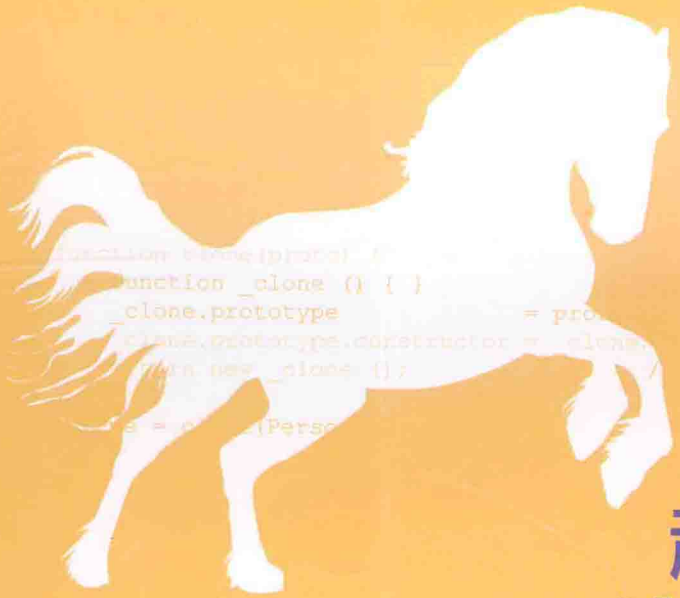
Broadview
www.broadview.com.cn

·代码逆袭·

始于功能，终于代码，拿来即用
语法、规范、理念、工具、技巧，尽在代码中



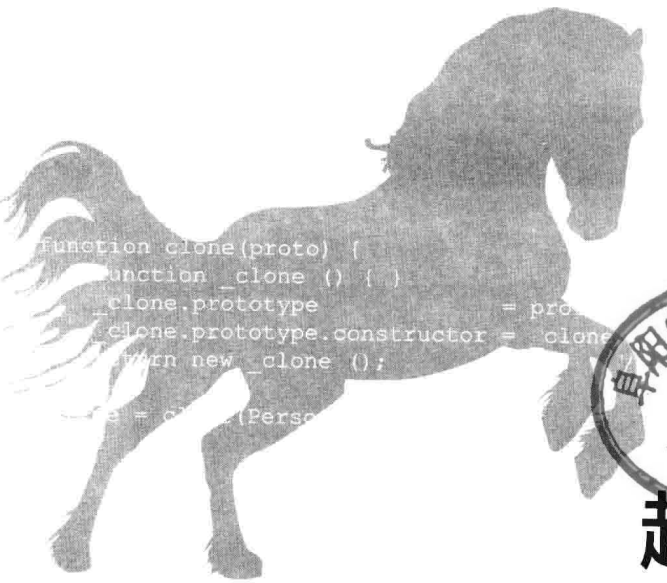
超实用的 JavaScript代码段

席新亮 编著

基于实用、实践、学以致用的原则讲解的全是干货，不讲废话，
实例几乎涵盖了所有的JavaScript代码模块，闭包、冒泡捕获、
DOM与BOM、JavaScript Function、跨设备兼容性……

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

·代码逆袭·



超实用的 JavaScript代码段

席新亮 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

基于实用、实践、前瞻性、学习的原则，笔者精选了近 300 段 JavaScript 代码和 400 种解决方案，覆盖了几乎所有的脚本处理模块，最大程度地帮助读者学习、实践 JavaScript 的各个方面，让读者成为一个有代码实践、有思想品质、有技术深度的 JavaScript 高手。

全书分为 9 章，包括 JavaScript 的一些必备知识，常用的表单处理、图片处理、内容展示、页面处理、日期处理、页面特效、移动开发等代码及其他常用代码，涉及闭包、Ajax、Data URI、DOM、BOM、数据字典、HTML 5、XML、JSON 等现代 JavaScript 开发常用的技术。对那些想迅速全面了解 JavaScript 代码处理技巧的前端开发人员有重要的指导意义。

