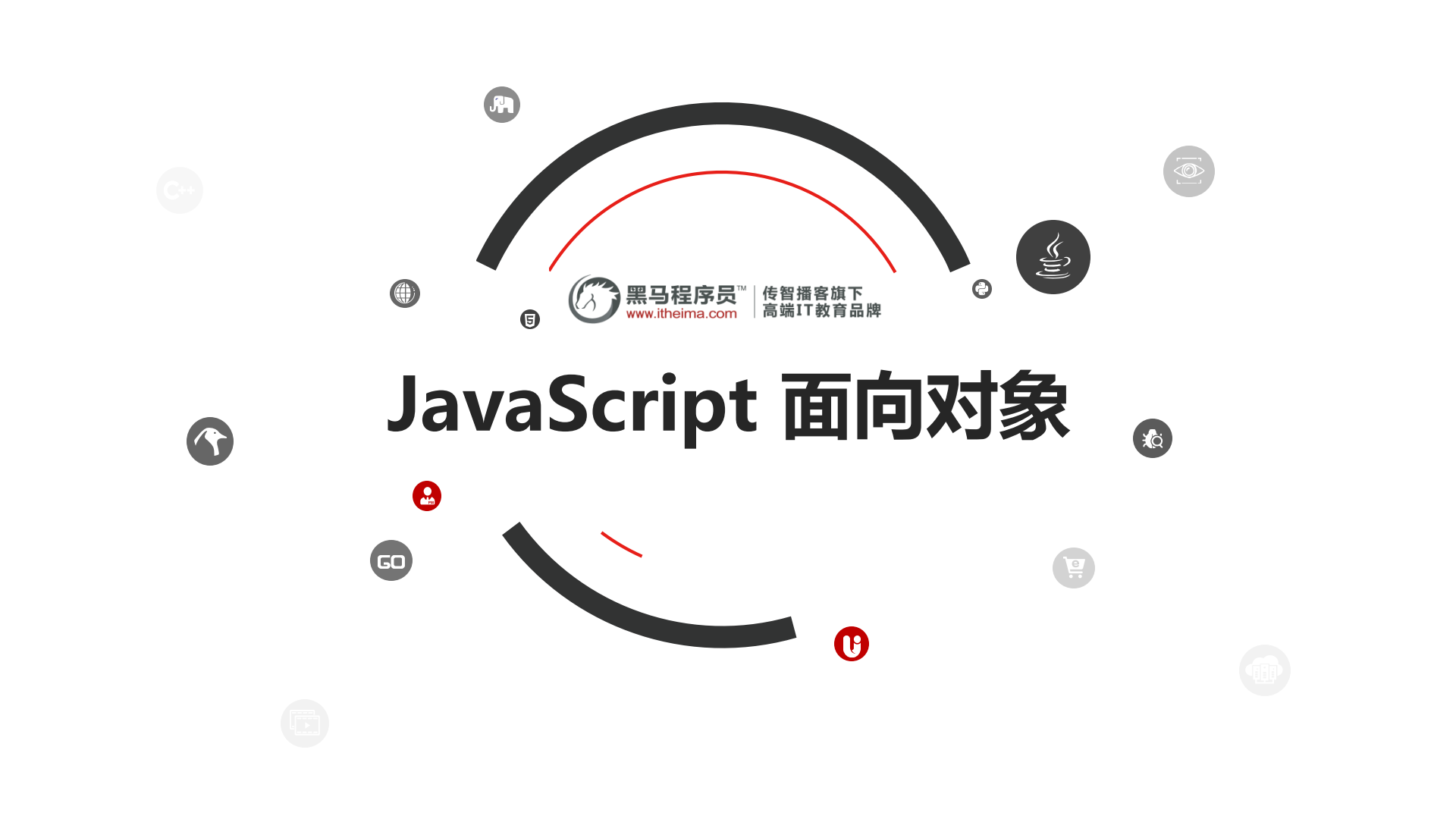


JavaScript 面向对象

目录 Contents

1. 面向对象编程介绍

1.1 两大编程思想

- 面向过程
- 面向对象

1. 面向对象编程介绍

1.2 面向过程编程 POP(Process-oriented programming)

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候再一个一个的依次调用就可以了。

举个栗子：将大象装进冰箱，面向过程做法。

1.打开冰箱门　　2.大象装进去　　3.关上冰箱门

面向过程，就是按照我们分析好了的步骤，按照步骤解决问题。

1. 面向对象编程介绍

1.3 面向对象编程 OOP (Object Oriented Programming)

