

js数据类型

7 大数据类型

- Number (数字)
- String (字符串)
- Boolean (布尔)
- Symbol (符号)
- Object (对象)
 - function (函数)
 - Array (数组)
 - Date (日期)
 - RegExp (正则表达式)
- Null (空)
- Undefined (未定义)

五种基本数据类型 Number,Null,Undefined,Boolean,String 一种复杂数据类型 Object es6 新加入 symbol 谈到 ES6, 就想到下面的问题了

null与undefined

- undefined 已声明, 未定义
- null 空值

判断数据类型的方法

一、typeof

- 优点: 能够快速区分基本数据类型
- 缺点: 不能将Object、Array和Null区分, 都返回object
- 注意! typeof(null)//object 但typeof 可以检测function

三、Object.prototype.toString.call()

```
var toString = Object.prototype.toString;

console.log(toString.call(1));           //[object Number]
console.log(toString.call(true));        //[object Boolean]
console.log(toString.call('abc'));       //[object String]
console.log(toString.call([]));          //[object Array]
console.log(toString.call({}));          //[object Object]
console.log(toString.call(function(){}));/[object Function]
console.log(toString.call(undefined));   //[object Undefined]
console.log(toString.call(null));        //[object Null]
```

- 优点: 精准判断数据类型
- 缺点: 写法繁琐不容易记, 推荐进行封装后使用

联系方式

- 电话号码/微信：15779887084
- QQ:958479953
- 邮箱：958479953@qq.com

个人信息

- 刘娇阳/女/1997.7
- 本科/信息学院网络工程专业/英语四级
- 技术博客：[Han's Blog](#)
- github: [liujiaoy](#)

个人技能

- 熟练使用HTML、CSS、jQuery、JS、Ajax进行功能开发；
- 熟练使用SVN、git版本控制工具，进行代码管理，实现敏捷开发；
- 熟悉Vue，VueX，axios，能使用脚手架进行开发；
- 能进行PC端，移动端网页开发；

工作经历

2019.12 - 至今 | 浙江亮鲸网络科技有限公司 | web前端开发工程师

负责公司ERP与MES的前端开发工作，**新需求开发与项目升级重构**

- 使用HTML，CSS，LigerUI编写页面，通过优雅降级处理**浏览器兼容问题**。
- 使用jQuery，原生JS实现页面所需交互，包括表格表头固定，tab切换，图片轮播等，**基础较扎实**。
- 将系统前后端未分离功能逐步**升级为前后端分离模式**，并优化性能。
- 封装公用方法与插件，并逐步整理成文档，**提高开发效率、规范前端团队开发流程**。

2019.7 - 2019.12 | 奥克斯空调股份有限公司 | WMS开发助理工程师

- 了解规范的开发流程，养成较好的注释及文档编写习惯。

项目经验

[VUE 仿去哪儿网项目移动端](#)

- 项目使用Vue脚手架开发，布局参考去哪儿网，实现了首页、景点详情、城市选择页面。
- 首页swiper轮播功能以及多区域列表的循环展示。
- header封装公用组件，利用slot满足不同模块需求。
- 城市选择页面实现城市展示、城市搜索、字母列表联动，VueX共享选中城市。
- 景点详情页面使用共用的画廊组件、渐隐渐显的header组件以及递归展示的列表组件。

一些常问问题

离职原因

- 留在一家公司满足三中其二：1.钱够 2.提升快 3.心情舒畅
- 想要离开说明不满足了，我想要寻找一个更合适的平台，希望技术水平和薪资都能得到较大提升。

关于薪资

- 原：工资6k * 13，提供住宿+午餐，综合 7.5k；
- 期望：9-10k

职业规划

这个问题有点大，也是我一直在思考的问题，近两三年的目标成为一名资深WEB前端工程师，扎实学习前端方面知识，厚积薄发。

垂直居中

高频问题 基本所有前端开发者都会遇到，日常常见问题，算一个基础问题，然后还能引申到flex布局上，面试上也很好切换问题方向 实现方式分几种

1. 基于定位

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
<style>
  #outBox{
    position: relative;
    width: 100%;
    height: 400px;
    background:aqua;
    overflow: auto;
  }
  /* 方法一 必须知道需要居中的box的宽度 */
  #centerBox{
    position: absolute;
    background-color: red;
    width:100px;
    height: 100px;
    top:50%;
    left: 50%;
    margin-top: -50px;
    margin-left: -50px;
  }
```

