JavaScript 基础第五天

对象

黑马程序员
www.itheima.com
传智教育旗下
高端IT教育品牌

目录

Contents

- 对象
- 综合案例
- 数据类型存储

01

对象

- 对象使用
- 操作对象
- 遍历对象
- 内置对象

对象-Object

对象（Object）：JavaScript里的一种数据类型（引用类型），也是用于存储数据的

好处：可以用来详细的描述某个事物，是用键值对形式存储语义更明了

特点： 对象数据是无序的，数组有序的

```
let arr = ['佩奇', 4, '女', 12.6]
```

```
let pig = {
  uname: '佩奇',
  age: 4,
  sex: '女',
  weight: 12.6
}
```

注意：声明对象使使用大括号，大括号就是对象字面量

```
let 对象名 = {}
```

对象-Object

对象组成

属性

- 事物的描述信息称为属性
- 如姓名、身高、年龄、性别等，一般是名词

```
let pig = {
  uname: '佩奇',
  age: 4,
  sex: '女',
  weight: 12.6
}
```

方法

- 事物的行为性称为方法
- 如跑步、唱歌等，一般是动词

```
let pig = {
  sing: function() {
    console.log('哼~~')
  }
}
```

1.1 对象使用-属性

1. 定义对象属性

属性都是成对出现的，包括属性名和属性值

```
let pig = {
  uname: '佩奇',
  age: 4,
  sex: '女',
  weight: 12.6
}
```

- 属性名和属性值之间使用英文：分隔
- 多个属性之间使用英文，分隔，属性是**无序的**
- 属性就是依附在对象上的变量（对象外是变量，对象内是属性）

2. 访问对象属性

访问对象属性可以得到里面的值

```
对象.属性名
```

```
console.log(pig.age)  // 4
```

- 点 . 可以理解为 的，对象的某个属性

总结

1. 对象属性有顺序吗？
   - 没有
2. 对象属性和值用什么符号隔开？多个属性用什么隔开？
   - 属性和值之间用；隔开
   - 多个属性之间用，逗号隔开
3. 如何得到对象的属性值？
   - 对象.属性

```
let pig = {
  uname: '佩奇',
  age: 4,
  sex: '女',
  weight: 12.6
}
```

```
console.log(pig.age)  // 4
```

练习

请声明一个商品对象，里面包如下属性：(同学独立完成)

要求：

商品对象名字： goods

商品名称命名为： name

商品编号： num

商品毛重： weight

商品产地： address

品牌： 小米（MI）

商品名称：小米小米10 青春版　商品编号：100012816024　商品毛重：0.55kg　商品产地：中国大陆

1.1 对象使用-方法

1. 定义对象方法

方法是成对出现的，包括方法名和匿名函数

```
let pig = {
  uname: '佩奇',
  sing: function() {
    console.log('哼~~')
  }
}
```

- 方法名和匿名函数之间使用英文：分隔
- 多个方法之间使用英文，分隔，方法是无序的
- 方法是依附在对象中的函数 （对象外是函数，对象内是方法）

2. 调用对象方法

通过对象.方法() 可以调用对象的方法

```
let pig = {
  uname: '佩奇',
  sing: function() {
    console.log('哼~~')
  }
}
pig.sing()  // 调用对象的sing方法
```

形参

实参

注意： 千万别忘了给方法名后面加小括号

总结

1. 定义对象方法时，方法和匿名函数用什么分割？
   - ：冒号分割
2. 对象访问方法是如何实现的？
   - 对象.方法()
3. 对象方法可以传递参数吗？
   - 可以，跟函数使用方法基本一致

```
let pig = {
  uname: '佩奇',
  sing: function() {
    console.log('哼~~')
  }
}
pig.sing()  // 调用对象的sing方法
```

