web APIs 第五天

BOM操作（本地存储）

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

学习目标

Learning Objectives

1. 依托 BOM 对象实现对历史、地址、浏览器信息的操作或获取
2. 具备利用本地存储实现学生就业表案例的能力

目录

Contents

BOM

- window对象
- 定时器-延时函数
- location对象
- navigator对象
- histroy对象
- 本地存储

- 目标：学习 window 对象的常见属性，知道各个 BOM 对象的功能含义

JavaScript的组成

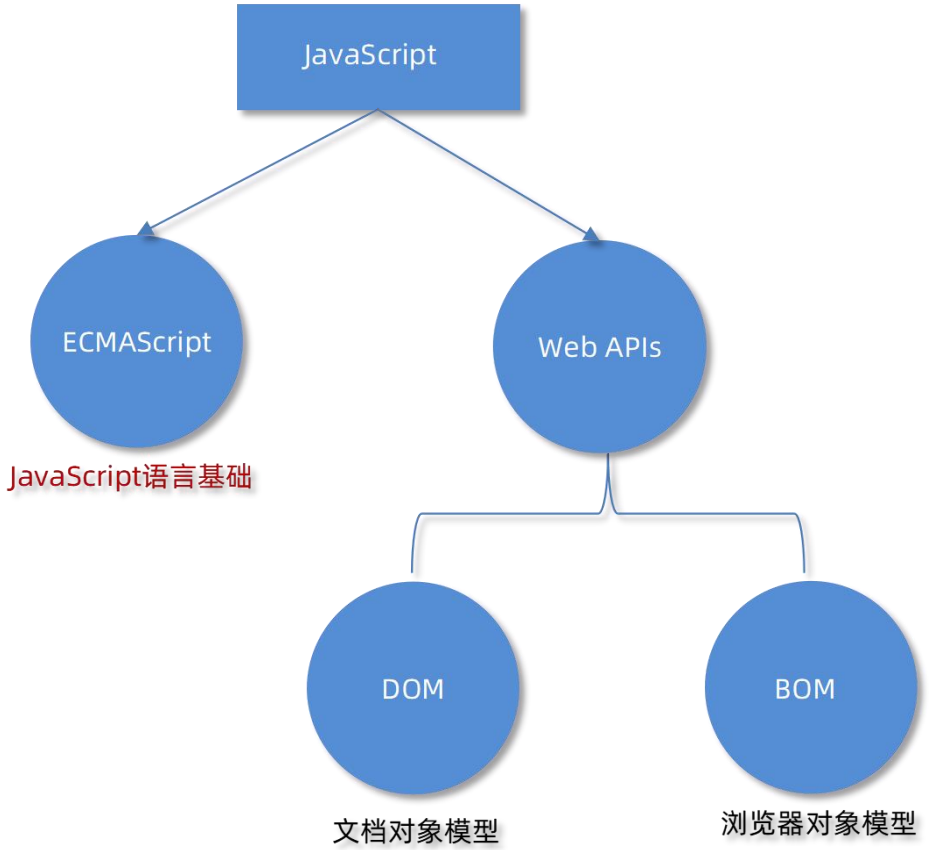
- ECMAScript:

  规定了js基础语法核心知识。

  - 比如：变量、分支语句、循环语句、对象等等

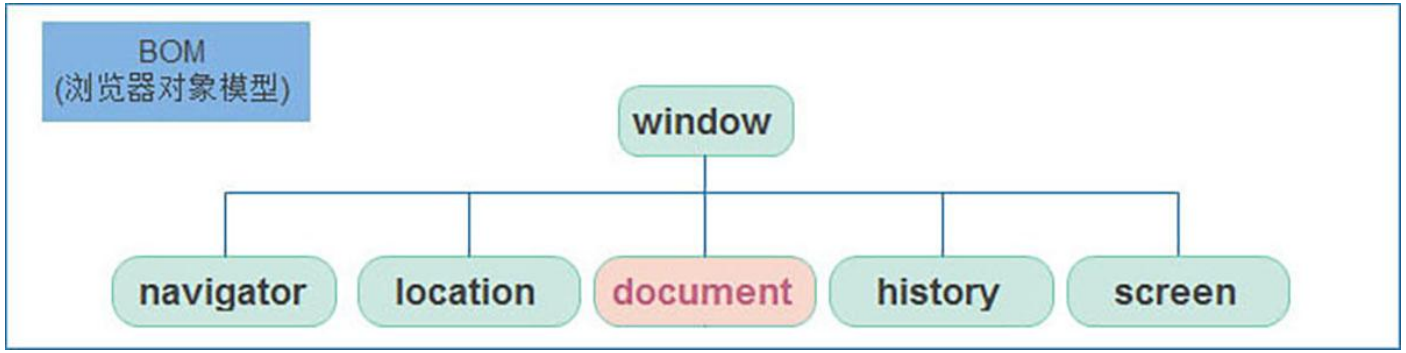
- Web APIs ：

  - DOM 文档对象模型， 定义了一套操作HTML文档的API
  - BOM 浏览器对象模型，定义了一套操作浏览器窗口的API

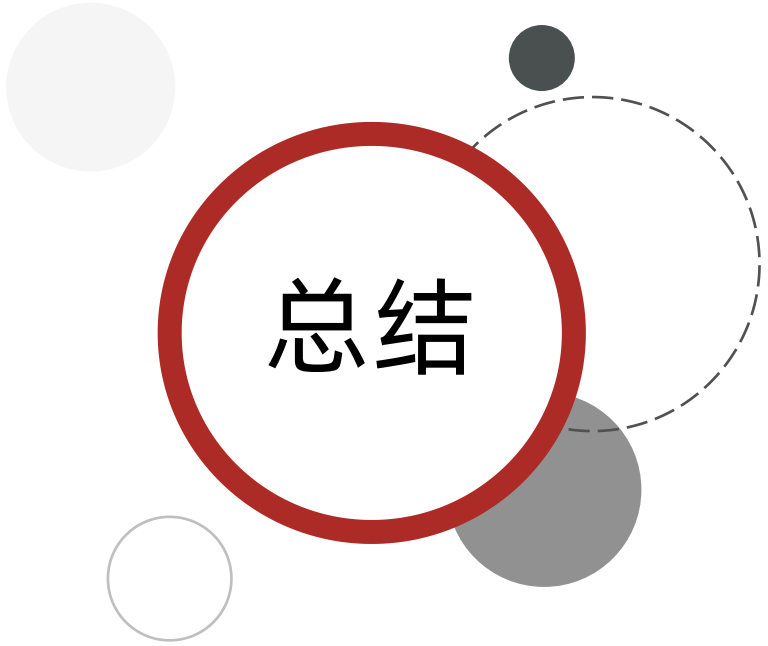


1.1 BOM

- BOM（Browser Object Model ）是浏览器对象模型



- window对象是一个全局对象，也可以说是JavaScript中的顶级对象
- 像document、alert()、console.log()这些都是window的属性，基本BOM的属性和方法都是window的
- 所有通过var定义在全局作用域中的变量、函数都会变成window对象的属性和方法
- window对象下的属性和方法调用的时候可以省略window