```

/* 方法二 可以不用知道需要居中的box的宽高, 但兼容性不好 */
#centerBox{
    position: absolute;
    background-color: red;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
/* 方法三 还是必须需要居中的box有宽高 */
#centerBox{
    position: absolute;
    background-color: red;
    width: 100px;
    height: 100px;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    margin: auto;
}

</style>
</head>
<body>
    <div id="outBox">
        <div id="centerBox">一个三四五</div>
    </div>
</body>
</html>

```

2. 通过js

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        #outBox{
            position: relative;
            width: 100%;
            height: 400px;
            background: aqua;
            overflow: auto;
        }
        #centerBox{
            position: absolute;
            background-color: red;
        }
    </style>
</head>
<body>
    <div id="outBox">
        <div id="centerBox">一个三四五</div>
    </div>
</body>
</html>

```

```

</style>
</head>
<body>
  <div id="outBox">
    <div id="centerBox">一个三四五</div>
  </div>
</body>
<script type="text/javascript">
  window.onload = function(){
    let outBox = document.getElementById("outBox"),
        outBoxH = outBox.offsetHeight,
        outBoxW = outBox.offsetWidth,
        centerBox = document.getElementById("centerBox"),
        centerBoxH = centerBox.offsetHeight,
        centerBoxW = centerBox.offsetWidth;
    centerBox.style.marginLeft = (outBoxW-centerBoxW)/2 + "px";
    centerBox.style.marginTop = (outBoxH-centerBoxH)/2 + "px";
  }
</script>
</html>
</script>

```

3. flex布局

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    #centerBox{
      background-color: red;
    }
    /*最简单，但也是兼容性问题*/
    #outBox{
      display: flex;
      justify-content: center;
      align-items: center;
      width: 800px;
      height: 400px;
      background: aqua;
    }
  </style>
</head>
<body>
  <div id="outBox">
    <div id="centerBox">一个三四五</div>
  </div>

```

```
</body>
</html>
```

清除浮动

float CSS属性指定一个元素应沿其容器的左侧或右侧放置，允许文本和内联元素环绕它。该元素从网页的正常流动(文档流)中移除，尽管仍然保持部分的流动性（与绝对定位相反）。

1. 触发BFC 即外层div增加属性overflow: hidden

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style type="text/css">
    /* 方法一 */
    #outBox{
      background-color:aqua;
      width: 400px;
      overflow: hidden;
    }
    #centerBox{
      float: left;
      background-color: red;
    }
  </style>
</head>
<body>
  <div id="outBox">
    <div id="centerBox">一个三四五</div>
  </div>
</body>
</html>
```

2. 额外增加标签并设置clear:both

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style type="text/css">
    /* 方法一 */
    #outBox{
```

```

        background-color:aqua;
        width: 400px;
    }
    #centerBox{
        float: left;
        background-color: red;
    }
    .for-clear-float{
        clear: both;
    }
</style>
</head>
<body>
    <div id="outBox">
        <div id="centerBox">一个三四五</div>
        <div class="for-clear-float"></div>
    </div>
</body>
</html>

```

3. 增加after伪元素

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style type="text/css">
/* 方法一 */
    #outBox{
        background-color:aqua;
        width: 400px;
    }
    #centerBox{
        float: left;
        background-color: red;
    }
    .clear{
        *zoom: 1;
    }
/*加在父div上*/
    .clear:after{
        content: "";
        display: block;
        clear: both;
        height: 0;
        visibility: hidden;
    }