本书内容简洁明了、代码精练、重点突出、实例丰富，语言通俗易懂，原理清晰明白，是广大前端入门者的最优选择。

图书在版编目（CIP）数据

超实用的 JavaScript 代码段 / 席新亮编著. —北京：电子工业出版社，2014.9
（代码逆袭）
ISBN 978-7-121-23970-0

Ⅰ. ①超… Ⅱ. ①席… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2014）第 177677 号

责任编辑：董　英
印　　刷：北京京科印刷有限公司
装　　订：北京京科印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：21　　字数：511 千字
版　　次：2014 年 9 月第 1 版
印　　次：2014 年 9 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件到 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

最近几年，HTML 5、Node.js 等新生技术给 JavaScript 带来了质的飞跃。很多对前端感兴趣或刚入门不久的朋友都在问笔者同一个问题，如何才能学好 JavaScript？笔者给出的答案是实践、实践、再实践。本书的几百段代码，全是笔者的原创，非常具有借鉴、学习、实践价值。学习编程就是学习一种思考方式，是学习看透事物本质的一种锻炼，只有不断学习代码才能知道怎样写出更好的代码。

JavaScript 中的那些槛儿

有些前端开发人员做了多年 JavaScript 开发，依然在面对问题时束手无策。这些问题很常见，你又能了解多少呢？

- 如何实现 JavaScript 的闭包？
- 为什么事件有冒泡捕获？
- 浏览器之间有何差异？
- 何为 DOM，何为 BOM，两者之间有何区别？
- JavaScript 里面的对象和 Java 的对象一样吗？
- XHTML 与 HTML 的区别是什么？
- JavaScript Function 的原理是什么？
- 浏览器都有哪些 JavaScript 引擎？
- 是否认识！、||、三元操作符、单链式写法、声明变量的缩略写法、惰性载入？
- 如何判断浏览器是否支持 HTML 5？
- 如何开发出跨设备、兼容性的代码？
- 如何在不同浏览器中调试代码？

以上所有内容在本书的代码中都有讲解，除这些常见的 JavaScript 门槛外，目前页面中常见的一些需求，在本书中都能够找到与之匹配的代码实例或解决方案。

如何学习 JavaScript

请记住以下 3 个词，只要您坚持下去，写 JavaScript 代码就是这么简单。

- **借鉴**：多阅读、多分析、多学习一些成功的 JavaScript 代码，多逛一些 JavaScript 类的技术社区、论坛、博客等，多与一些技术牛人交流学习，借鉴其经验、思路等。
- **思考**：思考的深度决定了代码质量，因此多思考、多问几个为什么会在很大程度上提高代码的质量。

- **实践**：实践是检验一切的真理，多练习代码、多做项目，在实践中提高是至今为止最有效的学习途径。

伟大产生于平凡，只要持之以恒地坚持以上几个原则，JavaScript 学习定能一马平川。

本书的编写特点

1. 独特切入，覆盖全面

与市面上其他 JavaScript 书不同，本书的代码实例几乎涵盖了所有的 JavaScript 代码模块，基于实用、实践、学以致用的原则，本书讲解的全是干货，不讲废话。

2. 应用场景与代码案例结合

本书每个代码段的开篇，都以应用场景为铺垫，让读者明白为什么要学习这段代码，知道什么情况下需要参考这个例子，学了这段代码之后还能干什么。

3. 去中心化，分布式学习

本书的代码实例都是独立的，你可以从中间开始学，也可以从头开始学。

4. 各个击破，综合布局

除了每段代码都有对应的单独运行页面外，笔者还将每章所有的单独代码进行了整合，为那些想晋级的读者，提供探索、学习的机会——如何将更多的代码进行整合、如何将每段代码进行复用、如何代码重构，提高读者对代码的整体把控能力。

