1. A

2.A

console.log( obj.options[obj.selectedIndex].text ) 输出的是1,

console.log( obj.options[obj.selectedIndex].value ) 输出的是a

3.C

文档类型

 DTD（文档类型定义）是一组机器可读的规则，他们定义 XML 或 HTML 的特定版本中允许有什么，不允许有什么。在解析网页时，浏览器将使用这些规则检查页面的有效性并且采取相应的措施。浏览器通过分析页面的 DOCTYPE 声明来了解要使用哪个 DTD ，由此知道要使用 HTML 的哪个版本。

 DOCTYPE 当前有两种风格，严格（ strict ）和过渡（ transitional ）。过渡 DOCTYPE 的目的是帮助开发人员从老版本迁移到新版本。

如果发送具有正确的 MIME 类型的 XHTML 文档，理解 XML 的浏览器将不显示无效的页面。

浏览器模式

浏览器有两种呈现模式：标准模式和混杂模式（quirks mode）。在标准模式中，浏览器根据规范呈现页面；在混杂模式中，页面以一种比较宽松的向后兼容的方式显示。

DOCTYPE 切换

对于 HTML 4.01 文档，

包含严格 DTD 的 DOCTYPE 常常导致页面以标准模式呈现。

包含过度 DTD 和 URI 的 DOCTYPE 也导致页面以标准模式呈现。

但是有过度 DTD 而没有 URI 会导致页面以混杂模式呈现。

DOCTYPE 不存在或形式不正确会导致 HTML 和 XHTML 文档以混杂模式呈现。

4.D

var a = [];

a[0]=1; a[1]=2; a[2]=3; a[5]=4;

则数组a为

Array[6]

0: 1

1: 2

2: 3

3:undefined

4:undefined

5: 4

length: 6

\_\_proto\_\_: Array[0]

5.D

--Math.round()执行标准舍入，即它总是将数值四舍五入为最接近的整数(这也是我们在数学课上学到的舍入规则)。

1. A  var a=b=3 时a是局部变量，而b是全局变量

7.D 首先span不是块级元素，是不支持宽高的，但是style中有了个float：left；就使得span变成了块级元素支持宽高，height:100%;即为，200，宽度由内容撑开。

但是内容中的 i 是绝对定位，脱离了文档流，所以不占父级空间，所以span的width=0

8.C

JS中this关键字， 它代表函数运行时，自动生成的一个内部对象，只能在函数内部使用。

 全局函数调用时，this指向全局函数； 应该是指向调用全局函数的对象。

关于JS中的this，详解请看博客：

http://www.cnblogs.com/kongxy/p/4581223.html

9.C

ECMAScript认为undefined是从null派生出来的，因此将它们定义为不严格意义上的相等，即undefined==null。

10.A

（1）Function是引用类型：保存在堆中，变量f1,f2是保存在栈中；

（2）闭包：一个函数（产生新的作用域）定义的局部变量、子函数的作用域在函数内， 但是一旦离开了这个函数，局部变量就无法访问，所有通过返回子函数到一个变量f1的方法，让f1指向堆中的函数作用域，这样可以使用局部变量i.

(3)   过程：

   第一次f1()  :f1=Foo()中，先执行Foo(): i = 0,return值返回给f1

 (f1指向子函数   f1()=function(){.....},因为子函数没有 定义i，所以向上找到父函数定义的 i:  )并执行子函数 输出i=0,再自加 i =1(覆盖了父函数Foo 的 i值);

  第二次f1() : 执行的是子函数 Function(){  ..},输出的是父函数 的 i=1,再自加 i =2;

  第一次f2():同第一次f1(),不同的是 f2指向堆中一个新的对象 function(){ ...},所有此i非彼i,输出i=0;如果再次f2(),那么和第二次f1(),一样输出i=1;

-------------------------------------------------------------------------------------------

二、

1.

双向绑定，**当数据发生变化的时候，用户界面发生相应的变化，开发者不需要手动的去修改dom**。

组件化开发：。组件可以扩展 HTML 元素，封装可重用的代码在较高层面上，组件是自定义元素，Vue.js 的编译器为它添加特殊功能。在有些情况下，组件也可以是原生 HTML 元素的形式，以 is 特性扩展。

1. <div v-for=”(item,index) in items”></div>

Vue.filter(“sum”,function(value){

return value+4

})

3.语法：call([thisObj[,arg1[, arg2[, [,.argN]]]]])

参数 thisObj 可选项。将被用作当前对象的对象。 arg1, arg2, , argN 可选项。将被传递方法参数序列。

说明   
call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。   
如果没有提供 thisObj 参数，那么 Global 对象被用作thisObj。说明白一点其实就是更改对象的内部指针，即改变对象的this指向的内容。这在面向对象的js编程过程中有时是很有用的。

**apply方法：**   
语法：apply(thisObj，[argArray])  
定义：应用某一对象的一个方法，用另一个对象替换当前对象。   
说明：   
如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。   
如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

用法基本一致，只是call和apply传入参数不同，apply需要传入一个数组来使用

4.什么是事件委托：通俗的讲，事件就是onclick，onmouseover，onmouseout，等就是事件，委托呢，就是让别人来做，这个事件本来是加在某些元素上的，然而你却加到别人身上来做，完成这个事件。也就是：利用冒泡的原理，把事件加到父级上，触发执行效果。提高性能。

5.利用<script>标签没有跨域限制的“漏洞”来达到与第三方通讯的目的。当需要通讯时，本站脚本创建一个<script>元素，地址指向第三方的API网址，形如：   
<script src="http://www.example.net/api?param1=1&param2=2"></script>   
并提供一个回调函数来接收数据（函数名可约定，或通过地址参数传递）。   
第三方产生的响应为json数据的包装（故称之为jsonp，即json padding），形如：   
callback({"name":"hax","gender":"Male"})   
这样浏览器会调用callback函数，并传递解析后json对象作为参数。本站脚本可在callback函数里处理所传入的数据。

6.

<!doctype html>

<html>

<head>

<title>二级联动</title>

<meta charset="utf-8">

</head>

<body>

<!--二级联动-->

<select id="province">

<option id='1'>山东省</option>

<option id='2'>辽宁省</option>

<option id='3'>吉林省</option>

</select>

<select id="city"> </select>

<script>

var provinceEle = document.getElementById("province");

provinceEle.onchange = function(){ var cityEle = document.getElementById("city");

cityEle.innerHTML = "";

// 每次进来先清空城市列表 var provinceValue = this.value;

/\* if(provinceValue == '山东省'){ cities = ['青岛市','济南市','威海市'] }

else if(provinceValue == '辽宁省'){ cities = ['沈阳市','大连市','锦州市'] }

else{ cities = ['吉林市','长春市','四平市'] }

\*/ switch (province){ case "山东省" cities = ['青岛市','济南市','威海市'] break;

case "辽宁省" cities = ['沈阳市','大连市','锦州市'] break;

case "山东省" cities = ['吉林市','长春市','四平市'] break; }

for(var i = 0;i<cities.length;i++){ var option = document.createElement('option');

//先创建option var textNode = document.createTextNode(cities[i]);

//再把城市名作为子节点填入，也可用innerHTML option.appendChild(textNode);

cityEle.appendChild(option); } }

</script>

</body>

</html>