目录

第一章　JavaScript 与面向对象

本章要点

- 解释什么是对象, 什么是面向对象, 以及面向对象编程是什么.
- 能解释什么是面向过程编程, 什么是面向对象编程.
- 能利用面向对象的方式对需求进行分析, 抽取出需要的对象.
- 能简要分析面向对象与面向过程之间的关系.

1.1　什么是对象

所谓对象(object), 即目标. 就是你所研究的目标, 具体的东西. 在前面我们已经学习过一些 内置对象 了. 例如: 数组, 字符串包装对象, 时间对象等. 你么我们回忆一下这些对象的使用. 同时思考一下对象的特征是什么?

1.1.1　内置对象的使用复习

我们分别讨论一下数组, 字符串包装对象和时间对象.

数组对象

首先考虑数组

```
1  var arr = [ 1, 2, 3, 4 ];
```

要对数组进行操作, 我们就需要使用数组提供的 *方法*, 例如要反转数组, 需要使用 `reverse` 方法; 要提取数组中的某一个子字符串, 则需要使用 `slice` 方法; 要判断数组中是否含有某一个字符, 则需要使用 `indexOf` 方法[1];如果需要往数组中添加一个元素, 则使用 `push` 方法, 或 `unshift` 方法; 如果是要从数组中移除某个元素, 可以使用 `pop` 方法, `shift` 方法, 或 `splice` 方法[2].

[1] `indexOf` 方法是 ES5 中引入的方法, 对于低版本的 IE 浏览器是不支持的. 需要自行定义

[2] 其具体使用语法可以参考代码: 01-01-01-array.html

字符串包装对象

然后我们讨论字符串的包装对象. 原本字符串是基本类型, 并不是对象. 但是在JavaScript中为了统一处理数据, 提供了基本类型的包装对象: 每一个基本类型依旧可以点出方法, 并调用得到结果(对于数字类型与布尔类型相对讨论较少, 但是对于字符串, 系统提供了很多的方法). 其本质是, 在调用的时候, 基本类型会转换成对象类型, 然后由对象调用对应的方法, 得到结果后对象就会被销毁. 这个过程虽然我们无法直接观察到, 但是我们需要了解其过程.

对于字符串

```
1  var str = 'hello JavaScript string';
```

如果要计算该字符串中有多少个单词, 可以使用 `split` 方法, 用空格分隔字符串, 然后计算得到的数组结果的长度; 如果要判断这个字符串中是否含有某一个子字符串, 则可以使用 `indexOf` 方法; 如果需要将字符串全部转换成大写或小写, 则可以使用 `toUpperCase` 或 `toLowerCase` 方法[3]. 很显然方法全部集中在一起, 然后需要使用变量点出来使用.

时间对象

前面我们讨论了数组, 字符串. 它们都是对象, 可以保存数据, 同时还有很多方法可以对数据进行操作. 接下来我们再来看看时间对象.

对于时间对象

```
1  var now = new Date();
```

使用 `new Date()` 创建一个时间对象, 对象中存储了从1970年1月1日0点0分0秒到现在的毫秒数, 即时间戳. 对象除了存储该时间戳外, 还提供了很多的方法. 例如: 要获得当前时间的年份数, 可以使用 `getFullYear` 方法; 如果要获得当前的月份 `getMonth` 方法[4].如果要获得当前时间的小时, 分钟以及秒数, 可以使用 `getHours` 方法, `getMinutes` 方法, 以及 `getSeconds` 方法[5].

总结特征

从面前的几个案例我们可以归纳出对象的特征:

1. 对象不是一个简单的数据类型, 其包含的东西很多(很直观的感觉).
2. 对象保存了数据, 数组对象保存的就是数组中的各个元素, 字符串的包装对象里面保存着字符串的数据, 时间对象保存的当前时间戳. 实际上对象可以保存很多数据, 不仅仅就这么一点.
3. 对象中保存着很多的方法, 而这些方法都是用来操作保存的数据用的. 例如, 数组的方法, 用于操作当前数组中的每一个元素; 字符串方法就是在操作当前字符串的每一个字符; 而时间方法实际上就是在时间戳上进行计算, 来转换成这个时间对应的年, 月, 日, 时, 分, 秒等.