5. 兼容，优化

本书针对每个需求都会多提供几种解决方案，并通过分析来总结出最实用的优化方案和兼容方案，读者越早动手编码，就能学得越快。

本书的代码设计始于需求、终于代码，是前端开发人员的案头必备。

本书的内容安排

本书共 9 章，各章节为不同的 JavaScript 技术点和应用场景。

第 1 章为 JavaScript 必须知道的事儿，介绍如何调用 JavaScript、使用什么工具开发 JavaScript、如何调试 JavaScript 等。

第 2 章为表单常用代码，包括关闭输入法、禁止输入、限制字符长度、Checkbox 的全选、反选、下拉框二级联动、回车提交表单、常见的验证规则、文件上传等。

第 3 章为图片控制常用代码，首先介绍如何开发动画模块管理所有的动画业务逻辑，然后介绍常用的代码，包括实时预览上传图片、图片放大镜效果、水中倒影效果、横向图片轮播、图片层叠轮播、类似 QQ 相册效果等。

第 4 章为内容显示常用代码，包括表格光棒效果、表格内容拖曳、英文字符串自动换行、调整字体大小、关键字高亮、打字机效果、字幕上下滚动、弹出层、层模拟提示消息框、遮罩层、对联浮动广告、QQ 消息窗口提示等。

第 5 章为页面控制常用代码，包括打开新页面、打开子模式窗口、子窗口全屏、禁止滚动条、屏蔽右键、禁止查看源代码、添加到收藏夹、网页另存为、判断上一页来源、检

测某个网站的链接速度、脚本永不出错等。

第 6 章为日期处理常用代码，包括获取日期的指定部分、显示当前时间、获取短日期格式、日期比较大小、对指定日期进行加减、判断是闰年还是平年、日期的合法性验证、日期格式化等。

第 7 章为页面特效常用代码，包括悬浮导航、滑动门导航、树形菜单导航、仿 QQ 菜单页面、五颜六色的雪花等。

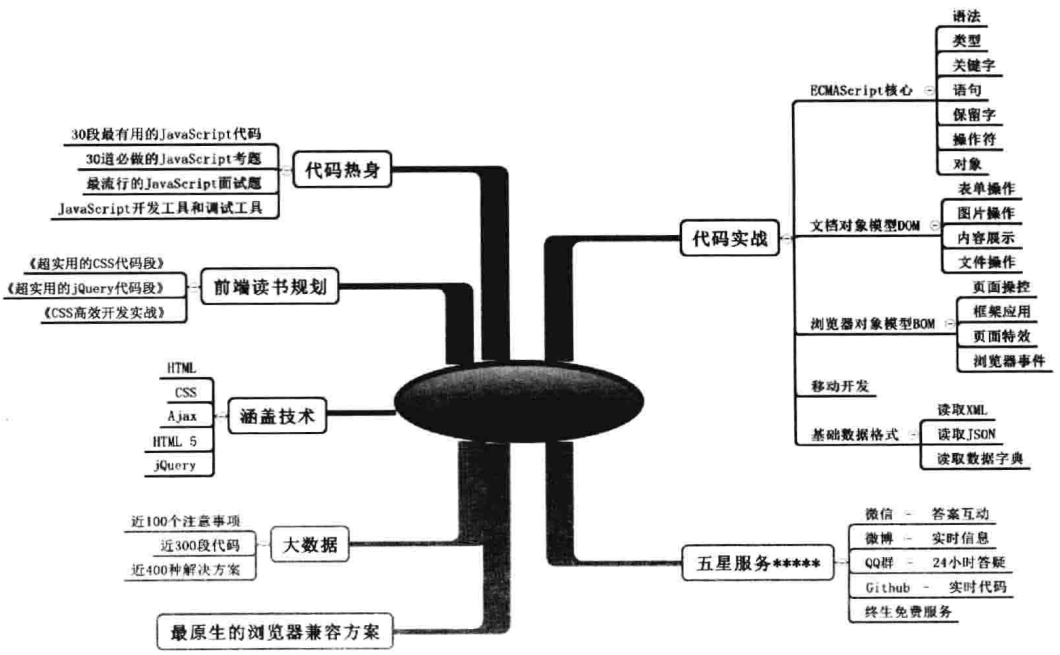
第 8 章为移动开发常用代码，包括如何区分不同平台类型、检测平台方向的变更、防止网页触摸滚动等。