01

对象

- 对象使用
- 操作对象
- 遍历对象
- 内置对象

1.2 操作对象

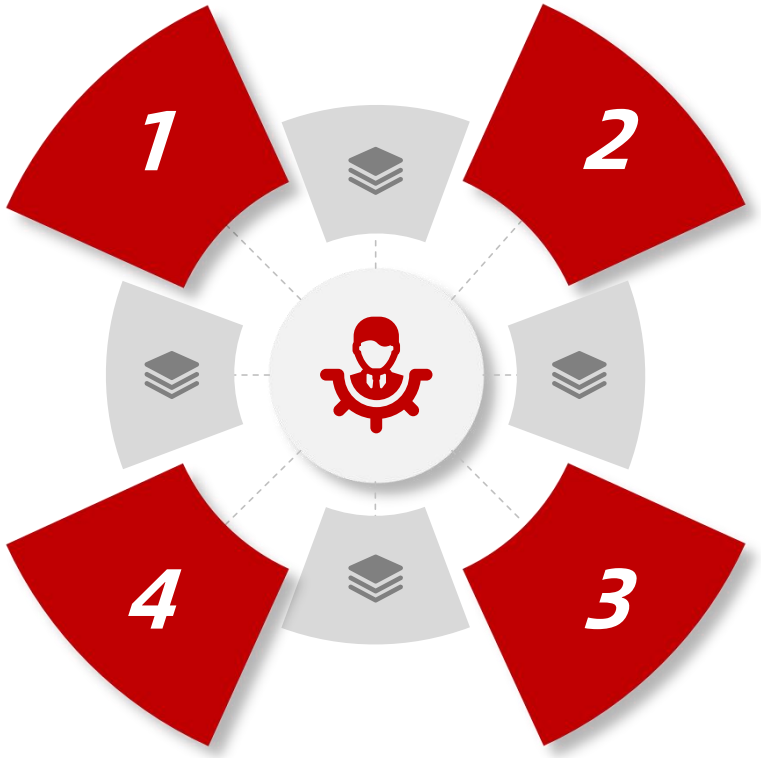
- 对象本质是无序的数据集合，操作对象数据无非就是 增 删 改 查 语法：

查询对象

对象.属性

修改对象

对象.属性 = 新值



增加对象内容

对象.新属性 = 新值

对象.新方法名 = function(){}

删除对象内容

delete 对象名.属性名

1.2 操作对象 - 查、改、增、删

查找语法	对象.属性
修改语法	对象.属性 = 新值
新增语法	对象.新属性 = 新值
总结	改和增语法一样，判断标准就是对象有没有这个属性，没有就是新增，有就是改
删除语法	delete 对象.属性（了解）

练习 请对产品对象，做如下操作：(同学自行完成)

要求:

1. 请将商品名称name里面的值修改为： 小米10 PLUS
2. 新增一个颜色属性： color 为 '粉色'
3. 请依次页面打印输出所有的属性值

原先:

商品对象名字： goods

商品名称命名为： name

商品编号： num

商品毛重： weight

商品产地： address

品牌： 小米（MI）

商品名称：小米小米10 青春版　商品编号：100012816024　商品毛重：0.55kg　商品产地：中国大陆

1.2 操作对象

属性-查的另外一种写法

- 对于多词属性比如中横线分割的属性，点操作就不能用了
- 我们可以采取： 对象['属性'] 方式， 单引号和双引号都阔以，当然也可以用于其他正常属性

总结：

多词属性或者需要解析变量的时候使用 [] 语法，其余的直接使用点语法

```
let pig = {
  'pig-name': '佩奇',
  age: 4
}
console.log(pig.pig - name)  // NaN
```

```
let pig = {
  'pig-name': '佩奇',
  age: 4
}
console.log(pig['pig-name']) // 佩奇
```

总结

1. 对象访问属性有哪两种方式？
   - 点形式 对象.属性
   - [] 形式 对象['属性']
2. 两种方式有什么区别？
   - 点后面的属性名一定不要加引号
   - [ ] 里面的属性名一定加引号
   - 后期不同使用场景会用到不同的写法
     - 多词属性，加了中横线等
     - 解析变量

```
let pig = {
  'pig-name': '佩奇',
  age: 4
}
console.log(pig['pig-name']) // 佩奇
```

01

对象

- 对象使用
- 操作对象
- 遍历对象
- 内置对象

1.3 遍历对象

- for 遍历对象的问题：
  - 对象没有长度length，而且是无序的
- **利用 for in 遍历对象**

```
for (let 变量 in 对象) {
  console.log(变量)  // 属性名
  console.log(对象[变量])  // 属性值
}
```

  - for in语法中的 k 是一个变量, 在循环的过程中依次代表对象的属性名
  - 由于 k 是变量, 所以必须使用 [ ] 语法解析
  - 一定记住： **k** 是获得对象的**属性名**， **对象名[k]** 是获得 **属性值**
  - 一般不用这种方式遍历数组、主要是用来遍历对象

```
let pig = {
  uname: '佩奇',
  age: 4,
  sex: '女',
  weight: 12.6
}
console.log(pig.uname)  // 佩奇
console.log(pig.age)  // 4
console.log(pig.sex)  // 女
console.log(pig.weight)  // 12.6
```

```
let pig = {
  uname: '佩奇',
  age: 4,
  sex: '女',
  weight: 12.6
}
for (let k in obj) {
  console.log(k)  // 属性名
  console.log(obj[k]) // 属性值
}
```