[3]其具体使用语法可以参考代码: 01-01-02-string.html
[4]月份的计算从0开始, 即返回0表示一月份, 返回11表示第十二月份
[5]其具体使用语法可以参考代码: 01-01-03-date.html

通过前面的小结. 我们不难想象: 对象就是一个复合结构, 其中包含了数据和方法. 数据用于描述对象的特征与信息; 而方法用途描述对象具有的能力与行为.

1.1.2 对象的概念

对象(Object), 是一种数据的组织方式, 是一个抽象的概念. 是程序员组成代码的一种方法. 前面我们复习了常见的内置对象, **对象就是数据与方法的综合体**. 在早期的编程方法中, 数据单独定义, 函数也是单独定义, 将数据与操作数据的函数分开编写, 使用的时候在合到一起. 这样对于小规模的程序问题并不算太糟. 但是当多人协作开发, 以及大量代码与功能组合后, 维护代码将变得十分困难. 例如, 当多人分别开发不同功能的时候, 变量的命名就有可能造成代码工作时出现问题. 而针对 JavaScript 这一解释型编程语言, 重复定义变量又不会提示与警告, 这对调错又进一步增加了困难.

而将数据与功能"打包"到一起后, 就可以自己来维护与管理自己, 从而使得编码与维护变得相对容易. 当然对于小规模的代码, 依旧会变得糟糕. 因为可能一句话就可以解决的事情, 由于对象的加入, 可能需要好几句话才可以调用完成. 这也是初学者最容易困惑的地方. 但是也不用担心, 在积累一定代码与功能后, 初学者也可以很快感受到使用对象来组织代码的便利之处.

在接下来的篇幅中, 我们一一讨论面向对象的所有主题, 来体会面向对象所带来的优势.

1.2 面向对象编程

在讨论面向对象编程之前, 我们需要了解一下什么是面向过程. 我们再对比面向过程与面向对象之间的的关系.

1.2.1 面向过程编程

面向过程(Procedure Oriented), 就是以过程为中心的一种编程思想. 面向, 就是指使用, 关注, 以什么的方式的含义. 面向过程就是以过程的方式, 通过处理每一件事情来处理代码. 那么过程就需要关注每一个步骤与细节, 对于开发者, 必须关注到每一件事情, 每一个函数的调用.

用一个比较简单的例子来说明:

> 例如我们考虑一下吃早饭. 如果是在大年初一的早上. 我们要吃面怎么处理呢? 既然大年初一, 街上肯定没有小吃, 只能自己做了. 所以我们需要考虑一下做面的步骤了.
>
> 1. 首先需要先和面. 把面饭和水混合到一起, 然后揉啊揉, 揉啊揉, ...
> 2. 不知过去了多少时间, 然后将和好的面压成细面(这还需要压面机, 否则只有自己拉了...)
> 3. 然后生火烧水, 与此同时可以做点臊子(就是加载面中调味使用的配菜)
> 4. 待水烧开即可下面, 接着就是等待, 等待. 还需不时的捞一捞, 以免面粘锅.

5. 最后就可以迟到好吃的面条了. 只可惜最后还需要洗锅, 洗完...

看来吃个面也不是那么容易的.

说到这里, 不难发现这个与我们现在吃面的方式不太相同. 现在大多数人想要吃面, 直接找个面馆, 然后点一碗面即可.

在这个案例中, 就是面向过程的. 在这个制作面的过程中, 我们需要亲力亲为, 需要自己一步一步的处理, 不能先下面, 再和面; 也不能先吃面条再煮面. 所有的一切都说明每一个细节步骤都不能跳过, 也不能随便处理. 只有每一个步骤做到极致, 面才可口好吃.

结合到编码中, 就是在完成一段逻辑的时候, 考虑先做什么, 后做什么, 然后接着做什么. 即为面向过程的思考方式.

我们可以思考下面的案例:

例1 在页面中创建一个select标签, 标签中的选项为: 北京, 上海, 广州, 和深圳. 其取值分别为1, 2, 3, 4.