第 9 章为其他常用代码，包括 JavaScript 调用百度地图、实现跨浏览器 Ajax、获取当前地理坐标、等级星投票、HTML 5 版 MP3 播放器等。

本书面对的读者

- JavaScript 开发入门者
- 前端开发入门者
- 网页美工人员
- 中小型企业网站开发者
- 大中专院校的学生
- 各种 IT 培训学校的学生
- 网站后台开发人员（PHP、JSP、Java、ASP.NET 开发人员）
- 网站建设与网页设计的相关威客兼职人员

本书的思维导图



编者推荐

会 JavaScript 的人很多，但精通 JavaScript 的人很少！本书提供了近 300 段代码和 400 种解决方案，面对网页开发的千变万化，JavaScript 解决方案也层出不穷。本书的目的就是将最有用的代码与读者分享，包含了网页开发人员在实战中必须具备的所有技巧和方法，读者可以拿来就用。市场上没有一本 JavaScript 书能与时俱进，本书填补了这一空白。本书将 JavaScript 的函数设计、原型扩展、模块管理、兼容优化、闭包、Ajax、JSON、面向对象等难点都融入一个个的小例子中，是一本最值得拥有的 JavaScript 代码实战书。

本书的服务

笔者能力有限，如果书中有什么疏漏，或者对内容有什么疑问，可通过以下方式与我们沟通。

- QQ 群：296811675，作者在线答疑。
- 扫描封底的微信二维码，时刻参与我们的图书互动和本书的考题答案。
- @博文视点 Broadview 的微博，了解我们发布的信息和各种前端流行技术。
- 博文视点官方网站 http://www.broadview.com.cn/，下载本书所有实例源代码。
- Github，https://github.com/maomaoshu/jsBook，了解代码的实时更新和迭代过程，可以在每章代码下参与讨论，也可以观看其他读者提出的问题，还可以随时随地下载代码。

很多读者在学习过程中苦于无法交流，小故障无法及时解决，加入我们的服务方阵，我们将为您提供终身免费的服务。一本书、一段情、一辈子。

本书主要由席新亮编写，参与编写的人员还有赵荣娇、任建智、李勇、周敏、王铁民、张兴瑜、马新原、薛淑英、殷龙、谢郁、于健、周洋、翟世伟、李兰英。

序 1　30 段简单趣味的 JavaScript 代码

过去，在大多数编程人员眼中，JavaScript 属于二等公民，它的安全性经常受到编程人员的质疑。但峰回路转，在后来的浏览器争夺战中，JavaScript 成了主要的争夺战场，导致 JavaScript 编程有了质的飞越。一方面，它不但为各大浏览器立下了汗马功劳，并且有力地支撑了 HTML 5 这一战略性语言的市场推广；另一方面，Node.js 的诞生，让 JavaScript 在服务器端编程中，也掀起了一场革命。

学习 JavaScript 的目标

如今，JavaScript 的强大性已经毋庸置疑，不论是在前端编写效果、处理 DOM，还是处理服务端的业务（服务端的 JavaScript 超出了本书的范围，属于 Node），它都游刃有余。

JavaScript 这么酷，学习它的目标是什么呢？大道至简，目标就是针对 Web 项目需求，迅速找到合理的解决方案。

30 段 JavaScript 代码

1. 前端人员经常遇到的问题就是如何区分 IE 及非 IE 浏览器，JavaScript 代码是：

```
if(!+[1,]){                    //IE 11 不支持
alert("这是 IE 浏览器");
}else{
    alert("这不是 IE 浏览器");
}
```