```

```
</style>
</head>
<body>
  <div id="outBox" class="clear">
    <div id="centerBox" >一个三四五</div>
  </div>
</body>
</html>
```

4. 引申思考

1) 既然会有高度塌陷问题，那为什么要用float，他干什么用的

float设计用来做文字环绕效果，这也是他出现高度塌陷的原因，只有高度塌陷了才好实现文字环绕效果 具体可以查看博客[CSS布局\(四\) float详解](#)

关于BFC

这个问题可以穿插到清除浮动时问。对于BFC，感觉是知其然而不知其所以然，只了解基本概念，比如全称是Block Formatting Context（块格式化上下文），可以清除浮动。却没有详细去了解，今天查看了一下才有一种恍然大悟的感觉

[参考博客地址](#) [MDN文档地址](#) BFC的特性 1、属于同一个BFC的两个相邻容器的上下margin会重叠（重点） 2、计算BFC高度时浮动元素也参与计算（重点） 3、BFC的区域不会与浮动容器发生重叠（重点） 4、BFC内的容器在垂直方向依次排列 5、元素的margin-left与其包含块的border-left相接触 6、BFC是独立容器，容器内部元素不会影响容器外部元素

BFC的触发条件 1、根元素（html） 2、float值非none 3、overflow值非visible 4、display值为inline-block、table-cell、table-caption、flex、inline-flex 5、position值为absolute、fixed

在验证margin重叠问题时，以下情况会出现margin重叠

```
<body>

  <div class="boxa">
    boxa
  </div>
  <div class="boxb">
    boxb
  </div>

</body>
```



```
.boxa, .boxb{
  margin: 20px;
  height: 20px;
}
```



```

.bboxa{
  background-color: aqua;
  overflow: hidden;
}
.bboxb{
  background-color: bisque;
}

```

 图片 此时我想我触发boxa BFC应该就可以了，然后我在boxa 上添加属性overflow:hidden后没有效果，具体也没查到这个问题， 其余博客都是说在boxa外层再加一个div，这个div触发BFC即可。我试了确实成功了...困惑了很久 现在我猜测的是根元素是BFC，所以boxa，boxb属于同一个BFC，即使boxa触发了BFC，他们也还是同一个BFC 只要再boxa外层在套一个div触发BFC，boxa和boxb才处于不同的BFC。。。但是我通过设置boxa为inline-block来触发BFC，margin又不重叠了.....这下我真的迷惑了，继续找结果，最后在知乎找到了 直接放图片了  关于margin重叠

介绍一下标准的css的盒子模型？ 低版本IE的盒子模型有什么不同

标准盒模型 content 的宽高为元素的width， height， 一个块的总宽 = width + padding +border +margin; IE盒模型 content + padding + border 才是元素的width， height， 一个块的总宽 = width + margin; 通常开发中，用的多的是IE盒模型

css3新特性

1. 动画 animation

animation是个缩写属性，包含以下六个属性 默认none 0 ease 0 1 normal animation-name: keyframe 名称 animation-duration: 完成动画时间 animation-timing-function: 规定动画的速度曲线 animation-delay: 动画开始之前的延迟 animation-iteration-count: 规定动画应该播放的次数 animation-direction: 规定是否应该轮流反向播放动画 例子：实现一个简单的音乐震动条吧

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style type="text/css">
    #outBox{
      background-color:aqua;
      width: 800px;
      height:400px;
      display: flex;
      align-items: center;
      justify-content: center;
    }
    #centerBox{
      height: 100px;
      width: 100px;
    }
  </style>
</html>

```