1. JavaScript有几部分组成的？
   - ECMAScript 。 定义基本js语法
   - web APIs。（DOM、BOM）
2. BOM是什么？
   - 浏览器对象模型，定义了一套操作浏览器窗口的API
3. window对象是什么？ 平时可以省略吗？
   - 是一个全局对象，也可以说是JavaScript中的顶级对象
   - window对象下的属性和方法调用的时候可以省略window

BOM

- window对象
- 定时器-延时函数
- location对象
- navigator对象
- histroy对象
- 本地存储

- 目标：学习 window 对象的常见属性，知道各个 BOM 对象的功能含义

1.2 定时器-延迟函数

- JavaScript 内置的一个用来让代码延迟执行的函数，叫 setTimeout
- **语法：**

```
setTimeout(回调函数, 等待的毫秒数)
```

- setTimeout 仅仅只执行一次，所以可以理解为就是把一段代码延迟执行，平时省略window
- 间歇函数 setInterval ：每隔一段时间就执行一次， ，平时省略window
- **清除延时函数：**

```
let timer = setTimeout(回调函数, 等待的毫秒数)
clearTimeout(timer)
```

- **注意点**
  - 延时函数需要等待，所以后面的代码先执行
  - 返回值是一个正整数，表示定时器的编号

案例

5秒钟之后消失的广告

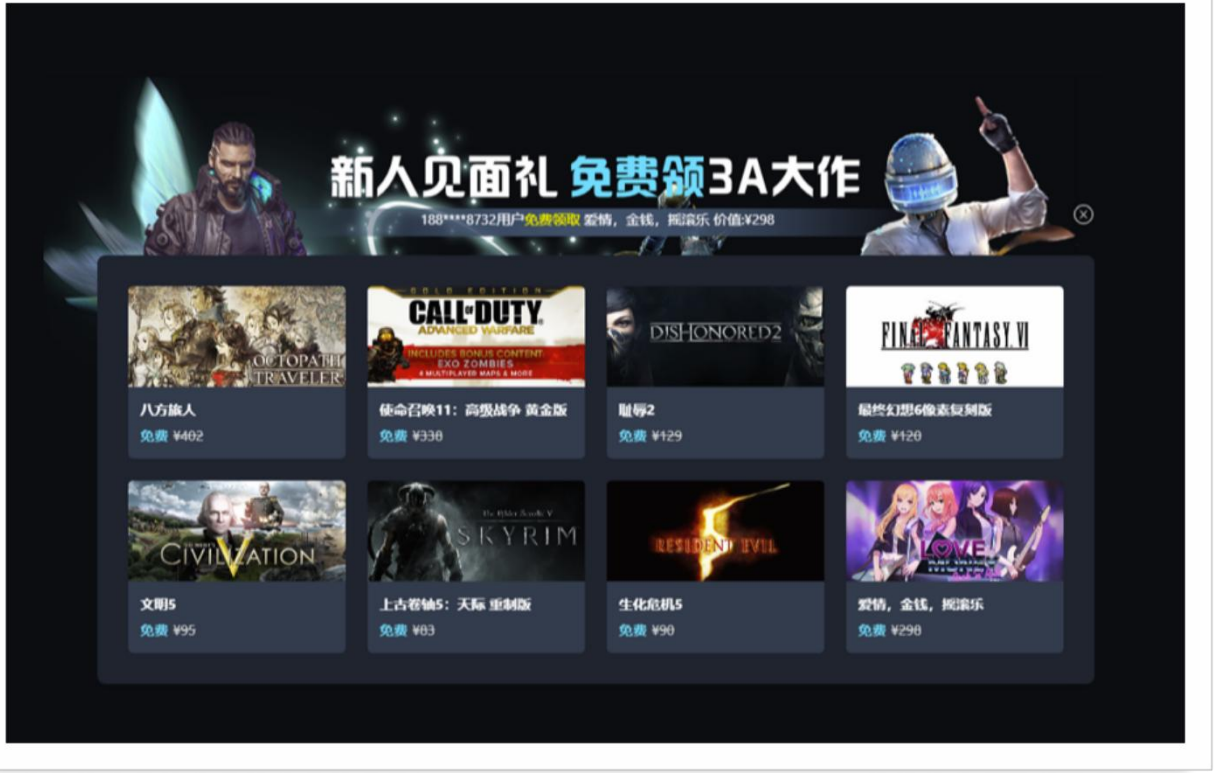
需求：5秒钟之后，广告自动消失

分析：

①：点击关闭按钮可以关闭

②：设置延迟函数时间为5秒

③：调用点击事件 click()

