珠峰培训就业班-第一次月考答案

> 01~10题，每题2分
>
> 11~20题，每题3分
>
> 21~30题，每题5分

1. JS中的数据类型都有哪些？它们之间有什么区别？该如何检测？

> 基本数据类型：number、string、boolean、null、undefined
>
> 引用数据类型：
>
> - object（普通对象、数组对象、正则对象、日期对象、Math、实例、prototype等）
> - function（普通函数和类）
>
> 特殊数据类型：Symbol
>
> --
>
> 基本数据类型按值操作，引用数据类型按照堆内存的引用地址来操作
>
> --
>
> 数据类型检测四种方式：
>
> - typeof
> - instanceof
> - constructor
> - Object.prototype.toString.call()

2. 常用浏览器的内核都有哪些？

> webkit、Gecko、Trident、Presto等

3. 数组中常用的迭代方法有哪些？都是什么意思？（至少四种）

> forEach、map、find、some、filter、reduce、every、sort等

4. 说一下你对闭包的理解，以及工作中什么地方你用到了闭包？

> 函数执行会形成一个私有的作用域（私有栈内存），这就是闭包，在真实项目中主要应用闭包的两大作用
>
> - 保护
> - 保存

> 之前研究过像JQUERY等类库的源码，为了防止全局变量污染，基本上所有代码都是放到闭包中保护起来的
>
> 项目中遇到循环事件绑定，也会基于闭包保存的作用，去存储对应的索引
>
> 之前研究过JS高阶编程技巧：柯理化函数编程思想，这个就是闭包的应用，重写过debounce、throttle函数节流和防抖的方法
>
> ……
>
> 因为闭包会产生不释放的栈内存，所以尽可能在项目中少使用

5. 阐述一下let/var/const三者之间的区别？

> let 和 var 的区别
>
> - 不存在变量提升
> - 不允许重复声明
> - 在全局作用域下设置变量不会给window设置属性
> - 存在块级作用域
> - 解决了一些暂时性死区问题
>
> let 和 const 的区别
>
> - const 创建的是常量，存储的值不能被修改（准确说是不能修改变量的指向）

6. 阐述一下call/apply/bind三者之间的区别，以及应用场景？

> call 和 apply 的区别
>
> - 给方法传参的时候，call是按顺序，一个个传递；apply是把传递的参数放到一个数组中传递；
> - 在传递参数较多的情况下， call的性能要高于apply；
>
> call 和 bind 的区别
>
> - call在改变函数this指向的时候，会把函数立即执行，而bind不会把函数立即执行，只是预先处理了this和实参信息；
>
> 真实项目中，我们需要改变this指向的时候，会应用这三个方法，例如：
>
> - 给元素进行事件绑定，我们需要把事件触发，所执行的函数中this和参数进行预先处理，此时可以使用bind进行处理；
> - 我们可以基于call方法，让类数组借用数组原型上的方法，例如：把类数组转换为数组
> - 可以基于apply传参是一个数组，借用Math.max获取数组中的最大值
> - ……

7. 怎么让一个 div 水平垂直居中？（不少于三种解决方案）

```css
/* 已知宽高 */
.box {
    position: absolute;
    top: 50%;
    left: 50%;
    margin-top: -50px;
    margin-left: -50px;
    width: 100px;
    height: 100px;
}
```

```css
/* 未知宽高 */
.box {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
----------
.box {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    margin: auto;
}
----------
.container {
    display: flex;
    justify-content: center;
    align-items: center;
}
.container .box {}
```

8. 写出下面代码的输出结果

> "NaN北京珠峰培训nulltrue[object Object]"

9. 写出下面代码的输出结果

> "number"

10. 写出下面代码输出结果

> undefined 20

> 13 13
>
> undefined 13

11. 写出下面代码输出的结果

> 9 10 10 1

12. 写出下面代码输出结果

> 11 6
> 13
> 10 6

13. 写出下面代码输出结果

> {0: 10, 1: 20, length: 2, push: ƒ}

14. 写出下面代码输出结果

> 13
>
> 234
>
> 95 234

15. 写出下面代码的运行结果

> "培训"
> "珠峰"
> "培训"

16. 写出下面代码的运行结果

> false true Object 10 21 30

17. 写出下面代码的运行结果

> 10 "A"
>
> NaN undefined
>
> NaN undefined

18. 写出下面代码的运行结果

> "1" 、 "1" 、 "2" 、 "1" 、 "3"、“报错”

19. 写出下面代码的运行结果

> "WINDOW" 、“WINDOW” 、 “OBJ” 、 “WINDOW”

20. 写出下面代码的运行结果

> "undefinedJerryTom" 或者“写报错”

21. 下面代码能否实现预期效果？如不能请写出正确答案！

```
/*方案一*/
for(let i=0;i<btnList.length;i++){
    btnList[i].onclick=function(){
        box.style.background=colorAry[i];
    }
}
/*方案二*/
for(var i=0;i<btnList.length;i++){
    btnList[i].onclick=(function(i){
        return function(){
            box.style.background=colorAry[i];
        }
    })(i)
}
/*方案三*/
for(var i=0;i<btnList.length;i++){
    btnList[i].index=i;
    btnList[i].onclick=function(){
        box.style.background=colorAry[this.index];
    }
}
/*方案四*/
let $box=$('#box'),
    $btnList=$('button'),
    colorAry=['red','green','blue'];
$btnList.click(function(){
    let index=$(this).index();
    $box.css('background',colorAry[index]);
});
```

