师



 钱进老师