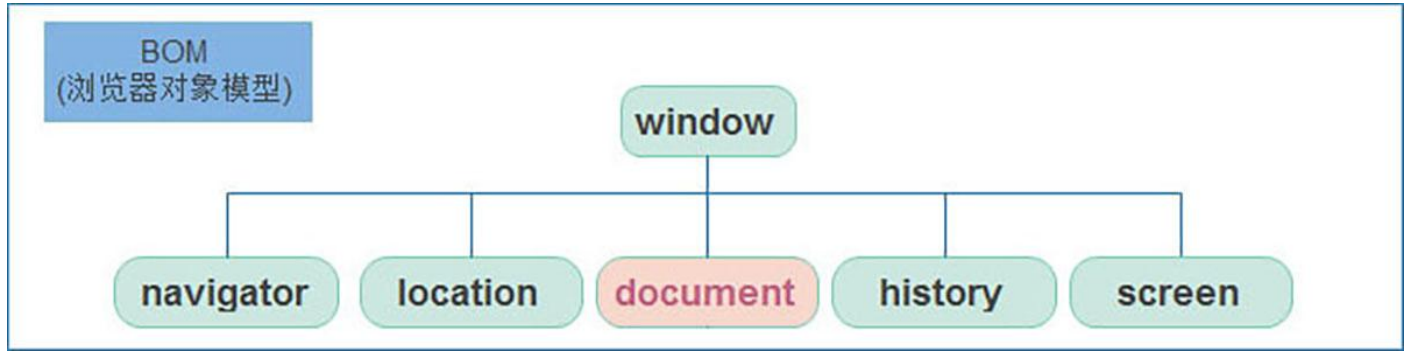


BOM

- window对象
- 定时器-延时函数
- location对象
- navigator对象
- histroy对象
- 本地存储

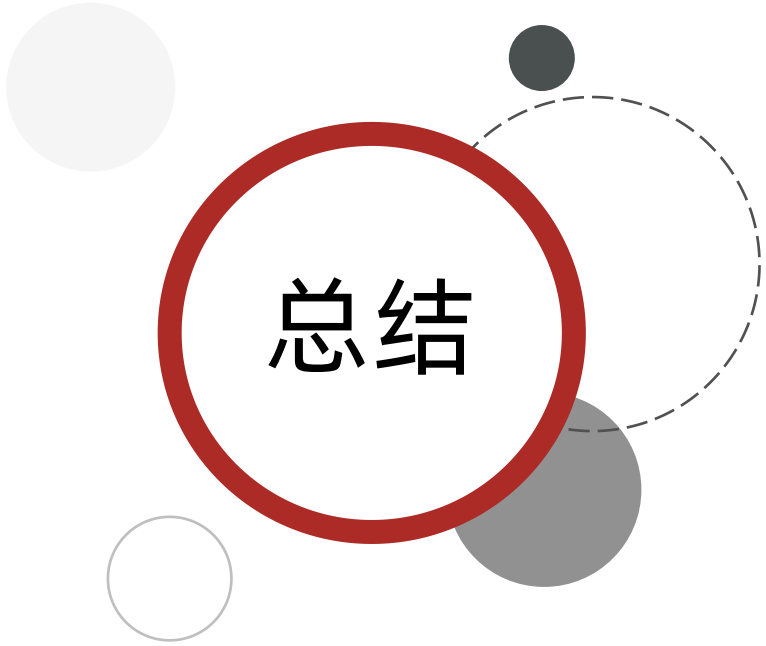
- 目标：学习 window 对象的常见属性，知道各个 BOM 对象的功能含义

1.3 location对象



- location（地址）它拆分并保存了 URL 地址的各个组成部分， 它是一个对象
- **常用属性和方法：**

属性/方法	说明
href	属性，获取完整的 URL 地址，赋值时用于地址的跳转
search	属性，获取地址中携带的参数，符号 ？ 后面部分
hash	属性，获取地址中的啥希值，符号 # 后面部分
reload()	方法，用来刷新当前页面，传入参数 true 时表示强制刷新



location 是对象，它拆分并保存了 URL 地址的各个组成部分

属性/方法	说明
href	属性，获取完整的 URL 地址，赋值时用于地址的跳转（重点）
search	属性，获取地址中携带的参数，符号 ？ 后面部分
hash	属性，获取地址中的啥希值，符号 # 后面部分
reload()	方法，用来刷新当前页面，传入参数 true 时表示强制刷新

案例

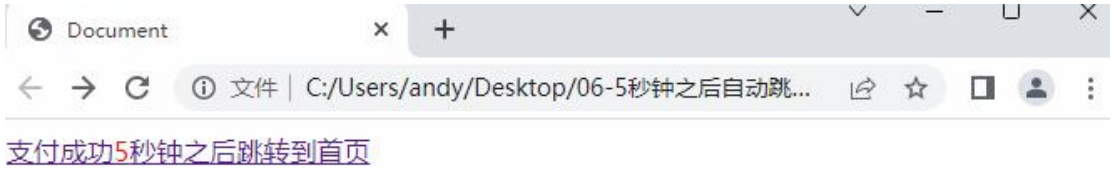
5秒钟之后跳转页面

需求：用户点击可以跳转，如果不点击，则5秒之后自动跳转，要求里面有秒数倒计时

分析：

①：利用定时器间歇函数设置链接里面的数字倒计时

②：时间到了，关闭定时器同时自动跳转到新的页面 （location.href）



BOM

- window对象
- 定时器-延时函数
- location对象
- navigator对象
- histroy对象
- 本地存储

- 目标：学习 window 对象的常见属性，知道各个 BOM 对象的功能含义

1.5 navigator对象

- navigator是对象，该对象下记录了浏览器自身的相关信息
- **常用属性和方法：**
  - 通过 userAgent 检测浏览器的版本及平台

```
// 检测 userAgent（浏览器信息）
(function () {
    const userAgent = navigator.userAgent
    // 验证是否为Android或iPhone
    const android = userAgent.match(/(Android);?[\s\/]+([\d.]+)?/)
    const iphone = userAgent.match(/(iPhone\sOS)\s([\d_]+)/)

    // 如果是Android或iPhone，则跳转至移动站点
    if (android || iphone) {
        location.href = 'http://m.itcast.cn'
    }
})();
```

BOM

- window对象
- 定时器-延时函数
- location对象
- navigator对象
- histroy对象
- 本地存储

- 目标：学习 window 对象的常见属性，知道各个 BOM 对象的功能含义

1.5 histroy对象

- history（历史)是对象，主要管理历史记录， 该对象与浏览器地址栏的操作相对应，如前进、后退等
- 使用场景

history 对象一般在实际开发中比较少用，但是会在一些 OA 办公系统中见到。

- 常见方法：

history对象方法	作用
back()	可以后退功能
forward()	前进功能
go(参数)	前进后退功能 参数如果是 1 前进1个页面 如果是-1 后退1个页面