在这个案例中我们要创建一个select标签. 按照我们常规的想法:

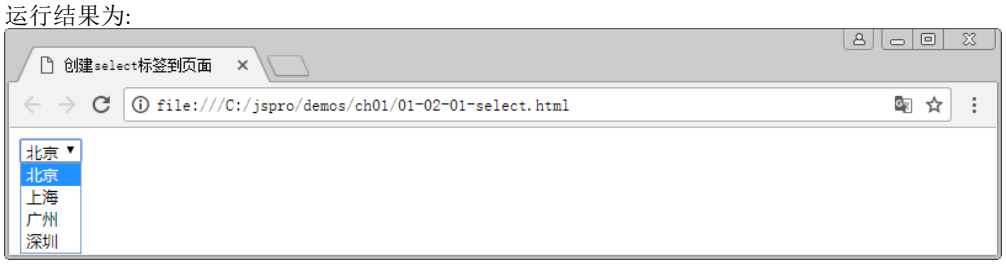
1. 首先调用`document.createElement`方法创建select对象.
2. 然后接着调用`document.createElement`方法创建option对象.
3. 接着使用对象的`innerHTML`属性, 赋值内部文本.
4. 使用`value`属性来设置选中后的取值.
5. 接着使用select对象的`appendChild`方法, 来追加option对象.
6. 使用循环, 根据“北京”, “上海”, “广州”, “深圳”数据循环生成option对象依次加入.
7. 最后将select对象加到页面中.

参考代码为[6]:

```
1  var select = document.createElement( 'select' );
2  var data = [ '北京', '上海', '广州', '深圳' ];
3  for ( var i = 0; i < data.length; i++ ) {
4      var option = document.createElement( 'option' );
5      option.value = i + 1;
6      option.innerHTML = data[ i ];
7      select.appendChild( option );
8  }
9  document.getElementById( 'dv' ).appendChild( select );
```

[6]完整代码请参考: 01-02-01-select.html

运行结果为:



在这段代码中所有的考虑都是一步步完成的, 不能颠倒也不能省略某一个步骤. 这便是面向过程的编码方式.

1.2.2 面向对象编程

面向对象编程(OOP, Object Oriented Programming), 就是使用对象进行编程. 即在需要某一个任务的时候, 不再考虑自己亲力亲为, 而是找到一个对象, 要求对象帮助你完成.

还是来考虑吃面的例子. 在面向对象的思想中, 要吃面了, 不是考虑怎么去做, 而是考虑先找到一个可以做面的对象, 即面馆. 然后告知你要吃什么面. 然后等待即可得到一碗符合你要求的面. 这就是面向对象的思想.

实际上在生活中我们时时刻刻的来利用对象帮我们完成所需:

- 我们要到某一个地方, 我们不需要了解先到哪里, 然后再怎么走. 叫一个出租车(滴滴)即可(面向滴滴出行).
- 我们要买食材, 不需要到菜地里挑选, 找到菜市场全有了(面向菜市场购物).
- 我们要洗衣服的时候, 不用手洗, 找到洗衣机, 放入衣服和洗衣液, 按下按钮即可(面向洗衣机).
- 我们要写代码, 不需要一步一步思考处理步骤, 找到对象, 告诉它要做什么, 它就会处理好.
- ... ...

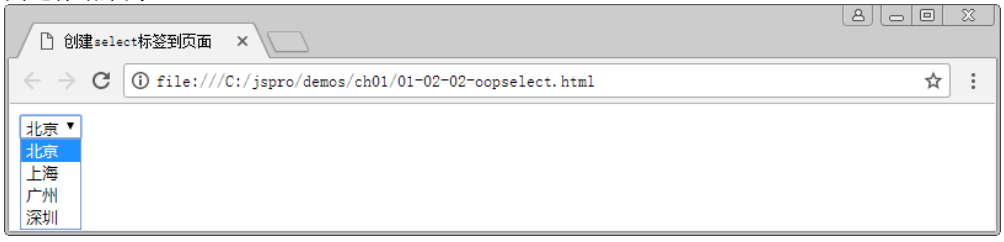
这样的案例还有很多, 我们再来仔细看看使用 OOP 的方式如何创建 select 标签.