```

}
#wave{
  height: 100%;
  width: 100%;
  display: flex;
  /* flex-direction: column; */
  flex-wrap: wrap;
  justify-content: center;
  align-items: flex-end;
  padding: 0;
  margin: 0;
}
#wave li{
  list-style-type: none;
  height: 0;
  width: 15px;
  background-color: brown;
  margin: 0 2px;
}
#centerBox #wave .li1{
  animation: waves .8s linear 1s infinite alternate;
}
#wave .li2{
  animation: waves 1s linear .2s infinite alternate;
}
#wave .li3{
  animation: waves 2s linear .5s infinite alternate;
}
#wave .li4{
  animation: waves 1s linear .5s infinite alternate;
}
#wave .li5{
  animation: waves 1.5s linear .5s infinite alternate;
}
@keyframes waves{
  10% {
    height: 20%;
  }
  20%{
    height: 60%;
  }
  40% {
    height: 40%;
  }
  50% {
    height: 100%;
  }
  100%{
    height: 50%;
  }
}
</style>
</head>
<body>

```

```

<div id="outBox">
  <div id="centerBox">
    <ul id="wave">
      <li class="li1"></li>
      <li class="li2"></li>
      <li class="li3"></li>
      <li class="li4"></li>
      <li class="li5"></li>
    </ul>
  </div>
</div>
</body>
</html>

```

2. 过渡 transition

transition: css属性, 花费时间, 曲线效果, 延迟时间(默认0) 引用W3C的例子, 鼠标放上去出效果

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
<style type="text/css">
  #outBox{
    background-color:aqua;
    width: 800px;
    height:400px;
  }
  #centerBox{
    float: left;
    background-color: red;
    width: 100px;
  }

  /* transition过渡效果 */
  #centerBox:hover{
    width:300px;
    transition: width,.5s,linear,2s;
  }
</style>
</head>
<body>
<div id="outBox" class="clear">
  <div id="centerBox" >鼠标放上来</div>
</div>
</body>
</html>

```

3. 形状转换

translate前面的垂直居中用到过 rotate 旋转 做一个导航栏悬浮箭头旋转90度的案例，这里设计到自己用css画三角形

1)css画三角形，利用border实现

```
<div id="triangle-up"></div>
<div id="triangle-down"></div>
<div id="triangle-right"></div>
<div id="triangle-left"></div>
```

```
/* 实现三角形 */
#triangle-up {
    width: 0;
    height: 0;
    /* 构造三条边 */
    border-left: 50px solid transparent;
    border-right: 50px solid transparent;
    border-bottom: 100px solid red;
}

#triangle-down {
    width: 0;
    height: 0;
    /* 构造三条边 */
    border-left: 50px solid transparent;
    border-right: 50px solid transparent;
    border-top: 100px solid blue;
}

#triangle-left{
    width: 0;
    height: 0;
    /* 构造三条边 */
    border-top: 50px solid transparent;
    border-bottom: 50px solid transparent;
    border-right: 100px solid green;
}

#triangle-right{
    width: 0;
    height: 0;
    /* 构造三条边 */
    border-top: 50px solid transparent;
    border-bottom: 50px solid transparent;
    border-left: 100px solid pink;
}
```

2)导航栏案例

```
<div id="nav">
  <ul>
    <li class="title" id="title">
      <span>一级标题</span><span class="triangle-right"></span>
      <ul class="sub-title" >
        <li>二级标题</li>
        <li>二级标题</li>
        <li>二级标题</li>
      </ul>
    </li>
  </ul>
</div>
```

```
.title{
  position: relative;
}
.triangle-right{
  position: absolute;
  top: 6px;
  width: 0;
  height: 0;
  /* 构造三条边 */
  border-top: 5px solid transparent;
  border-bottom: 5px solid transparent;
  border-left: 10px solid pink;
}
.sub-title{
  visibility: hidden;
  opacity: 0;
}
#title:hover .sub-title{
  visibility: visible;
  opacity: 1;
  transition: opacity, 3s, linear, 2s;
}
#title:hover .triangle-right{
  transform: rotate(90deg);
  transition: all, .5s, ease, 1s;
}
```