BOM

- window对象
- 定时器-延时函数
- location对象
- navigator对象
- histroy对象
- 本地存储

- 目标：学习 window 对象的常见属性，知道各个 BOM 对象的功能含义

1.6 本地存储（今日重点）

本地存储：将数据存储在本地浏览器中

常见的使用场景：

https://todomvc.com/examples/vanilla-es6/　　页面刷新数据不丢失

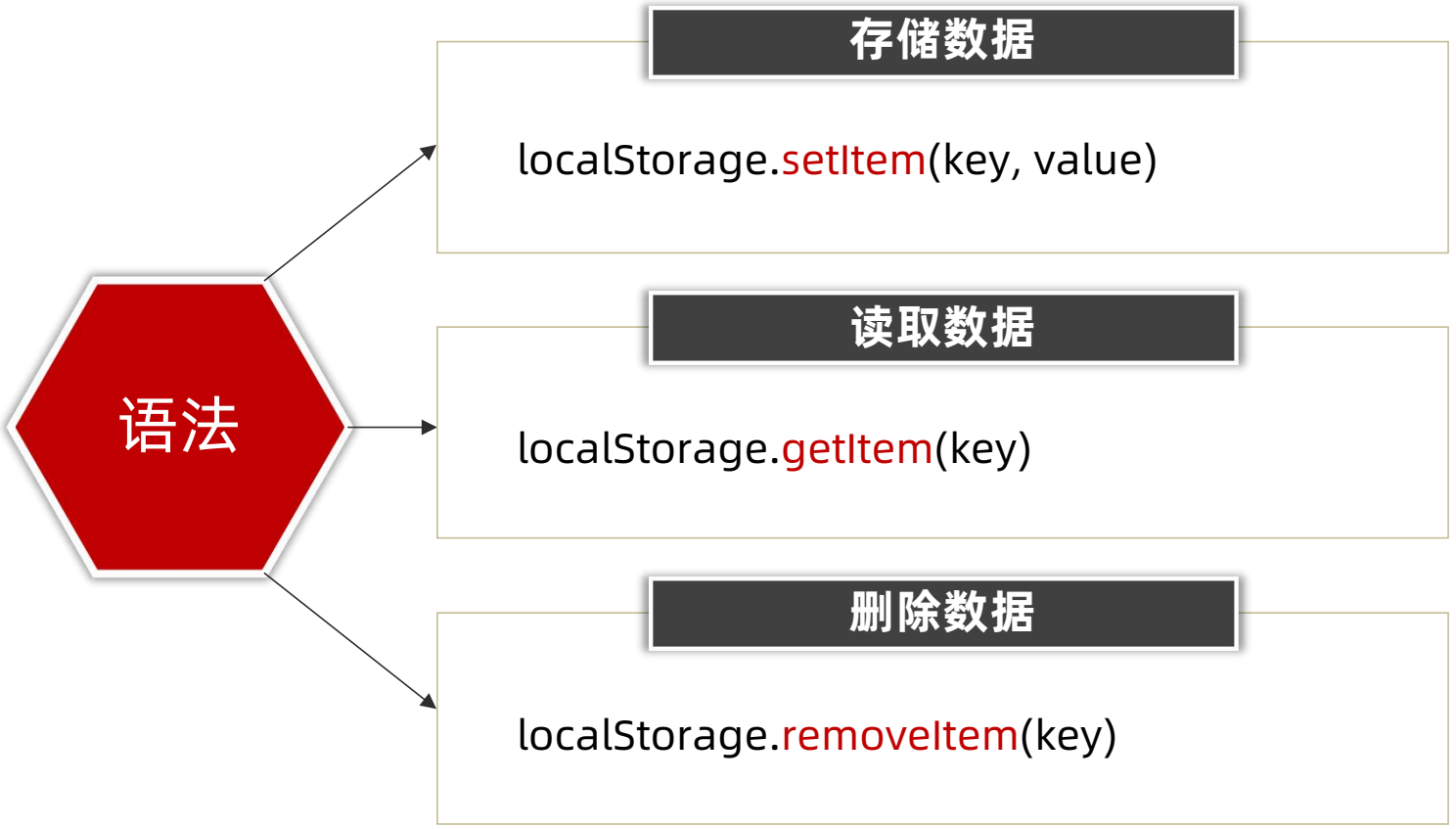
好处：

1、页面刷新或者关闭不丢失数据，实现数据持久化

2、容量较大，sessionStorage和 localStorage 约 5M 左右

1.6 本地存储分类- localStorage（重点）

- **作用：**数据可以长期保留在本地浏览器中，刷新页面和关闭页面，数据也不会丢失
- **特性：**以键值对的形式存储，并且存储的是字符串， 省略了window



1.6 本地存储分类- sessionStorage（了解）

- **特性：**
  - 用法跟localStorage 基本相同
  - 区别是：当页面浏览器被关闭时，存储在 sessionStorage 的数据会被清除

存储：sessionStorage.setItem(key, value)

获取：sessionStorage.getItem(key)

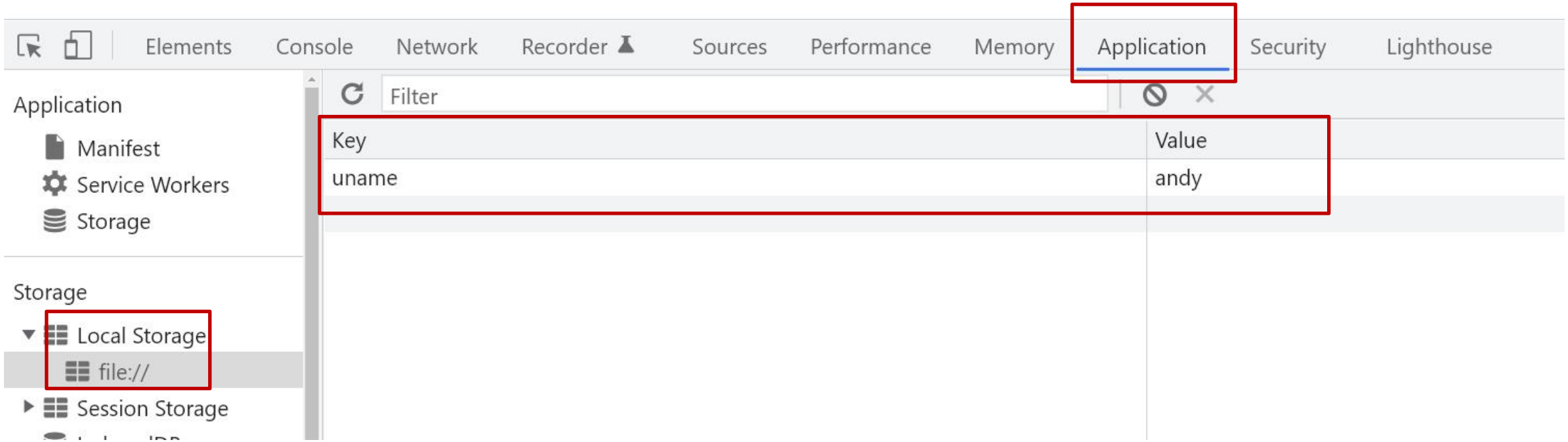
删除：sessionStorage.removeItem(key)

总结

1. localStorage 作用是什么？
   - 数据可以长期保留在本地浏览器中，刷新页面和关闭页面，数据也不会丢失
2. localStorage 存储，获取，删除的语法是什么？
   - 存储：localStorage.setItem(key, value)
   - 获取：localStorage.getItem(key)
   - 删除：localStorage.removeItem(key)