这里我们暂时只是来说明面向对象与面向过程的区别, 暂时不考虑技术, 所以代码只给出了使用对象的片段[7]:

```
1  var select = new Select({
2      container: document.getElementById( 'dv' )
3      , data: [ '北京', '上海', '广州', '深圳' ]
4  });
5  select.init();
```

[7]完整的代码可以参考: 01-02-02-oopselect.html

其运行结果为:



在这段代码中, 我们需要创建一个下拉列表, 即 select 标签. 我们只需要先找到 `Select` 构造函数. 然后创建 `Select` 对象. 同时告诉对象我们需要将创建的标签加到哪一个容器中. 以及我应该使用什么数据来显示下拉列表.

这便是使用面向对象的思想在组织代码.

1.2.3 怎样抽取对象

我们可以发现, 面向对象与面向过程有一个重要的不同点. 就是面向过程, 重点在考虑如何做? 每一步该怎么写? 而面向对象在考虑, 做什么? 什么对象可以以做?

那么问题也来了, 在进行开发的时候怎么找对象呢?

显然, 就需要学习和积累了. 如 C#, Java 等这一类编程语言(或平台), 提供了大量的类库. 我们在学习和工作中不断积累. 了解各种库的使用方法与优劣. 在遇到需要解决的问题时自然就知道该用什么对象了. 但是如果按照当下我们的经验, 不足以知道应该使用什么库呢? 那就需要自己来进行编写对象了.

前面我们了解到, 所谓的对象就是数据与方法的结合体. 所以编写对象就是在编写需要存储什么数据, 以及需要提供什么方法. 那么首先我们得知道我们需要什么对象. 看 图1.1, 其中我们可以用自己的话来描述一下界面上都有什么东西. 例如, 从上往下, 首先有一个大的名字: 叫“跳一跳”, 然后是一个方盒子(搁棋子用), 接着是一个搁着棋子的方盒子, 然后一个按钮, 上面写着“开始游戏”, 最后还有一个排行榜.

用自己的话描述完这个界面后, 我们可以简单分析一下, 在我们描述的这段话中, 有几个名词?

1. 一个标题: “跳一跳”
2. 两个方盒子
3. 一个棋子
4. 开始游戏的按钮
5. 排行榜按钮

一共 6 个名词. 那么我们可以得到 6 个对象.

也就是说, 用自己的话描述界面, 有几个名词就可以得到几个对象. 我们称这个方法为*名词提炼法*.

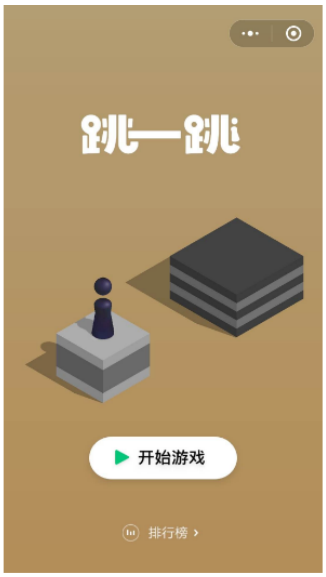


图 1.1: 跳一跳小游戏

对象有了, 接下来就需要讨论一下对象有什么数据与方法了.

标题对象 标题对象中一个非常重要的数据就是标题内容. 即存储显示字符串. 除此之外还有坐标. 该标题在页面的上部的正中间. 因此还需要存储坐标, 以及标题的宽度高度. 再细化, 还需要存储字体, 前景颜色, 背景颜色等.

因此该对象可以抽取出的数据有: 显示文本, 距离屏幕左边框的长度, 距离屏幕顶边框的长度, 字体, 前台颜色, 和背景颜色.

盒子对象 屏幕中有两个盒子, 即两个盒子对象. 每一个盒子对象除了表面颜色, 大小不一样外, 其结构是相同的, 都是一个立方体.

因此可以抽取的数据有: 坐标, 矩形边长, 高, 以及外部贴图.

棋子对象 棋子可以在界面中跳来跳去, 因此, 棋子有坐标. 除此之外棋子的外形也可以抽取出数据来. 例如颜色, 外形曲线等[8].

棋子可以跳, 因此它还具有 `jump` 方法.