面向对象是把事务分解成为一个个对象，然后由对象之间分工与合作。

举个栗子：将大象装进冰箱，面向对象做法。

先找出对象，并写出这些对象的功能：

1. 大象对象

- 进去

2. 冰箱对象

- 打开
- 关闭

3. 使用大象和冰箱的功能

面向对象是以对象功能来划分问题，而不是步骤。

1. 面向对象编程介绍

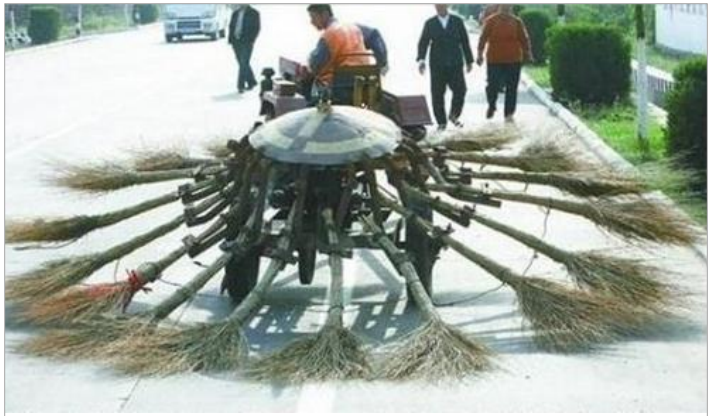
1.3 面向对象编程 OOP (Object Oriented Programming)

在面向对象程序开发思想中，每一个对象都是功能中心，具有明确分工。

面向对象编程具有灵活、代码可复用、容易维护和开发的优点，更适合多人合作的大型软件项目。

面向对象的特性：

- 封装性
- 继承性
- 多态性



继承：继承自拖拉机，实现了扫地的接口

封装：无需知道如何运作，开动即可

多态：平时扫地，天热当风扇

重用：没用额外动力，重复利用了发动机能量

多线程：多个扫把同时工作

低耦合：扫把可以换成拖把而无须改动

组件编程：每个配件都是可单独利用的工具

适配器模式：无需造发动机，继承自拖拉机，只取动力方法

代码托管：无需管理垃圾，直接扫到路边即可

1. 面向对象编程介绍

1.4 面向过程和面向对象的对比

面向过程

- 优点：性能比面向对象高，适合跟硬件联系很紧密的东西，例如单片机就采用的面向过程编程。
- 缺点：没有面向对象易维护、易复用、易扩展

面向对象

- 优点：易维护、易复用、易扩展，由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统 更加灵活、更加易于维护
- 缺点：性能比面向过程低

用面向过程的方法写出来的程序是一份蛋炒饭，而用面向对象写出来的程序是一份盖浇饭。

目录 Contents

2. ES6 中的类和对象

面向对象

面向对象更贴近我们的实际生活, 可以使用面向对象描述现实世界事物. 但是事物分为具体的事物和抽象的事物

手机　　抽象的(泛指的)

具体的(特指的)

面向对象的思维特点:

1. 抽取（抽象）对象共用的属性和行为组织(封装)成一个类(模板)
2. 对类进行实例化, 获取类的对象

面向对象编程我们考虑的是有哪些对象，按照面向对象的思维特点,不断的创建对象,使用对象,指挥对象做事情.

2. ES6 中的类和对象

2.1 对象

现实生活中：万物皆对象，对象是一个具体的事物，看得见摸得着的实物。例如，一本书、一辆汽车、一个人可以是“对象”，一个数据库、一张网页、一个与远程服务器的连接也可以是“对象”。

在 JavaScript 中，对象是一组无序的相关属性和方法的集合，所有的事物都是对象，例如字符串、数值、数组、函数等。

对象是由属性和方法组成的：

- 属性：事物的**特征**，在对象中用**属性**来表示（常用名词）
- 方法：事物的**行为**，在对象中用**方法**来表示（常用动词）

2. ES6 中的类和对象

2.2 类 class

在 ES6 中新增加了类的概念，可以使用 class 关键字声明一个类，之后以这个类来实例化对象。

类抽象了对象的公共部分，它泛指某一大类（class）

对象特指某一个，通过类实例化一个具体的对象

2. ES6 中的类和对象

2.2 类 class

类抽象了对象的公共部分，它泛指某一大类（class）

对象特指某一个，通过类实例化一个具体的对象

面向对象的思维特点:

1. 抽取（抽象）对象共用的属性和行为组织(封装)成一个**类**(模板)
2. 对类进行实例化, 获取类的**对象**

2. ES6 中的类和对象

2.3 创建类

语法：

```
class name {
  // class body
}
```

创建实例：

```
var xx = new name();
```

注意： 类必须使用 new 实例化对象

2. ES6 中的类和对象

2.4 类 constructor 构造函数

constructor() 方法是类的构造函数(默认方法)，用于传递参数,返回实例对象，通过 new 命令生成对象实例时，自动调用该方法。如果没有显示定义, 类内部会自动给我们创建一个constructor()

语法:

```
class Person {
  constructor(name,age)  {    // constructor  构造方法或者构造函数
      this.name = name;
      this.age = age;
    }
}
```

创建实例:

```
var ldh = new Person('刘德华', 18);
console.log(ldh.name)
```

2. ES6 中的类和对象

2.5 类添加方法

语法：

```
class Person {
  constructor(name,age)  {    // constructor  构造器或者构造函数
      this.name = name;
      this.age = age;
    }
   say() {
      console.log(this.name  + '你好');
   }
}
```