总结

1. 遍历对象用哪个语句？
   - for in
2. 遍历对象中，for k in obj，获得对象属性是哪个？获得值是那个？
   - 获得对象属性是 k
   - 获得对象值是 obj[k]　注意 k是变量，所以采用 对象[] 语法获得属性值
3. 什么时候用[] 获取属性值？
   - 多词属性，比如中横线
   - 解析变量

```
let obj = {
  'pig-name': '佩奇'
}
console.log(obj['pig-name'])
```

```
for (let k in obj) {
  console.log(k)  // 属性名
  console.log(obj[k]) // 属性值
}
```

案例 **遍历对象数组**

// 定义一个1个学生信息的对象

```
{ name: '小明', age: 18, sex: '男', hometown: '河北省' },
```

案例 **遍历对象数组**

// 定义一个若干学生信息对象

```
{ name: '小明', age: 18, sex: '男', hometown: '河北省' },
{ name: '小红', age: 19, sex: '女', hometown: '河南省' },
{ name: '小刚', age: 17, sex: '男', hometown: '山西省' },
{ name: '小丽', age: 18, sex: '女', hometown: '山东省' }
```

案例

遍历对象数组

需求：请把下面所有数据中的家乡打印出来：

// 定义一个存储了若干学生信息的数组

```
let students = [
  { name: '小明', age: 18, sex: '男', hometown: '河北省' },
  { name: '小红', age: 19, sex: '女', hometown: '河南省' },
  { name: '小刚', age: 17, sex: '男', hometown: '山西省' },
  { name: '小丽', age: 18, sex: '女', hometown: '山东省' }
]
```

以上数据称为 对象数组

案例

渲染表格案例

需求：根据以上数据渲染生成表格

学生信息

将数组中存储的学生信息，以表格的形式把数据渲染到页面中...

序号	姓名	年龄	性别	家乡
1	小明	18	男	河北省
2	小红	19	女	河南省
3	小刚	17	男	山西省
4	小丽	18	女	山东省

案例

渲染表格案例

需求：根据以上数据渲染生成表格

分析：

①：中间的行根据数据对象个数，利用字符串拼接，生成对应的行标签

②：把生成的字符串放入表格 tbody里面

```
let students = [
  { name: '小明', age: 18, sex: '男', hometown: '河北省' },
  { name: '小红', age: 19, sex: '女', hometown: '河南省' },
  { name: '小刚', age: 17, sex: '男', hometown: '山西省' },
  { name: '小丽', age: 18, sex: '女', hometown: '山东省' }
]
```

学生信息

将数组中存储的学生信息，以表格的形式把数据渲染到页面中...

序号	姓名	年龄	性别	家乡
1	小明	18	男	河北省
2	小红	19	女	河南省
3	小刚	17	男	山西省
4	小丽	18	女	山东省

01

对象

- 什么是对象
- 对象使用
- 遍历对象
- 内置对象

1.4 内置对象

目标：学会调用JavaScript为我们准备好的内置对象

学习路径

1. 内置对象是什么
2. 内置对象Math
3. 生成任意范围随机数

1.4 内置对象

- **内置对象：**JavaScript内部提供的对象，包含各种属性和方法给开发者调用
- 思考：我们之前用过内置对象吗？
  - document.write()
  - console.log()

1.4 内置对象-Math

- **介绍：**Math对象是JavaScript提供的一个“数学”对象
- **作用：**提供了一系列做数学运算的方法
- Math对象包含的有： Math对象在线文档

属性/方法	作用	说明
PI	圆周率	Math.PI 属性，返回圆周率
max	找最大值	Math.max(8, 3, 1) 方法，返回 8
min	找最小值	Math.min(8, 3, 1) 方法，返回 1
abs	绝对值	Math.abs(-1) 方法，返回 1
ceil	向上取整	Math.ceil(3.1) 方法，返回 4
floor	向下取整	Math.floor(3.8) 方法，返回 3
round	四舍五入取整	Math.round(3.8) 方法，返回 4， 遇到.5则舍入到相邻的在正无穷（+∞）方向上的整数

1.4 内置对象Math-生成任意范围随机数

- Math.random()　随机数，　返回一个0 - 1之间，并且包括0不包括1的随机小数　[0, 1）
- 如何生成0-10的随机整数呢？

```
Math.floor(Math.random() * (10 + 1))
```

- 如何生成5-15的随机整数？

```
Math.floor(Math.random() * (10 + 1)) + 5
```

- 如何生成N-M之间的随机整数

```
Math.floor(Math.random() * (M - N + 1)) + N
```

```
Math.floor(Math.random() * (差值 + 1)) + 最小值
```