按钮对象 界面中有两个按钮, 一个是开始游戏, 一个是排行榜. 都是点击后可以做指定的事情.

开始游戏按钮点击后即可进入游戏. 从界面上看, 按钮上有一个绿色的图标, 有一段文本. 因此开始按钮可以抽取出的数据有, 坐标, 文本, 字体, 前后背景色, 宽高, 还有图标.

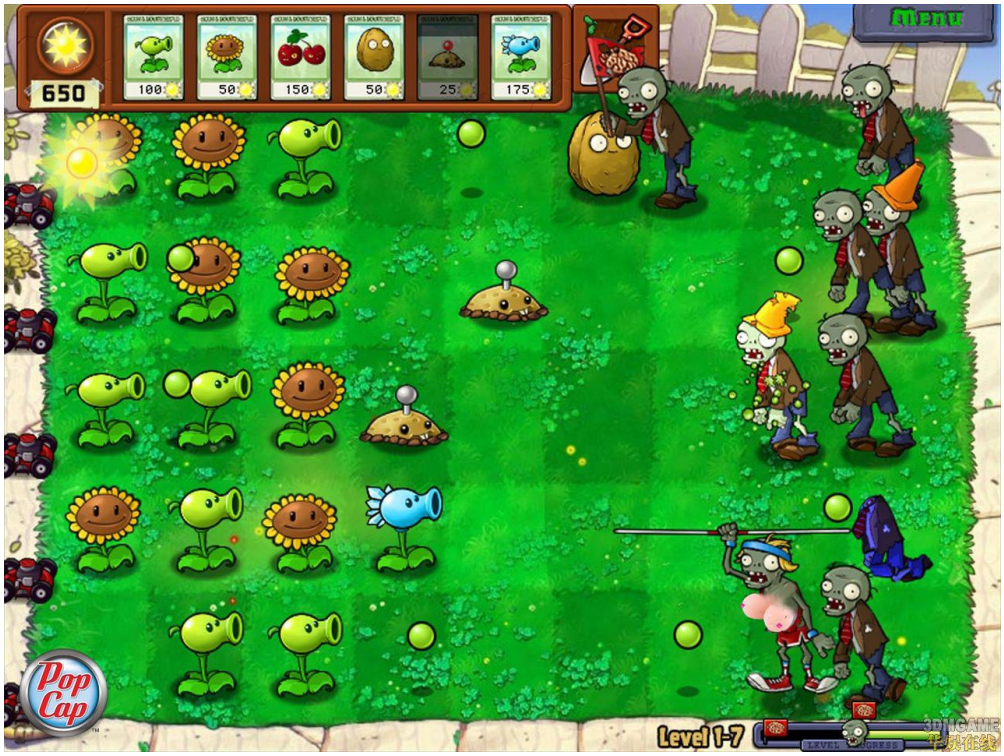
[8]这里简化了棋子, 将其看成一个点

排行榜按钮与开始按钮类似, 也有图标和文字, 而且右边还有一个箭头. 同样可以抽取的数据有, 坐标, 文本, 字体, 前后背景色, 宽高, 还有图标.

因此按钮可以抽取的数据有, 显示的文本, 显示的图标, 前景颜色, 背景颜色, 按钮宽度, 按钮高度, 按钮距离屏幕左边与顶边的距离等.

我们可以看一个更为复杂的例子:

例2 看下面植物大战僵尸的游戏截图:



在这个界面上有什么对象呢?

简单分析这张游戏截图:

- 整个界面背景可以看成一张大的铺满屏幕的图片, 可以抽取成一个对象. 当然由于铺满整个界面, 因此也就不需要考虑其他数据了, 就一个图片数据.
- 图片左上角有一个植物等待进度面板. 可以抽取成对象. 其包含的数据可以有: 坐标, 宽高, 植物列表, 以及积累的太阳数量等.
- 下方有种植的植物—太阳花. 作为对象需要保存的数据有, 坐标, 宽高, 以及产生小太阳的时间间隔. 还有一个特殊的数据, 就是在僵尸吃植物的时候, 需要吃几次才可以消除该植物. 即还需要存储血量. 同时太阳花提供生产小太阳的方法.

