**一．事件冒泡：**

当一个元素接受到事件时，会把他接受到的所有事件传播给他的父级，一直到顶层window,这样的事件叫冒泡机制(事件冒泡) (事件源-->根节点,由内到外进行传播)

阻止事件冒泡：

IE10之前： ev.cancelBubble=true ;

IE10之后 : ev.stopPropagation();

**二．事件捕获**：

DOM标准事件流执行顺序是**：**先捕获再冒泡，即当触发dom事件时，会先进行事件捕获，捕获到事件源之后通过事件传播进行事件冒泡。

事件委托：事件委托也叫事件代理，就是事件冒泡和事件捕获的运用。

基本概念：一般来讲，会把一个或者一组元素的事件委托到它的父层或者更外层元素上，真正绑定事件的是外层元素，当事件响应到需要绑定的元素上时（事件捕获），会通过事件冒泡机制从而触发它的外层元素的绑定事件上，然后在外层元素上去执行函数

1. **面向对象：**

概念：是计算机的一种编程架构，由子程序作用的单个或者多个对象组合而成，包含属性和方法的对象是类的实例。

什么是对象：封装多个数据和方法的储存空间。

什么是自定义对象：封装现实中一个东西的属性和功能的存储空间。

1. **创建面向对象的几种方式**：
2. 字面量方式创建对象

缺点：只能创建单个对象，代码复用性太差

1. 工厂函数创建对象
2. 构造函数创建对象

原理：1.构造函数没有new Object();但后台会自动创建obj=new Object();

           2.this就相当于obj;

          3.构造函数不需要返回对象引用，直接new出即可。

构造函数的特点:

1.没有显示的创建对象

2.直接将属性和方法赋给了this对象

3.没有return 语句

构造函数规范：

1：构造函数也是函数，但函数名第一个大写；

       2：不用像普通函数那样调用，直接new出来就行了。

优点：解决了识别问题，即创建的每个实例对象都是CreatePerson这个对象的实例,

创建自定义函数意味着将来可以将它的实例标识为一种特定的类型，这是构造函数胜过工厂模式的地方

缺点：每个方法都要在每个实例上重新创建一遍

1. 构造函数加原型创建对象
2. **this指向问题**

this关键字：运行时，指代正在调用方法的对象

this本质是window下唯一的一个指针，指向当前正在调用方法的对象

如何在方法内访问当前对象自己的属性：this.属性名

在方法内访问当前对象自己的属性，必须使用this.属性名

This和定义在哪儿无关！仅和调用时使用的当前对象有关如果无主的调用或赋值，默

认this都是window!

**六．原型与原型链**：

原型 :

定义：去改写对象下面公用的方法或者属性，让公用的方法或者属性在内存中存在一份(提高性能)

理解：我们创建的每个函数都有一个prototype(原型)属性，这个属性是一个对象，它的用途是包含可以由特定类型的所有实例共享的属性和方法。逻辑上可以这么理解：prototype 通过调用构造函数而创建的那个对象的原型对象。使用原型的好处可以让所有对象实例共享它所包含的属性和方法。也就是说，不必在构造函数中定义对象信息，而是可以直接将这些信息添加到原型中。

优点：解决了消耗内存问题。

原型的属性：

prototype : 每个函数都有一个prototype(原型)属性, //prototype,构造函数的属性作用是指向当前构造函数的原型

constructor：原型的属性，作用是指向当前的构造函数

\_\_proto\_\_，实例化对象的属性，作用也是指向当前对象的原型

原型链：

　原型链：实例对象与原型之间的连接，叫做原型链

　　 \_proto\_(隐式连接)

  　　Object对象类型是原型链的最外层

　　 查找方式：实例对象->先查找自己本身下面的属性和方法->自身没找到会沿着原型链找到该对象的原型，再查看原型上是否有要查找的属性或方法->依次继续查找如果找到的话则返回，否则找到最顶层Object上还没有就真没有了

**七. 继承**

继承：

定义： 子类不影响父类，子类可以继承父类的一些功能 (代码复用)

1拷贝继承：

for in (将父类的原型拷贝给子类的原型)

2类式继承(借用构造函数继承):

使用构造函数的call(this,name,age) apply(this,[name,age]) 实现继承

3原型链继承:

将父类的实例作为子类的原型

4组合继承:

借用构造函数继承+ 原型链继承

5寄生式继承：

6寄生组合式继承

**八.函数的声明方式**

1 普通声明（命名函数，有名字）

2 匿名函数（拉姆达函数，没有名字）

调用： 1、自调用：

2、赋值给变量，通过变量调用

3 通过构造函数声明

**九.函数名的运用**

1. 单独的函数名可以当成是一个变量，该变量中存储着这个函数；
2. 通过函数名可以将一个函数通过参数的形式传入另一个函数；
3. 通过函数名还能将函数作为另一个函数的返回值返回。

**十.函数的内部属性**

1、this：执行函数的对象（该函数是哪个对象的方法，则this就是该对象）；

2、arguments：实参形成的集合

3、length：指定函数的形参个数；

4、prototype：指定函数的原型；

**十一.命名函数的函数声明提升**

命名函数在执行代码之前会先读取函数声明，我们可以在申明函数的前面调用

**十二.作用域：**

作用域就是变量与函数的可访问范围

1全局作用域：

1. 没有在任何函数内部的变量
2. 隐式声明的局部变量（变量提升）
3. 所有window对象的属性

2局部作用域：

函数内部申明的变量（函数作用域）

3作用域链：当多个函数发生嵌套的时候，系统从函数的最内部开始向外搜索，直到全局作用域。

4在函数内部使用var声明的变量，不论在何处声明，该变量都拥有整个函数的作用域。

**十三.闭包**

1概念：能够读取其他函数内部变量的函数 （函数嵌套函数）

2核心原理：被内部函数访问的外部函数的变量可以保存在外部函数作用域内而不是被回收

3作用： 可以读取函数内部的变量。

让一些变量的值长期驻扎(保存)在内存中

4垃圾回收：

1）标记清除：先给所有变量一个标记，去掉环境中的变量及被环境中变量引用的变量的标记，然后系统再销毁那些带有标记的变量。

2）引用计数：跟踪记录每个值被引用的次数，当声明一个变量并将一个引用类型的值赋给该变量时，这个值的引用次数就是1，如果这个值再被赋值给另一个变量，则引用次数加1。相反，如果一个变量脱离了该值的引用，则该值引用次数减1，当次数为0时，就会等待垃圾收集器的回收