案例

随机显示名字案例

需求：请把 ['赵云', '黄忠', '关羽', '张飞', '马超'] 随机显示一个名字到页面中

0 1 2 3 4

分析：

①：利用 random 随机生成一个数字（整数）作为索引号

②： 数组[随机数] 生成到页面中

案例

猜数字游戏

需求：程序生成 1~10 之间的一个随机整数，用户输入一个数字，返回猜的是否正确

①：如果输入数字大于随机数字，就提示：'数字猜大了，继续猜'

②：如果输入数字小于随机数字，就提示：'数字猜小了，继续猜'

③：如果输入数字等于随机数字，就提示猜对了，程序结束

案例

猜数字游戏

需求：程序随机生成 1~10 之间的一个数字，用户输入一个数字，返回猜的是否正确

分析：

①：利用 random 生成一个随机整数

②：需要一直猜，所以需要不断的循环（无限循环 while）

③：用户输入数字，利用多分支语句判断，猜对了就退出，否则提示猜大了还是猜小了

案例

生成随机颜色

需求：该函数接收一个布尔类型参数，表示颜色的格式是十六进制还是rgb格式。

①：如果参数传递的是 true 或者无参数，则输出 一个随机十六进制的颜色

②：如果参数传递的是 false ，则输出 一个随机rgb的颜色

③：格式：

```
function getRandomColor(flag) {
  // 内容
}
console.log(getRandomColor(true))    // #ffffff
console.log(getRandomColor(false))   // rgb(255,255,255)
```

案例

生成随机颜色

需求：该函数接收一个布尔类型参数，表示颜色的格式是十六进制还是rgb格式。

分析：

①： 封装颜色函数 getRandomColor

②： 返回 rgb 格式颜色

③： 返回 #ffffff 格式颜色

案例

生成随机颜色

需求：该函数接收一个布尔类型参数，表示颜色的格式是十六进制还是rgb格式。

分析：

rgb颜色格式为：'rgb(255, 255, 255) ' 其中255可以是任意0-255之间的数字

步骤：

①：利用random随机生成 3 个 0 ~ 255 之间的整数，分别作为 rgb 三个颜色

②：把变量利用模板字符串 `rgb(255, 255, 255)` 填充并返回

案例

生成随机颜色

需求：该函数接收一个布尔类型参数，表示颜色的格式是十六进制还是rgb格式。

分析：

①： 封装颜色函数 getRandomColor

②： 返回 rgb 格式颜色

③： 返回 #ffffff 格式颜色

案例

生成随机颜色

需求：该函数接收一个布尔类型参数，表示颜色的格式是十六进制还是rgb格式。

分析：

16进制颜色格式为：'#00ffff'　其中f可以是任意 0-f 之间的字符,需要用到数组，

let arr = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f']