2. 将日期直接转换为数值：

```
+new Date();                   //结果 1396338783628
```

3. 非 IE 浏览器下将类数组对象“arguments”转为数组：

```
Array.prototype.slice.call(arguments);
```

arguments 不是 Array 的实例，因此不是真正的数组，也就没有 slice()，那为什么使用“Array.prototype.slice”而不是“Array().slice”或“[].slice”呢？因为这两种方法效率比较

低，故使用代码中的写法访问 Array 的内置函数。

4. 最简单的选择运算符||：

```
var a = 0 || 3;
console.log(a);                    //结果 3
```

如果=后面的第 1 个值计算结果为布尔值“真”，则 a 的值取第 1 个，否则取第 2 个。

5. 单链式运算（如：a++ - 1）：

```
var a=10;
console.log(a++ - 1);
```

先执行“a - 1”，再执行“a = a + 1”。

6. 有趣的 void 操作符：

```
<a href="javascript:void(0)">我是一个死链接</a>
```

void 是一种操作符，用来计算一个表达式但不返回值。用法：javascript:void (expression)，expression 是一个要计算的 JavaScript 标准表达式。

7. 跳转至新页面，并且保证浏览器不会再回退：

```
location.replace("http://www.niaowo.mobile");        //跳转至互联网招聘网站
```

location 的 replace()方法可以用一个新的文档替换当前文档，并且该方法还会覆盖 History 对象中的记录。

8. 几秒钟之后返回上一页：

```
<meta http-equiv="refresh" content="3; url=javascript:window.history.go(-1);">
```

其中 content 为设置的时间。

9. 在打开的子窗口中刷新父窗口：

```
window.opener.location.reload() ;
```

10. 验证是否为负数的正则表达式：

```
/^-\d+$/.test(str);
```

11. 用 JavaScript 打印页面：

```
window.print()
```

window.print()属于浏览器内置的 API，可以直接打印页面。

12. 显示/隐藏一个 DOM 元素：

```
el.style.display = "";
el.style.display = "none";                    //el 是待操作的 DOM 元素
```

DOM 元素的显示/隐藏主要是通过设置元素的样式 display 属性来实现。

13. 实现 alert()中的文本换行：

```
alert("p\np");
```

“\n”代表换行符。

14. 实现 ECMAScript 5 中的 Object.create()函数：

```
function clone(proto) {
    function _clone () { }
    _clone.prototype                = proto;
    _clone.prototype.constructor = _clone;
    return new _clone ();                    //等价于 Object.create(Person);
}
```

```
var me = clone(Person);
```

用原型链形式继承，构造函数重新指向新创建的对象。

15. 理解 JavaScript 中的闭包：

例如，以下代码会输出 5 次，结果都是 5，如何输出 0、1、2、3、4？

```
for(var i = 0; i < 5; i++) {
    setTimeout(function() {
        console.log(i);
    }, 1000);
}
```

利用闭包的原理实现，代码如下：

```
for(var i = 0; i < 5; i++) {
    (function(e) {
        setTimeout(function() {
            console.log(e);
        }, 1000);
    })(i);
}
```

16. 检测 Shift、Alt、Ctrl 键：

```
//以下是浏览器内置的检测方法
event.shiftKey;          //检测 Shift
event.altKey;            //检测 Alt
event.ctrlKey;           //检测 Ctrl
```

17. 获取屏幕分辨率的宽、高：

```
window.screen.height;    //获取屏幕的高
window.screen.width;     //获取屏幕的宽
```

window.screen 这个对象包含了有关用户屏幕的信息。

18. 脚本永不出错的方式：

```
window.onerror = function(m, f, l){
    return true;
};
```

19. 让 JavaScript 处理字符与 ASCII 码之间的转换：

```
console.log("a".charCodeAt(0));          //97
console.log(String.fromCharCode(75));    //K
```

charCodeAt()返回指定位置字符的 Unicode 编码；fromCharCode()接收一个指定的 Unicode 值，然后返回一个字符串。

20. 访问对象属性的代码：

```
var demo = {name: 'ki'}
demo.name;               //ki
demo['name'];            //ki
```