1.6 本地存储分类- localStorage

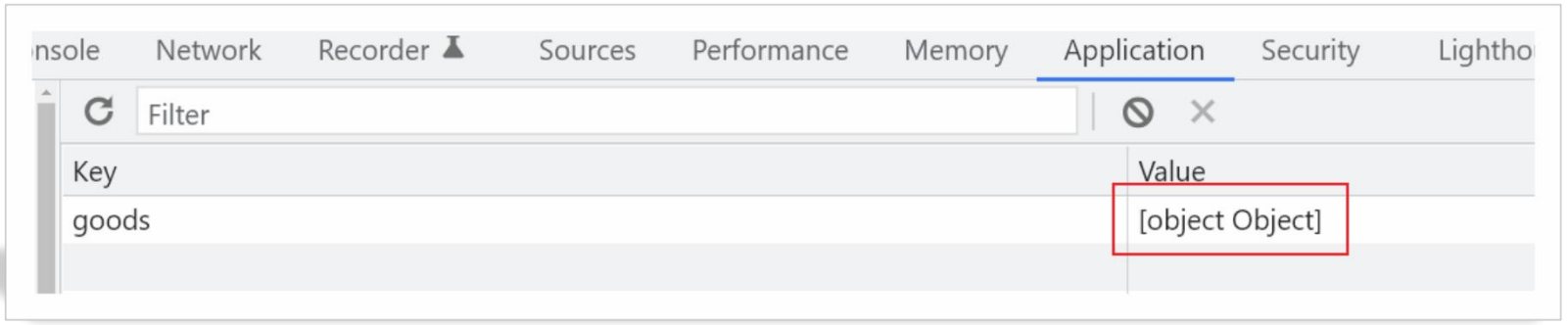
- 浏览器查看本地数据：



1.6 localStorage 存储复杂数据类型

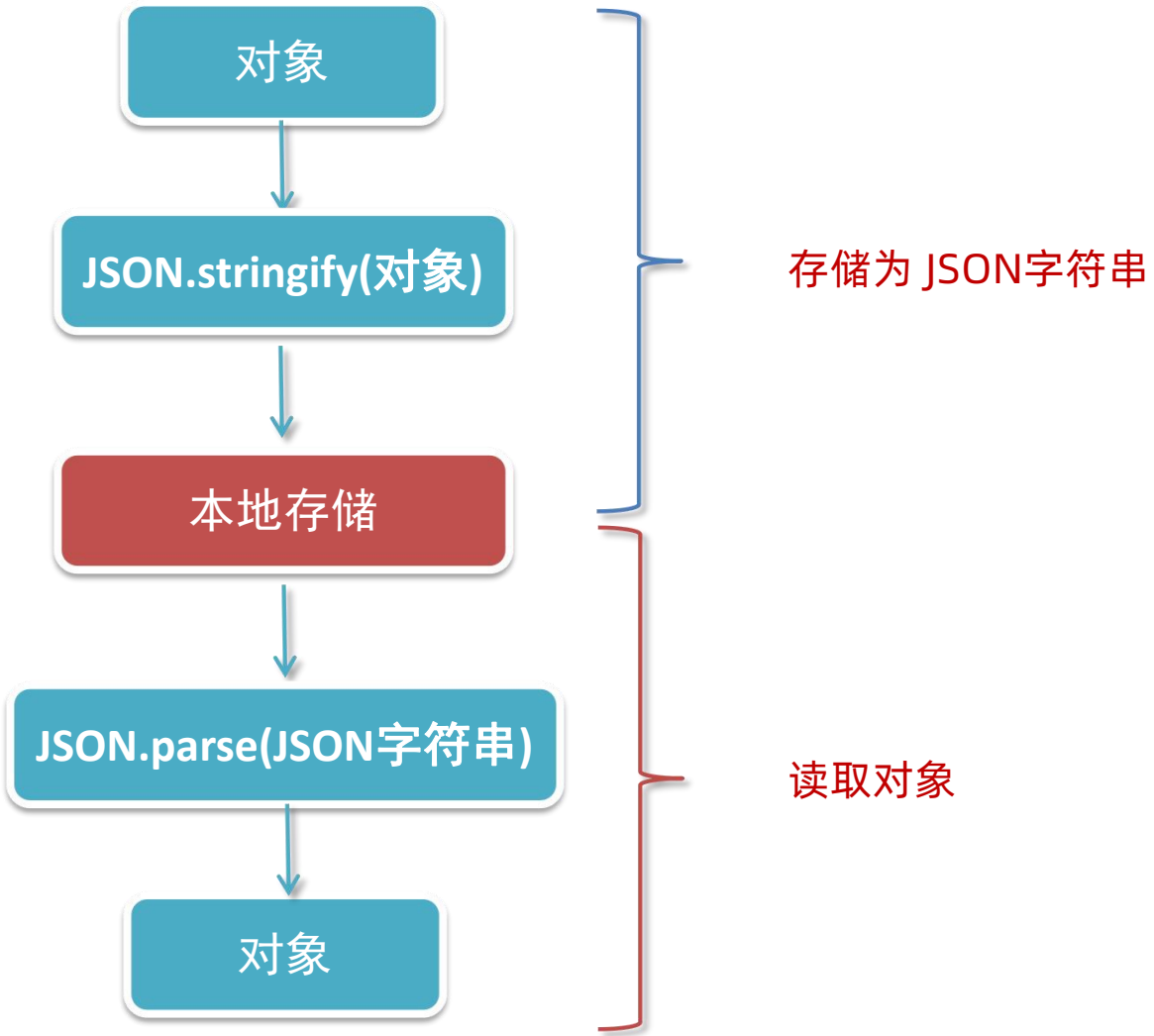
- **问题：**本地只能存储字符串，无法存储复杂数据类型.

```
const goods = {
  name: '小米10',
  price: 1999
}
localStorage.setItem('goods', goods)
```



1.6 本地存储分类- localStorage

- 解决方案：



1.6 本地存储分类- localStorage

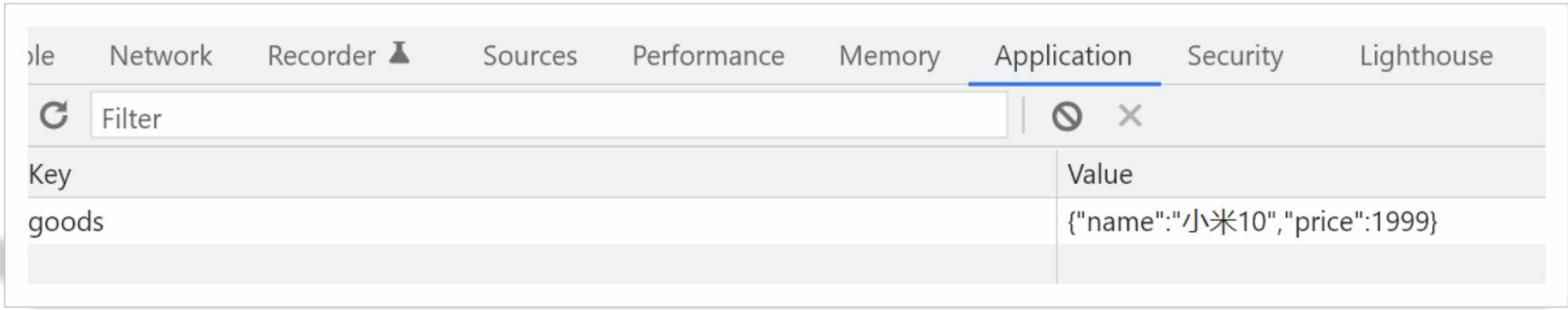
- **解决：**需要将复杂数据类型转换成 JSON字符串,在存储到本地
- **语法：**JSON.stringify(复杂数据类型)

```
const goods = {
  name: '小米10',
  price: 1999
}
localStorage.setItem('goods', JSON.stringify(goods))
```