- 除了太阳花, 还有豌豆, 豌豆的存储的数据与太阳花类似, 有坐标, 宽度与高度, 血量. 存储的时间间隔不是用来生成小太阳花, 而是用来发射豌豆. 即发射豌豆的时间间隔. 同时它还需要提供发射豌豆方法.

- 地雷. 地雷需要的数据有坐标, 宽高, 以及等待的时间间隔. 同时提供爆炸方法.

- 僵尸. 僵尸存储的有坐标, 移动速度等. 由于僵尸可以被豌豆等消灭, 所以僵尸也有血量. 僵尸所提供的的方法有移动和吃.

由于现在主要考虑的是如何抽取对象. 所以这里还可以继续分析其成员与方法. 由于本节主要是介绍怎么提取对象, 至于怎么实现对象还需要后续的课程学习.

1.3 面向对象与面向过程的关系

前面我们已经看到, 面向对象编程总结起来就是, 要做什么事情, 找到合适的对象, 告诉它去完成即可. 很显然面向对象非常好, 那是不是就不需要面向过程了呢? 既然面向对象这么好, 那谁还去学习面向过程呢? 那么, 我们首先来看看使用面向对象的策略:

1. 首先分析问题, 看系统是否提供相应对象可供使用. 如果有直接使用即可.

2. 如果没有, 系统没有提供对应的功能, 查看是否有第三方提供相应的解决方案. 例如是否有相应的库, 可以帮助我们快速解决问题. 如果有, 下载下来直接使用.

3. 若系统不支持, 第三方库也没有, 货支持的不够好, 那么我们就需要自己开发了.

很显然面在有内置对象或第三方对象时, 我们可以很轻松的面向对象开发. 什么意思呢?

- 例如, 要对数组排序, 我们不需要关系数组的排序算法是怎样的. 虽然我们已经熟练使用冒泡排序, 但是, 针对大量数据, 冒泡排序的性能还是非常非常低的. 因此在系统内部, 排序采用了一些策略. 但是我们根本不用知道它怎么处理, 因为它已经面向对象了. 我们只要告诉它我们要排序即可.

- 再比如, 我们要获得今天的年份数, 月份数, 或日期. 我们知道, `Date` 对象存储了当前时间戳. 它是一个以毫秒为单位的数字. 如果要计算出年份, 想一想该怎么算? 除以毫秒, 分钟, 小时, 天, ... 还要加上闰年? 很明显不是那么好计算. 但是我们不担心, 因为有方法, `getFullYear` 方法, `getMonth` 方法, 以及 `getDate` 等方法, 可以很容易的取得索要的数据. 根本不用担心如何计算

- ... ...

但是没有内置对象或第三方对象的时候呢? 虽然我们也可以编写对象. 编写方法. 但是我们在编写方法的时候依旧是在一步一步的书写. 也就是说, 我们在写对象的时候, 也就是在打包面向对象.

所以面向对象实际上是对面向过程的“打包”, 将繁琐的处理步骤写到方法中, 使得我们在使用的时候尽可能的简单. 我们将其称为封装性(encapsulation). 即对象底层封装的就是过程. 过程是实现, 没有过程就没有对象.

1.4 小结

1. 面向对象是对面向过程的封装. 不是谁取代谁的关系.
2. 面向对象是一种代码组织的思想, 是一种处理方式. 本质没有添加新的技术, 只是在组织代码的时候更加便于使用.
3. 使用面向对象编程的基本处理: 首先明确问题, 然后查看是否有内置对象或第三方对象提供实现, 最后考虑自定义实现.
4. 在分析实际业务的时候, 抽取对象的方法可以使用名词提炼法. 即用自己的话描述一下业务内容, 然后在这句话中寻找名字. 凡是名字(或名词)就可以抽取为对象. 然后在根据设计来确定对象存储什么数据, 含有什么方法.

第二章　JavaScript 使用对象

2.1　怎么创建对象

2.1.1　使用 `new Object()` 来创建对象

2.1.2　使用对象的字面量(literal)来创建对象

2.1.3　使用工厂函数(factory function)来创建对象

2.2　构造函数

2.3　案例