22. 完成如下需求

```
~function(){
    function flatten(){
        /*第一种*/
        return JSON.stringify(this).replace(/(\
[|\])/g,'').split(',').map(item=>{
            return Number(item);
        });
        /*第二种*/
        return this.flat(Infinity);
```

```
        /*第三种*/
        return this.toString().split(',').map(item=>{
            return Number(item);
        });
        /*第四种*/
        let arr=this.slice(0);
        while (arr.some(item => Array.isArray(item))) {
            arr = [].concat(...arr);
        }
        return arr;
        /*第五种*/
        let result = [];
        let fn = function (ary) {
            for (let i = 0; i < ary.length; i++) {
                let item = ary[i];
                if (Array.isArray(ary[i])) {
                    fn(item);
                } else {
                    result.push(item);
                }
            }
        }
        fn(this);
        return result;
    }

    function unique(){
        /*第一种*/
        return Array.from(new Set(this));
        /*第二种*/
        return [...new Set(this)];
        /*第三种*/
        let obj={};
        for(let i=0;i<this.length;i++){
            let item=this[i];
            if(typeof obj[item]!=='undefined'){
                this[i]=this[this.length-1];
                this.length--;
                i--;
                continue;
            }
            obj[item]=item;
        }
        return this;
    }
```

```
    Array.prototype.flatten=flatten;
    Array.prototype.unique=unique;
}();
let arr = [[1, 2, 2], [3, 4, 5, 5], [6, 7, 8, 9,[11, 12, [12, 13, [14]]]], 10];
ary.flatten().unique().sort((a,b)=>a-b); //=>[1, 2, 3, 4, 5, 6, 7, 8, 9....]
```

23. 完成如下需求

```
function _new(Fn, ...arg) {
    let obj = {};
    obj.__proto__ = Fn.prototype;
    Fn.call(obj, ...arg);
    return obj;
}
```

24. 完成如下需求

```
function $attr(property, value) {
    let elements = document.getElementsByTagName('*'),
        arr = [];
    elements = Array.from(elements);
    elements.forEach(item => {
        let itemValue = item.getAttribute(property);
        if (property==='class') {
            new RegExp("\\b" + value + "\\b").test(itemValue)?arr.push(item):null;
            return;
        }
        if (itemValue === value) {
            arr.push(item);
        }
    });
    return arr;
}
console.log($attr('class', 'box'));
```

25. 完成如下需求

```
~function(){
    function queryRandomName(){
        let time=new Date().getTime();
        return '$zhufeng'+time;
    }
```

```
    function change(context=window,...args){
        let _this=this,
            result=null,
            ran=queryRandomName();
        context[ran]=_this;
        result=context[ran](...args);
        delete context[ran];
        return result;
    };
    Function.prototype.change=change;
}();
```

26. 完成如下需求

```
~function(){
    //=>bind方法在IE6~8中不兼容，接下来我们自己基于原生JS实现这个方法
    function bind(context){
        context=context||window;
        var _this = this,
            outerArg=[].slice.call(arguments,1);
        return function anonymous() {
            var innerArg=[].slice.call(arguments,0);
            _this.apply(context, outerArg.concat(innerArg));
        }
    };
    Function.prototype.bind=bind;
}();
```

27. 完成如下需求

```
<script src="js/jquery.min.js"></script>
<script>
//=>函数节流
function throttle(func, wait) {
    let timer = null,
        result = null,
        previous = 0;
    return function anonymous(...args) {
        let context = this,
            now = new Date,
            spanTime = wait - (now - previous);
        if (spanTime <= 0) {
            result = func.call(context, ...args);
            clearTimeout(timer);
            timer = null;
```

```
            previous = now;
        } else if (!timer) {
            timer = setTimeout(() => {
                result = func.call(context, ...args);
                timer = null;
                previous = new Date;
            }, spanTime);
        }
        return result;
    }
}
//=>业务模块
let imgsModule=(function(){
    let $imgBoxs=$('.imgBox'),
        $window=$(window);
    //=>图片延迟加载
    let lazyImgs=function(){
        let B = $window.outerHeight() + $window.scrollTop();
        $imgBoxs.filter("[isLoad!='TRUE']").each((index,
item) => {
            let $item = $(item),
                $img = $item.children('img'),
                A = $item.offset().top + $item.outerHeight()
/ 2;
            if (A <= B) {
                $item.attr('isLoad', 'TRUE');
                $img.attr('src', $img.attr('data-
img')).on('load', () => {
                    $img.stop().fadeIn(300);
                });
            }
        });
    };
    return {
        init:function(){
            setTimeout(lazyImgs, 500);
            window.onscroll = throttle(lazyImgs, 300);
        }
    }
})();
imgsModule.init();
</script>
```

28. 实现工具类方法toArray函数（至少两种解决方案）

```
let utils = (function(){
    function toArray(){
        /*方案一*/
        return Array.from(arguments);
        /*方案二*/
        return [].slice.call(arguments,0);
    }
    return {
        toArray
    };
})();
```

29. 完成如下需求

```
~function(){
    function queryURLParams(key){
        let obj = {};
        this.replace(/([^?=&#]+)=([^?=&#]+)/g, (...[, $1, $2]) => obj[$1] = $2);
        this.replace(/#([^?=&#]+)/g, (...[, $1]) => obj['_HASH'] = $1);
        if(key){
            return obj[key];
        }
        return obj;
    };
    String.prototype.queryURLParams=queryURLParams;
}();
```

30. 基于ES6中的class重构下面的代码

```
class Modal{
    constructor(x,y){
        this.x=x;
        this.y=y;
    }
    getX(){
        console.log(this.x);
    }
    getY(){
        console.log(this.y);
    }
    static setNumber(n){
        this.n=n;
```

```
    }
}
Modal.n=200;
Modal.prototype.z=10;
```