JSON字符串：

- 首先是1个字符串
- 属性名使用双引号引起来，不能单引号
- 属性值如果是字符串型也必须双引号

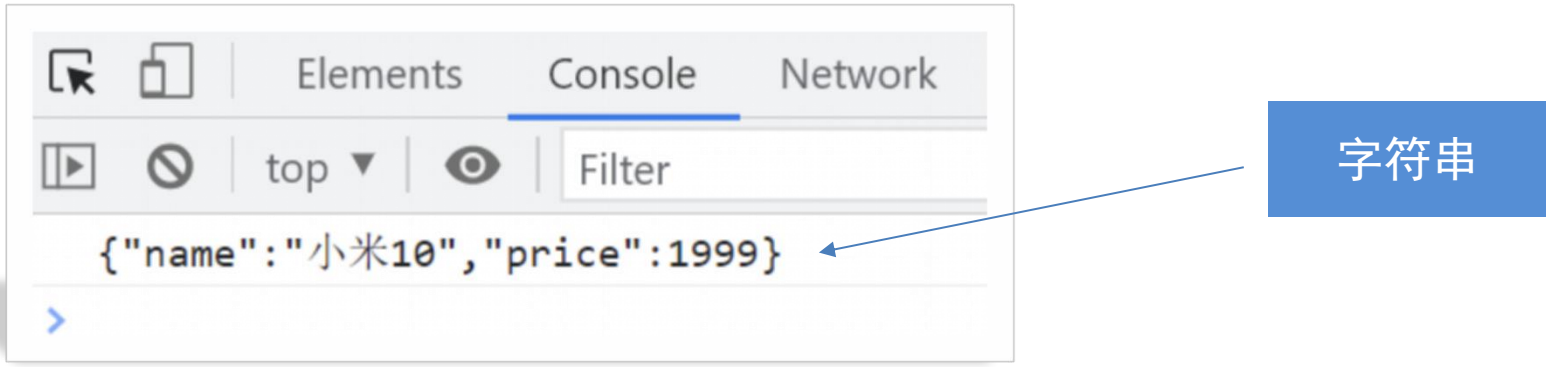
- 将复杂数据转换成JSON字符串　　**存储** 本地存储中



1.6 本地存储分类- localStorage

- **问题：**因为本地存储里面取出来的是字符串，不是对象，无法直接使用

```
const obj = localStorage.getItem('goods')
console.log(obj)
```

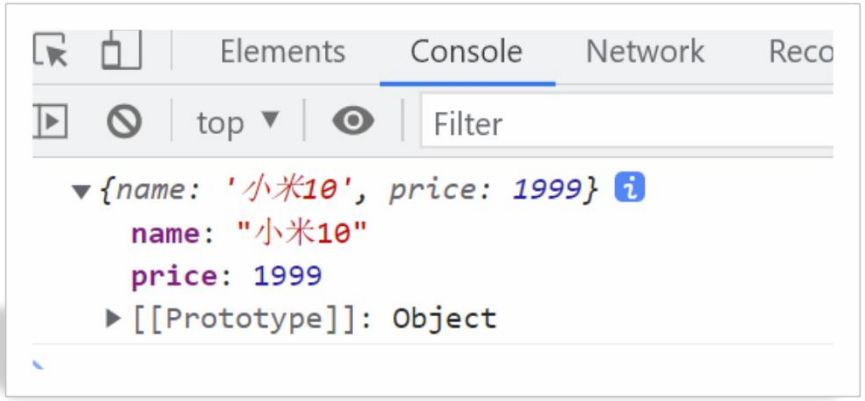


1.6 本地存储分类- localStorage

- **解决：**把取出来的字符串转换为对象
- **语法：**JSON.parse(JSON字符串)

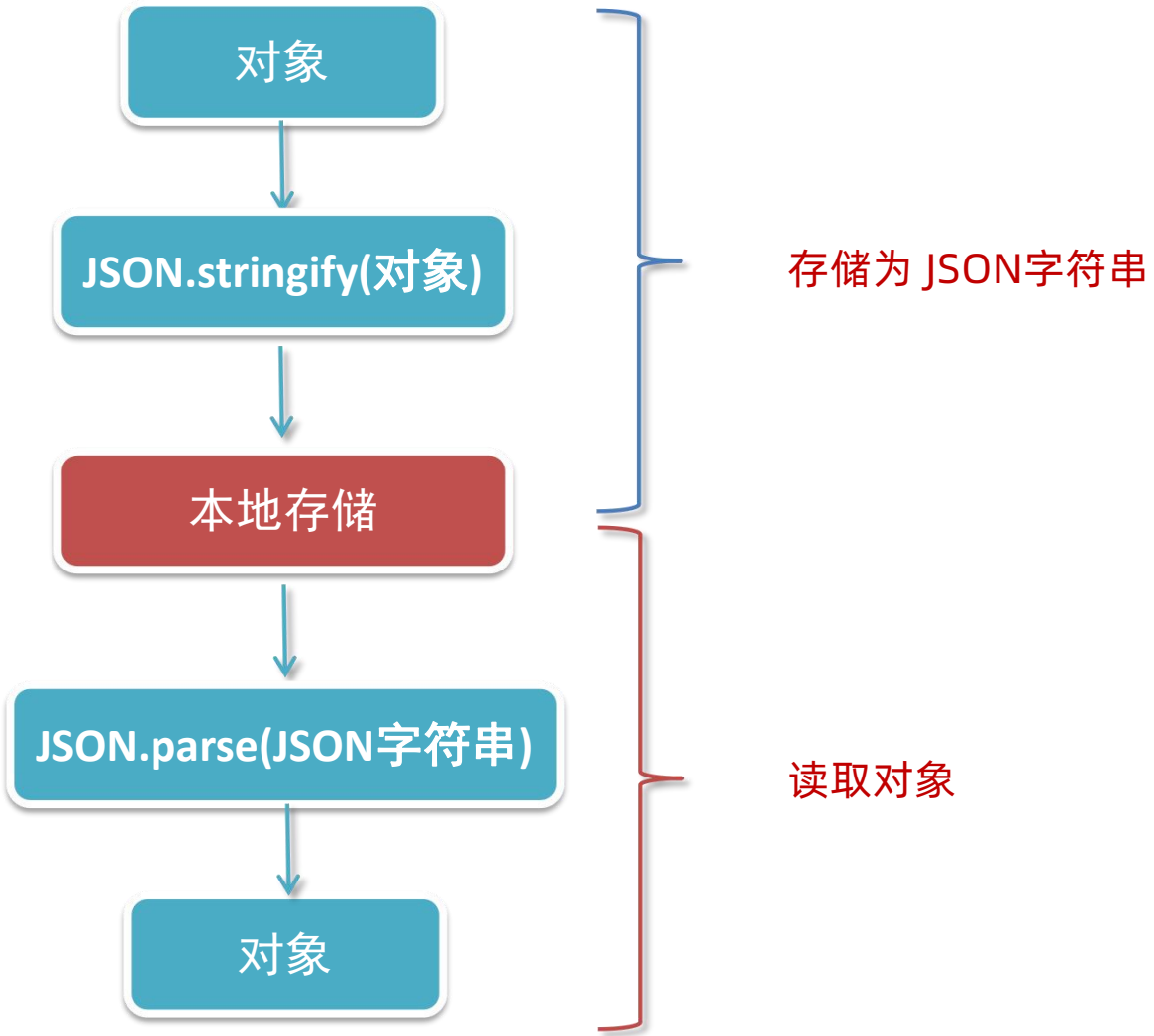
```
const obj = JSON.parse(localStorage.getItem('goods'))
console.log(obj)
```