步骤：

①：处理16进制颜色，核心思想是循环6次，生成随机的6个数字（取值范围0~15），

②：根据这个数字去找数组的值，拼接到字符串里面，并且返回

目录

Contents

- 对象
- 综合案例
- 数据类型存储

案例

学成在线页面渲染案例

需求：根据数据渲染列表页面

案例

学成在线页面渲染案例

需求：根据数据渲染列表页面

分析：

①：根据对象数组的数据来渲染页面

②：核心利用循环来做字符串拼接生成多个标签，然后添加到容器中渲染页面

目录

Contents

- 对象
- 综合案例
- 数据类型存储

数据类型小结

JS 数据类型整体分为两大类：

- 基本数据类型（简单数据类型）
- 引用数据类型（复杂数据类型）

基本数据类型

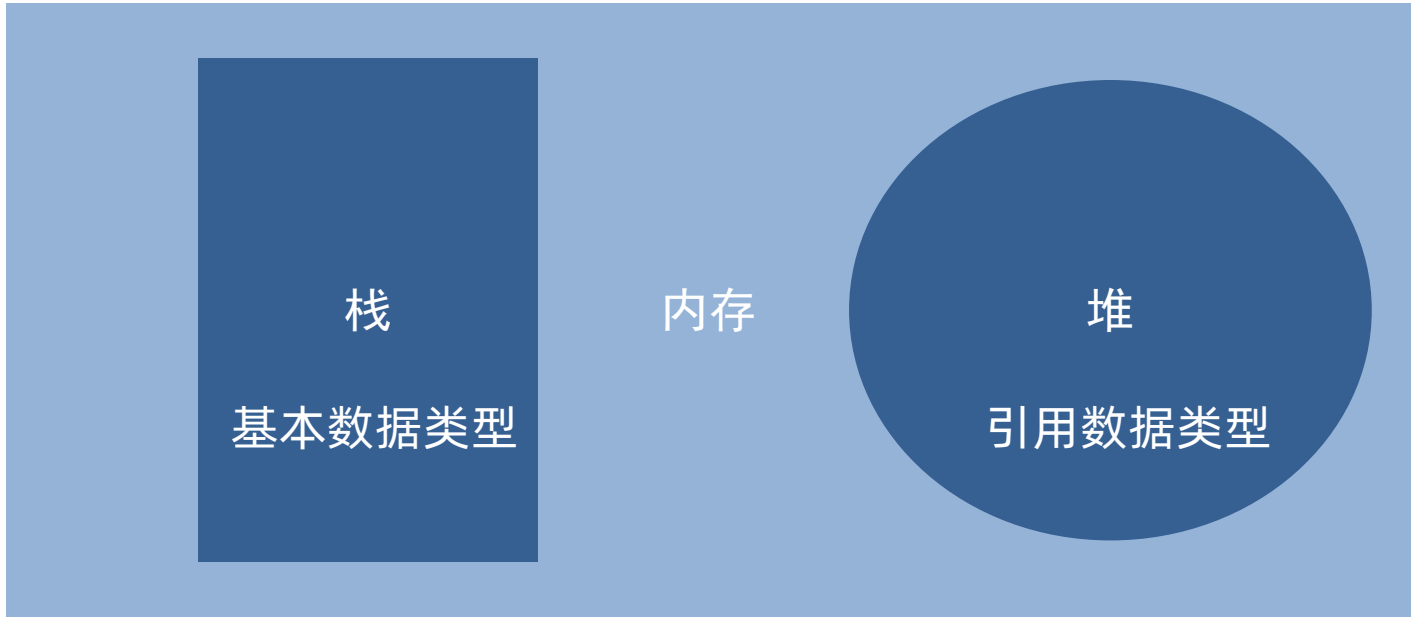
number 数字型
string 字符串型
boolean 布尔型
undefined 未定义型
null 空类型