访问对象属性一般存在两种方式，通过“.”或“[]”。一般情况下两种方式等效，但是“[]”还可以动态设置属性，如：

```
var get = 'name';
```

```
demo [get];                    //ki
```

21. 把一个值转换为布尔型的最简单方式：

```
!!'demo';          //true
!!'';              //false
!!'0';             //true
!!'1';             //true
!!{};              //true
!!true;            //true
```

使用“!”操作符两次，可以把一个值转换为布尔型。

22. 判断浏览器是否支持 HTML 5：

```
!!navigator.geolocation;
```

在 HTML 5 中，navigator.geolocation 可以获取设备的当前位置，通过双“!”就可以判断是否支持此 API，即是否支持 HTML 5。

23. 判断浏览器是否支持 Canvas：

```
function isCanvas () {
    return !!document.createElement('canvas').getContext;
}
```

24. 判断 IE 版本：

```
window.navigator.appVersion
```

上述代码返回一个字符串，表示所使用浏览器的版本号。它可能只包含一个数字，比如 5.0，还可能包含一些其他的相关信息。

25. 声明变量的缩略写法与复杂写法：

```
/*复杂写法*/
var x;
var y;
var z = 3;

/*缩略写法*/
var x, y, z = 3;
```

JavaScript 是比较灵活的语言，编程时尽量采用缩略写法，这样会提高 JavaScript 的性能。

26. 采取惰性载入的方案提高函数代码的性能：

```
function addEvents (type, element, fun) {
    if (element.addEventListener) {
        element.addEventListener(type, fun, false);
    }
    else if(element.attachEvent){
        element.attachEvent('on' + type, fun);
    }
    else{
        element['on' + type] = fun;
    }
}
```

所谓惰性载入就是在第一次执行代码后，用函数代码内部的方法覆盖原有代码，代码如下：

```
var addEvents = (function () {
    if (document.addEventListener) {
        return function (type, element, fun) {
            element.addEventListener(type, fun, false);
        }
    }
    else if (document.attachEvent) {
        return function (type, element, fun) {
            element.attachEvent('on' + type, fun);
        }
    }
    else {
        return function (type, element, fun) {
            element['on' + type] = fun;
        }
    }
})();
```

27. 捕捉 Ctrl+Enter 按键：

```
if (event.ctrlKey && event.keyCode == 13)
{
  console.log("You pressed the Ctrl + Enter")
}
```

event.ctrlKey 检测 Ctrl 键，event.keyCode == 13 检测 Enter 键。

28. 获取浏览器插件的数目：

```
navigator.plugins.length;
```

navigator 用来检测浏览器的版本、所支持的 MIME 类型、已安装的外挂程序（plugin）。

29. 实现对 Windows、Mac、Linux、UNIX 操作系统的判断：

```
var osType = "",
    windows = (navigator.userAgent.indexOf("Windows",0) != -1)?1:0,
    mac = (navigator.userAgent.toLowerCase().indexOf("mac",0) != -1)?1:0,
    linux = (navigator.userAgent.indexOf("Linux",0) != -1)?1:0,
    unix = (navigator.userAgent.indexOf("X11",0) != -1)?1:0;

    if (windows) osType = "Windows";
    else if (mac) osType = "Mac";
    else if (linux) osType = "Lunix";
    else if (unix) osType = "Unix";
    console.log(osType);
```

navigator.userAgent 表示用户代理。

30. 使用原生 JavaScript 判断是否是移动设备浏览器：

```
var mobileReg = /iphone|ipod|android.*mobile|windows.*phone|blackberry.*mobile/i;
```

```
if((mobileReg.test(window.navigator.userAgent.toLowerCase()))){
    alert("移动设备！");
}else{
    alert("非移动设备！");
}
```