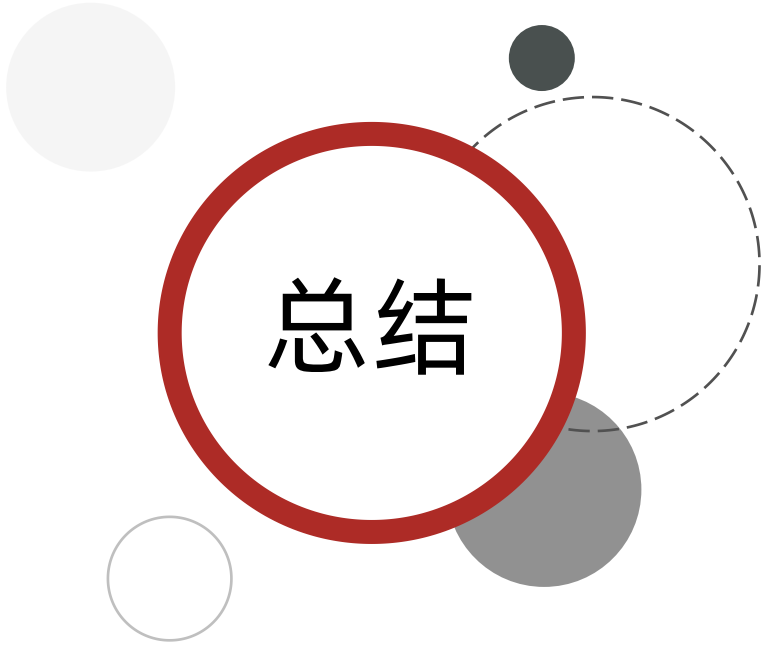
➢ 将JSON字符串转换成对象　　**取出** 时候使用



1.6 本地存储分类- localStorage

- 解决方案：





1. localStorage 如何存储复杂数据类型？

   - 把对象转换为JSON字符串。JSON.stringify(复杂数据类型)

```
const goods = {
  name: '小米10',
  price: 1999
}
localStorage.setItem('goods', JSON.stringify(goods))
```

2. localStorage 如何读取复杂数据类型？

   - 把JSON字符串转换为对象。JSON.parse()

```
const obj = JSON.parse(localStorage.getItem('goods'))
console.log(obj)
```

目录

Contents

案例

学生就业统计表

需求： 录入学生信息，页面刷新数据不丢失

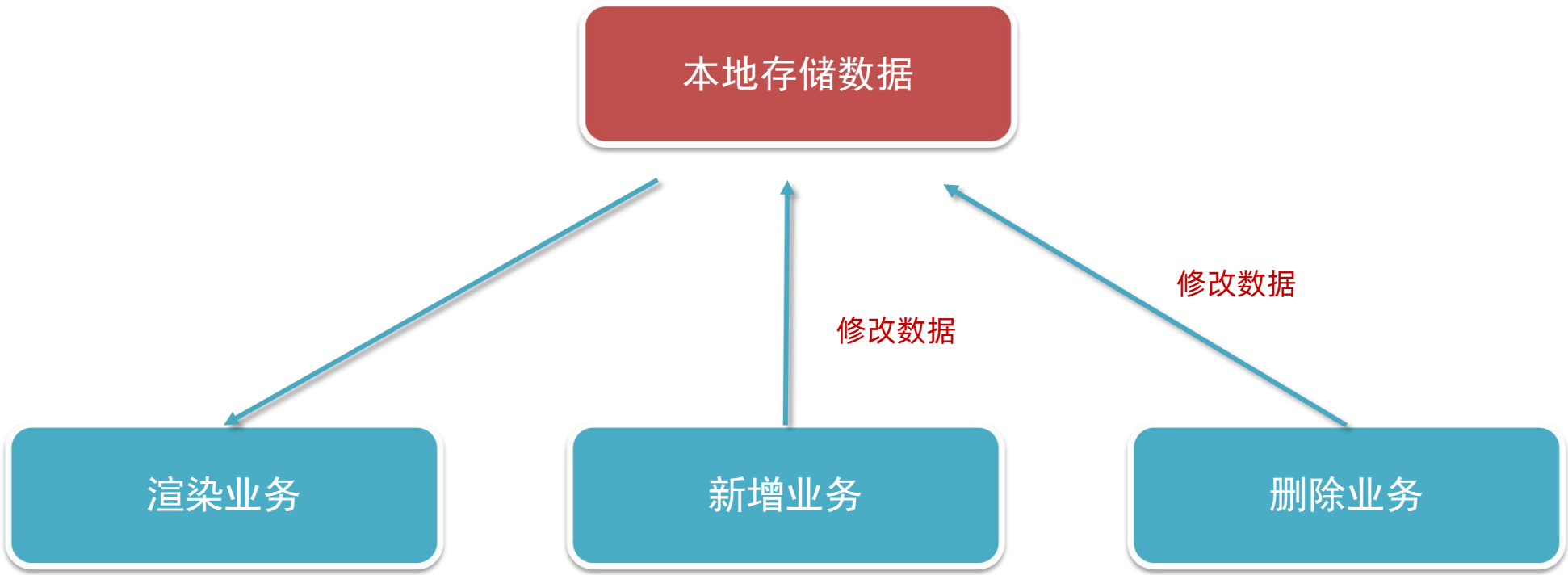
步骤

学生就业统计表

- 渲染业务
- 新增业务
- 删除业务

步骤

学生就业统计表



步骤

学生就业统计表

- 渲染业务
- 新增业务
- 删除业务

案例 学生就业信息表

渲染业务

```
// 参考数据
const initData = [
  {
    stuId: 1,
    uname: '迪丽热巴',
    age: 22,
    gender: '女',
    salary: '12000',
    city: '北京',
    time: '2099/9/9 08:08:08'
  }
]
```