创建实例：

```
var ldh = new Person('刘德华', 18);
ldh.say()
```

注意： 方法之间不能加逗号分隔，同时方法不需要添加 function 关键字。

目录 Contents

3. 类的继承

3.1 继承

现实中的继承：子承父业，比如我们都继承了父亲的姓。

程序中的继承：子类可以继承父类的一些属性和方法。

语法：

```
class Father{    // 父类
}
class  Son extends Father {  // 子类继承父类
}
```

3. 类的继承

3.1 继承

实例：

```
class Father {
      constructor(surname) {
        this.surname= surname;
      }
      say() {
        console.log('你的姓是'  + this.surname);

       }
}
class Son extends Father{   // 这样子类就继承了父类的属性和方法

}
var damao= new Son('刘');
damao.say();
```

3. 类的继承

3.2 super 关键字

super 关键字用于访问和调用对象父类上的函数。可以调用父类的构造函数，也可以调用父类的普通函数

语法：

```
class Person {    // 父类
      constructor(surname){
         this.surname = surname;
      }
}
class  Student extends Person {        // 子类继承父类
     constructor(surname,firstname){
          super(surname);              // 调用父类的constructor(surname)
          this.firstname = firstname; // 定义子类独有的属性
      }
}
```

注意: 子类在构造函数中使用super, 必须放到 this 前面（必须先调用父类的构造方法,在使用子类构造方法）

3. 类的继承

3.2 super 关键字

案例：

```
 class Father {
    constructor(surname) {
        this.surname = surname;
     }
    saySurname() {
      console.log('我的姓是' + this.surname);
    }
}
class Son extends Father { // 这样子类就继承了父类的属性和方法
    constructor(surname, fristname) {
         super(surname);    // 调用父类的constructor(surname)
         this.fristname = fristname;
     }
    sayFristname() {
         console.log("我的名字是：" + this.fristname);

    }
}
var damao = new Son('刘', "德华");
damao.saySurname();
damao.sayFristname();
```

3. 类的继承

3.2 super 关键字

super关键字 用于访问和调用对象父类上的函数。可以调用父类的构造函数，也可以调用父类的普通函数。

语法：

```
class Father {
     say() {
         return '我是爸爸';

     }
}
class Son extends Father { // 这样子类就继承了父类的属性和方法
     say() {
          // super.say()   super 调用父类的方法
          return super.say() + '的儿子';
     }
}
var damao = new Son();
console.log(damao.say());
```

ES6 中的类和对象

三个注意点:

1. 在 ES6 中类没有变量提升，所以必须先定义类，才能通过类实例化对象.
2. 类里面的共有属性和方法一定要加this使用.
3. 类里面的this指向问题.
4. constructor 里面的this指向实例对象, 方法里面的this 指向这个方法的调用者

目录 Contents

4. 面向对象案例

面向对象版 tab 栏切换

功能需求:

1. 点击 tab栏,可以切换效果.
2. 点击 + 号, 可以添加 tab 项和内容项.
3. 点击 x 号, 可以删除当前的tab项和内容项.
4. 双击tab项文字或者内容项文字,可以修改里面的文字内容.

4. 面向对象案例

面向对象版 tab 栏切换

抽象对象：Tab 对象

1. 该对象具有切换功能
2. 该对象具有添加功能
3. 该对象具有删除功能
4. 该对象具有修改功能

4. 面向对象案例

面向对象版 tab 栏切换 添加功能

1. 点击 + 可以实现添加新的选项卡和内容
2. 第一步: 创建新的选项卡li 和 新的 内容 section
3. 第二步: 把创建的两个元素追加到对应的父元素中.
4. 以前的做法: 动态创建元素 createElement ，但是元素里面内容较多, 需要innerHTML赋值,在 appendChild 追加到父元素里面.
5. 现在高级做法: 利用 insertAdjacentHTML() 可以直接把字符串格式元素添加到父元素中
6. appendChild 不支持追加字符串的子元素, insertAdjacentHTML 支持追加字符串的元素
7. insertAdjacentHTML(追加的位置，‘要追加的字符串元素’)
8. 追加的位置有: **beforeend** 插入元素内部的最后一个子节点之后
9. 该方法地址: https://developer.mozilla.org/zh-CN/docs/Web/API/Element/insertAdjacentHTML

4. 面向对象案例

面向对象版 tab 栏切换 删除功能

1. 点击 × 可以删除当前的li选项卡和当前的section
2. X是没有索引号的, 但是它的父亲li 有索引号, 这个索引号正是我们想要的索引号
3. 所以核心思路是: 点击 x 号可以删除这个索引号对应的 li 和 section
4. 但是,当我们动态删除新的li和索引号时,也需要重新获取 x 这个元素. 需要调用init 方法

4. 面向对象案例

面向对象版 tab 栏切换 编辑功能

1. 双击选项卡li或者 section里面的文字,可以实现修改功能
2. 双击事件是: ondblclick
3. 如果双击文字,会默认选定文字,此时需要双击禁止选中文字
4. window.getSelection ? window.getSelection().removeAllRanges() : document.selection.empty();
5. 核心思路: 双击文字的时候, 在 里面生成一个文本框, 当失去焦点或者按下回车然后把文本框输入的值给原先元素即可.

传智播客旗下高端IT教育品牌