引用数据类型

Object、Function、Array

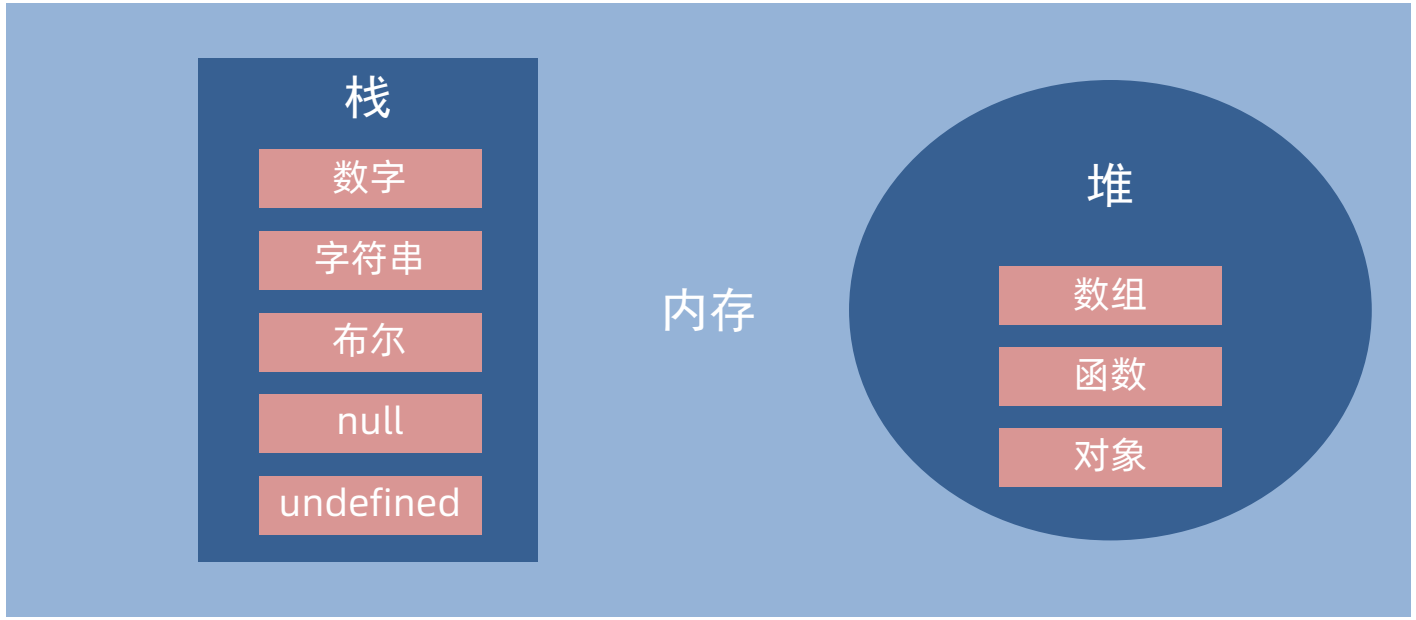
内存中堆栈空间分配区别：

- 栈: 优点访问速度快，基本数据类型存放到栈里面
- 堆:优点存储容量大，引用数据类型存放到堆里面



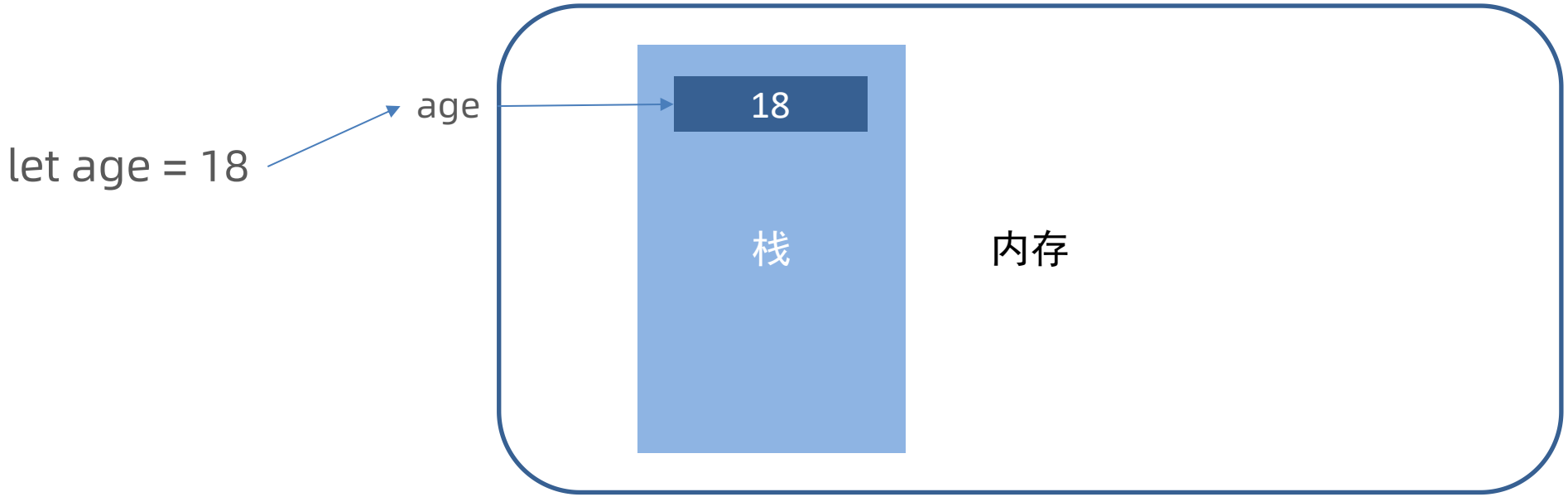
内存中堆栈空间分配区别：

- 栈: 优点访问速度快，基本数据类型存放到栈里面
- 堆:优点存储容量大，引用数据类型存放到堆里面



基本类型内存分配-栈

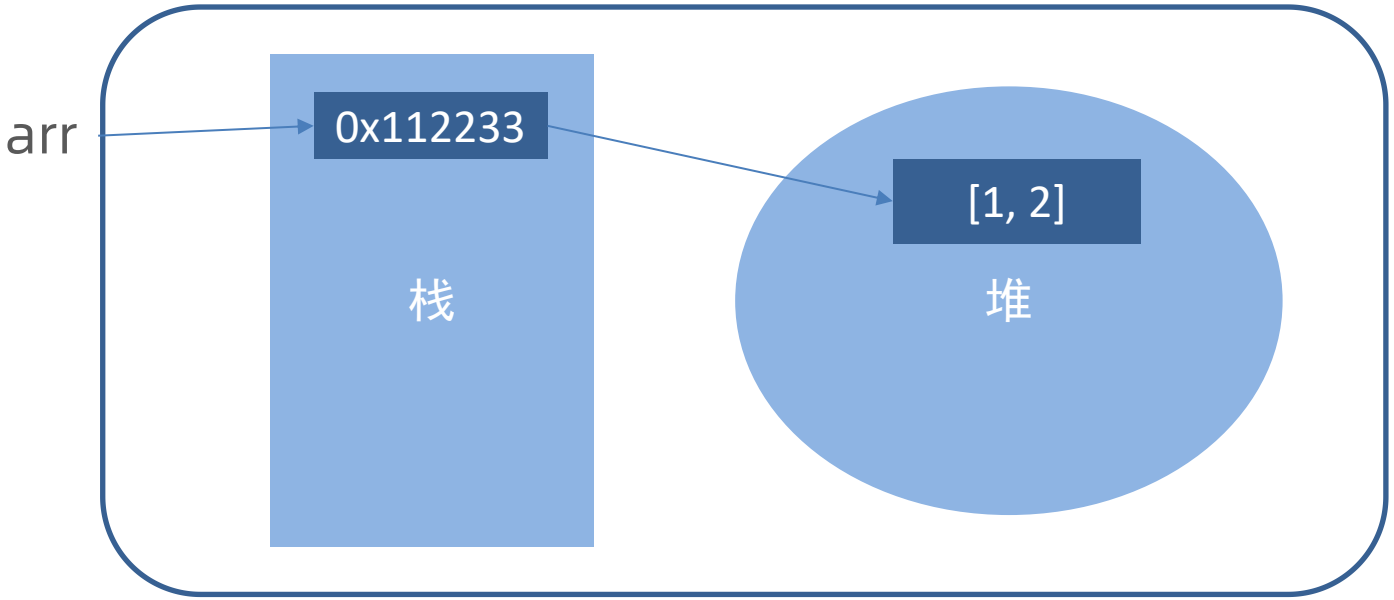
- 基本数据类型：变量的数据直接存放在栈空间中
- 优点：访问速度快



引用类型内存分配-堆

- 引用类型（复杂数据类型）：如 Object、Array、Function等
- 引用类型变量（栈空间）里存放的是地址，真正数据存放在堆空间中
- 优点：容量大

let arr = [1, 2]



思考

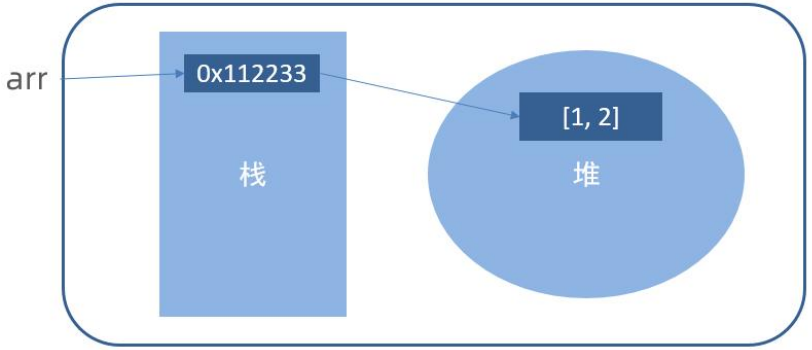
```
// 第一题
let x = 10
let y = x
x = 20
console.log(y)

// 第二题

let obj1 = {
  age: 18
}
let obj2 = obj1
obj1.age = 20
console.log(obj2)
```

总结

1. 基本数据类型存放内存哪个位置？ 引用数据类型呢？
   - 基本数据类型放 栈 里面
   - 引用数据类型放 堆 里面
2. 赋值变量时，基本数据类型存放的是什么？ 引用数据类型呢？
   - 基本数据类型赋值的是 值
   - 引用数据类型赋值的是 地址



变量声明总结

- 变量声明有三个 var　let　和 const，我们应该用哪个呢？
  - 首先 var 先排除，老派写法，问题很多，可以淘汰掉…
- let　or　const ？
- 建议：　const 优先，尽量使用const，原因是：
  - ✓ 很多变量声明的时候明确不会被更改了，为了安全性，那为什么不用 const 呢？
  - ✓ 实际开发中也是，比如 React、Vue，基本 const 声明
- 如果你还在纠结，那么我建议：
  - 有了变量先给const，如果发现它后面是要被修改的，再改为let

变量声明

- 请问以下的 可不可以把let 改为 const？

```
let num1 = +prompt('请输入第一个数值:')
let num2 = +prompt('请输入第二个数值:')
alert(`两者相加的结果是: ${num1 + num2}`)
```

```
function sum() {
  let x = 10
  let y = 20
  console.log(x + y)
}
sum()
```

变量声明

- 请问以下的 可不可以把let 改为 const？

```
<script>
    let num = 1
    num = num + 1
    console.log(num)  // 结果是 2
</script>
```

```
for (let i = 0; i < nums.length; i++) {
    document.write(nums[i])
}
```

