JavaScript 练习

补充说明

2018 年 7 月 3 日更新：

第二题中，我们假设同学熟悉面向对象程序设计（OOP）中的迭代器（Iterator）模式。此题要求提供的是 Java 风格的迭代器接口。

has_next 方法返回布尔类型，表示是否还有下一个元素，相当于 C++ 风格迭代器中与 end() 的比较；next_value 方法返回下一个元素，相当于 C++ 风格迭代器中解引用、然后步进 1 个元素。值得注意的是 has_next 方法不应有副作用，意即：连续多次调用 has_next 应与只调用一次等价。有针对这一点的测试用例。

目录

第一题：IP 地址排序（30 分）
第二题：节点迭代器（30 分）
第三题：== 运算符（20 分）
第四题：闭包（20 分）

提交要求

前三题为编程题。请以网络学堂上提供的 index.js 文件为基础进行补充完善。我们提供自测脚本，内含少量测试用例，请同学们在提交前自测，避免因输入输出格式等问题造成失分。其用法为：执行命令 node test.js index.js，其中第三个参数为你将在网络学堂提交的 index.js 文件；看到三个 pass 字样表示通过。

前三题的评分标准是

- 功能完整（100%）
- 要求代码在功能上正确
- 按通过的测试用例数量给分
- 代码风格良好
- 建议使用 ESLint 规范你的代码格式，避免使用 var，== 等
- 若此项存在明显问题，在得分的基础上倒扣，但扣分不超过 5 分

第四题为文字简答题，请将答案写好，保存为 index.pdf 文件。篇幅不限，重在说清问题，长篇大论未必得分。

请在网络学堂提交两个文件：
index.pdf
index.js

第一题：IP 地址排序（30分）

实现一个函数 ip_sort，接受一个字符串数组为参数，对该数组中的所有 IPv4 地址进行升序排序，返回排序后的数组。

输入说明

保证传入一个 JavaScript 列表，可能具有任意长度，保证里面的地址均合法。样例输入：
`['255.255.255.0', '123.124.123.2', '59.66.137.30']`
此例的输出应为
`['59.66.137.30', '123.124.123.2', '255.255.255.0']`

第二题：迭代器（30分）

请实现一个 TreeIterator 类，完成二叉树的中序遍历，对外提供迭代器接口。

你的 TreeIterator 类的构造函数应接受一个参数，为可能取值 null 的 JavaScript 对象，表示二叉树的根节点。该值取 null 表示空树；非 null 则有 val、left 和 right 三个字段：val 的取值是一个整数，left 和 right 分别是左、右子树的根节点。

你的 TreeIterator 类应当实现 has_next 和 next_value 方法。

样例

```
const tree = {
  val: 1,
  right: {
    val: 2,
    right: null,
    left: {
      val: 3,
      right: null,
      left: null
    }
```

```
  },
  left: null
};
/*
以上数据结构表示的树如下图所示：
   1 <- 根节点
    \
     2
    /
   3
*/
```

评测程序对你的 TreeIterator 类的调用方法是

```
const iterator = new TreeIterator(tree);
while(iterator.has_next()){
    console.log(iterator.next_value());
}
// 应依次输出 1, 3, 2
```

提示：ES6 的生成器（Generator）特性、yield 关键词可能对你有所帮助，可考虑使用。

第三题：equal（20分）

请了解 JavaScript 中的 == 操作符。

请在不允许使用 == 或等价物的情况下，实现一个相同功能的 equal 函数。参见 ES6 标准中关于 == 操作符的介绍：https://www.ecma-international.org/ecma-262/6.0/#sec-7.2.12

请在以后的编程中放弃使用 == 操作符。

输入说明

保证传入两个参数，传入的参数可能是任何合法的 JavaScript 对象。样例：

```
equal(2, 3); // 应输出 false
equal(2, "2"); // true
equal(null, undefined); // true
equal(0, ""); // true
equal({}, {}); // false
equal(null, ""); // false
```

判题说明

评测时将使用助教组定制的 JavaScript 解释器 fake-node，它和原版的 node 的区别就是：前者对于 == 操作符会有奇怪的行为。如果你使用了 == 操作符或其等价物（包括但不限于 != 操作符、eval 函数），那么几乎可以肯定你的程序无法通过评测。

```
$ fake-node
> 1 == 1
true
> 1 == '1'
false
> 1 != {}
false
> eval("'海巨" == 666')
true
```

第四题：闭包（20分）

阐述清楚什么是作用域链，什么是闭包。有些人认为近期相关概念的重要性在下降，请问这是为什么？（提示：与ES6的新特性有关）你如何看待这个观点？（提示：你可以拒绝这个观点）

以下这两段代码的结果分别是什么？为什么？

片段一：

```
for (var i = 0; i < 5; ++i) {
    setTimeout(function() {
        console.log(i + ' ');
    }, 100);
}
```

片段二：

```
function add() {
    let x = 1;
    return function() {
        console.log(++x);
    };
}
const num = add();
num();
num();
num();
```