对于以上 30 段代码，如果你都可以非常容易地找到解决方案，那么本书可能不太适合你。“且行且珍惜”是网络最近比较流行的词语，编程生涯亦如此。

序 2　30 个你不可能全部会做的 JavaScript 题目

你真的已经非常精通 JavaScript 了吗？以下几道题针对 JavaScript 这门语言进行简单的测试，浏览器的环境是 ECMA262 参考标准。

1. 以下表达式的运行结果是：

```
["1", "2", "3"].map(parseInt)
```

A. [“1”, “2”, “3”]

B. [1, 2, 3]

C. [0, 1, 2]

D. 其他

2. 以下表达式的运行结果是：

```
[typeof null, null instanceof Object]
```

A. ["object", false]

B. [null, false]

C. ["object", true]

D. 其他

3. 以下表达式的运行结果是：

```
[ [3,2,1].reduce(Math.pow), [].reduce(Math.pow) ]
```

A. 报错

B. [9, 0]

C. [9, NaN]

D. [9, undefined]

4. 以下表达式的运行结果是：

```
var val = 'smtg';
console.log('Value is ' + (val === 'smtg') ? 'Something' : 'Nothing');
```

A. Something

B. Nothing

C. NaN

D. 其他

5. 以下表达式的运行结果是：

```
var name = 'World!';
(function () {
    if (typeof name === 'undefined') {
        var name = 'Jack';
        console.log('Goodbye ' + name);
    } else {
        console.log('Hello ' + name);
    }
})();
```

A. Goodbye Jack

B. Hello Jack

C. Goodbye undefined

D. Hello undefined

6. 以下表达式的运行结果是：

```
var END = Math.pow(2, 53);
var START = END - 100;
var count = 0;
for (var i = START; i <= END; i++) {
    count++;
}
console.log(count);
```

A. 0

B. 100

C. 101

D. 其他

7. 以下表达式的运行结果是：

```
var ary = [0,1,2];
ary[10] = 10;
ary.filter(function(x) { return x === undefined;});
```

A. [undefined × 7]

B. [0, 1, 2, 10]

C. []

D. [undefined]

8. 以下表达式的运行结果是：

```
var two   = 0.2
var one   = 0.1
var eight = 0.8
var six   = 0.6
[two - one == one, eight - six == two]
```

A. [true, true]

B. [false, false]

C. [true, false]

D. 其他

9. 以下表达式的运行结果是：

```
function showCase(value) {
    switch(value) {
    case 'A':
        console.log('Case A');
        break;
    case 'B':
        console.log('Case B');
        break;
    case undefined:
        console.log('undefined');
        break;
    default:
        console.log('Do not know!');
    }
}
showCase(new String('A'));
```

A. Case A

B. Case B

C. Do not know!

D. undefined

10. 以下表达式的运行结果是：

```
function showCase2(value) {
    switch(value) {
    case 'A':
        console.log('Case A');
        break;
    case 'B':
        console.log('Case B');
        break;
    case undefined:
        console.log('undefined');
        break;
    default:
        console.log('Do not know!');
    }
}
showCase2(String('A'));
```

A. Case A

B. Case B

C. Do not know!

D. undefined

11. 以下表达式的运行结果是：

```
function isOdd(num) {
    return num % 2 == 1;
}
function isEven(num) {
    return num % 2 == 0;
}
function isSane(num) {
    return isEven(num) || isOdd(num);
}
var values = [7, 4, '13', -9, Infinity];
values.map(isSane);
```

A. [true, true, true, true, true]

B. [true, true, true, true, false]

C. [true, true, true, false, false]

D. [true, true, false, false, false]

12. 以下表达式的运行结果是：

```
parseInt(3, 8)
parseInt(3, 2)
parseInt(3, 0)
```

A. 3, 3, 3

B. 3, 3, NaN

C. 3, NaN, NaN

D. 其他

13. 以下表达式的运行结果是：

```
Array.isArray( Array.prototype )
```

A. true

B. false

C. 报错

D. 其他

14. 以下表达式的运行结果是：

```
var a = [0];
if ([0]) {
  console.log(a == true);
} else {
  console.log("wut");
}
```

A. true

B. false

C. ”wut”