变量声明

- 请问以下的 可不可以把let 改为 const？

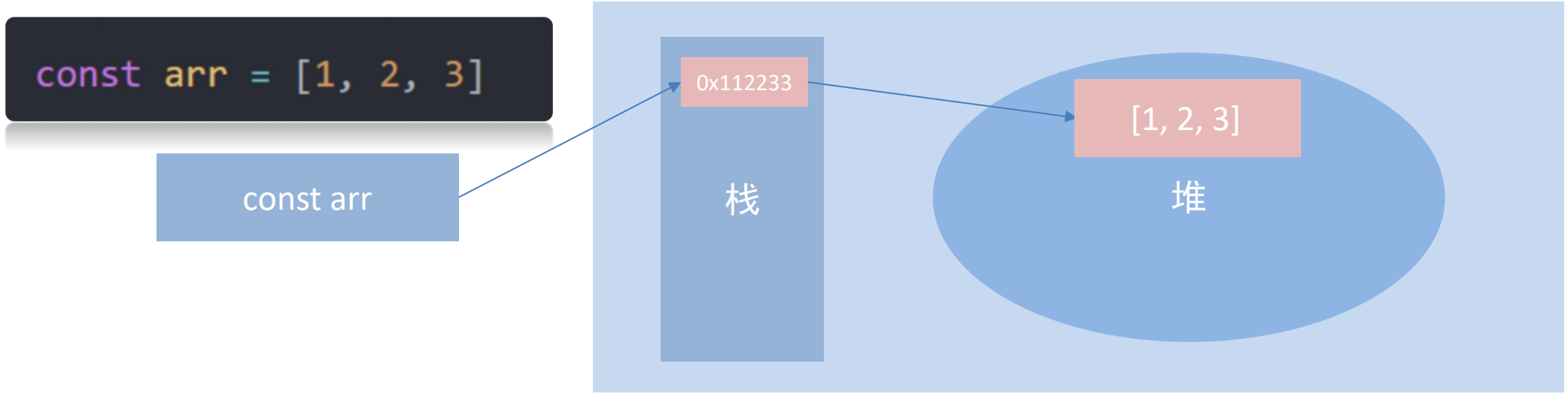
```
let arr = ['red', 'green']
arr.push('pink')
console.log(arr)  //  ['red', 'green', 'pink']
```

```
let pig = {
  uname: '佩奇',
  age: 4,
  sex: '女',
  weight: 12.6
}
pig.color = 'pink'
```

变量声明

- const 声明的值不能更改，而且const声明变量的时候需要里面进行初始化
- 但是对于引用数据类型，const声明的变量，里面存的不是 值，不是值，不是值，是 地址
- 所以： const声明的变量不可以修改，本质是说，const声明的变量 地址不能修改

```
const arr = [1, 2, 3]
```

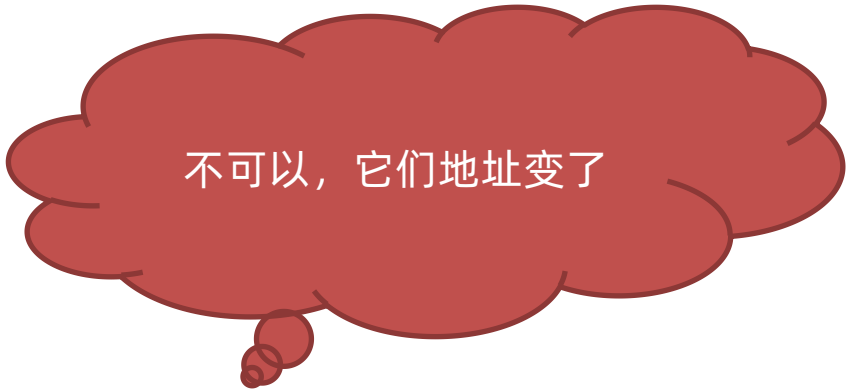


变量声明

- 考考你：下面写法可以吗？

```
const names = []
names = [1, 2, 3]
```

```
const obj = {}
obj = {
  uname: 'pink老师'
}
```



变量声明

- 考考你：下面写法可以吗？

```
const names = []
names = [1, 2, 3]
```

```
const names = []
names[0] = 1
names[1] = 2
names[2] = 3
```

```
const obj = {}
obj = {
  uname: 'pink老师'
}
```

```
const obj = {}
obj.uname = 'pink老师'
```

总结

1. 以后声明变量我们优先使用哪个？
   - const
   - 有了变量先给const，如果发现它后面是要被修改的，再改为let
2. 为什么const声明的对象可以修改里面的属性？
   - 因为对象是引用类型，里面存储的是地址，只要地址不变，就不会报错
   - 建议数组和对象使用 const 来声明
3. 什么时候使用let声明变量？
   - 变量被重新赋值的时候需要用let
   - 比如 一个变量进行加减运算，比如 for循环中的 i++

拓展-术语解释（导师讲解）

目标：知道一些术语，让自己更专业

术语	解释	举例
关键字	在JavaScript中有特殊意义的词汇	let、var、function、if、else、switch、case、break
保留字	在目前的JavaScript中没意义，但未来可能会具有特殊意义的词汇	int、short、long、char
标识（标识符）	变量名、函数名的另一种叫法	无
表达式	可以被求值的代码，一般配合运算符出现	10 + 3、age >= 18
语句	一段可执行的代码	if () for()

黑马程序员

www.itheima.com

传智教育旗下高端IT教育品牌