根据持久化数据渲染页面

核心步骤：

①：读取 localstorage 本地数据

- 如果有数据则转换为对象放到变量里面一会使用它渲染页面
- 如果没有则用默认空数组 []
- 为了测试效果，咱们可以先把initData 存入本地存储看效果

案例 学生就业信息表

```
// 参考数据
const initData = [
  {
    stuId: 1,
    uname: '迪丽热巴',
    age: 22,
    gender: '女',
    salary: '12000',
    city: '北京',
    time: '2099/9/9 08:08:08'
  }
]
```

根据持久化数据渲染页面

核心步骤：

②： 根据数据渲染页面。遍历数组， 根据数据生成 tr，里面填充数据， 最后追加给 tbody

ID	姓名	年龄	性别	薪资	就业城市	录入时间	操作
1	迪丽热巴	23	女	12000	北京	2023/2/2 11:33:09	删除

```
<tbody>
  <tr>
    <td>1</td>
    <td>迪丽热巴</td>
    <td>23</td>
    <td>女</td>
    <td>12000</td>
    <td>北京</td>
    <td>2099/9/9 08:08:08</td>
    <td>
      <a href="javascript:">
        <i class="iconfont icon-shanchu"></i>
        删除
      </a>
    </td>
  </tr>
</tbody>
```

字符串拼接新思路：(效果更高，开发常用的写法)

- 利用 map() 和 join() 数组方法实现字符串拼接

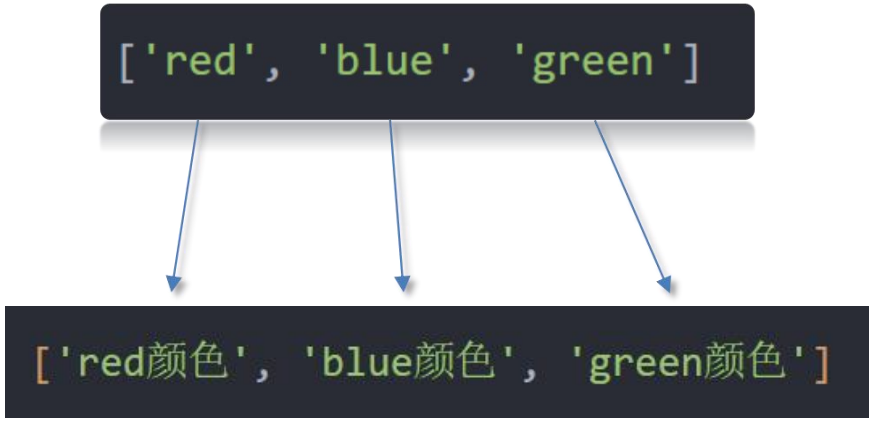
数组中map方法 迭代数组

```
map([🐮, 🍗, 🐔, 🌽], cook)
=> [🍔, 🍟, 🍗, 🍿]
```

- 使用场景：

map 可以遍历数组处理数据，并且**返回新的数组**

```
const arr = ['red', 'blue', 'green']
const newArr = arr.map(function (ele, index) {
  console.log(ele)  // 数组元素
  console.log(index) // 数组索引号
  return ele + '颜色'
})
console.log(newArr)   //  ['red颜色', 'blue颜色', 'green颜色']
```



map 也称为映射。映射是个术语，指两个元素的集之间元素相互“对应”的关系。

map重点在于有返回值，forEach没有返回值（undefined）

数组中join方法

- 作用：

  join() 方法用于把数组中的所有元素转换一个字符串

- 语法：

```
const arr = ['red颜色', 'blue颜色', 'green颜色']
console.log(arr.join(''))  // red颜色blue颜色green颜色
```

- 参数：

  数组元素是通过参数里面指定的分隔符进行分隔的，空字符串（''），则所有元素之间都没有任何字符。

数组中map + join 方法渲染页面思路：

ID	姓名	年龄	性别	薪资	就业城市	录入时间	操作
1	迪丽热巴	23	女	12000	北京	2023/2/2 11:33:09	删除

map遍历数组处理数据生成tr，返回一个数组

把数组转换为字符串

追加给tbody

```
[
<tr>
  <td>1</td>
  <td>pink</td>
  <td>18</td>
  <td>男</td>
  <td>10000</td>
  <td>北京</td>
  <td>…</td>
</tr>
<tr>
  <td>2</td>
  <td>red</td>
  <td>19</td>
  <td>男</td>
  <td>20000</td>
  <td>北京</td>
  <td>…</td>
</tr>
```

→ join('') →

```
<tr>
  <td>1</td>
  <td>pink</td>
  <td>18</td>
  <td>男</td>
  <td>10000</td>
  <td>北京</td>
  <td>…</td>
</tr>
<tr>
  <td>2</td>
  <td>red</td>
  <td>19</td>
  <td>男</td>
  <td>20000</td>
  <td>北京</td>
  <td>…</td>
</tr>
```

→

```
<tbody>
  <tr>
    <td>1</td>
    <td>pink</td>
    <td>18</td>
    <td>男</td>
    <td>10000</td>
    <td>北京</td>
    <td>…</td>
  </tr>
  <tr>
    <td>2</td>
    <td>red</td>
    <td>19</td>
    <td>男</td>
    <td>20000</td>
    <td>北京</td>
    <td>…</td>
  </tr>
</tbody>
```

```
// 参考数据
const initData = [
  {
    stuId: 1,
    uname: '迪丽热巴',
    age: 22,
    gender: '女',
    salary: '12000',
    city: '北京',
    time: '2099/9/9 08:08:08'
  }
]
```

案例 学生就业统计表

渲染业务

根据持久化数据渲染页面

核心步骤：

②： 根据数据渲染页面。遍历数组， 根据数据生成 tr，里面填充数据， 最后追加给 tbody

1. 渲染业务要封装成一个函数 render

2. 我们使用map方法遍历数组，里面更换数据，然后会返回 有数据的 tr 数组

3. 通过 join 方法把map返回的数组转换为字符串

4. 把字符串通过 innerHTML 赋值给 tbody

ID	姓名	年龄	性别	薪资	就业城市	录入时间	操作
1	迪丽热巴	23	女	12000	北京	2023/2/2 11:33:09	删除

步骤

学生就业统计表

- 渲染业务
- 新增业务
- 删除业务

步骤

学生就业统计表

新增业务

点击新增按钮，页面显示新的数据

核心步骤：

①：给form注册提交事件，要阻止默认提交事件（阻止默认行为）

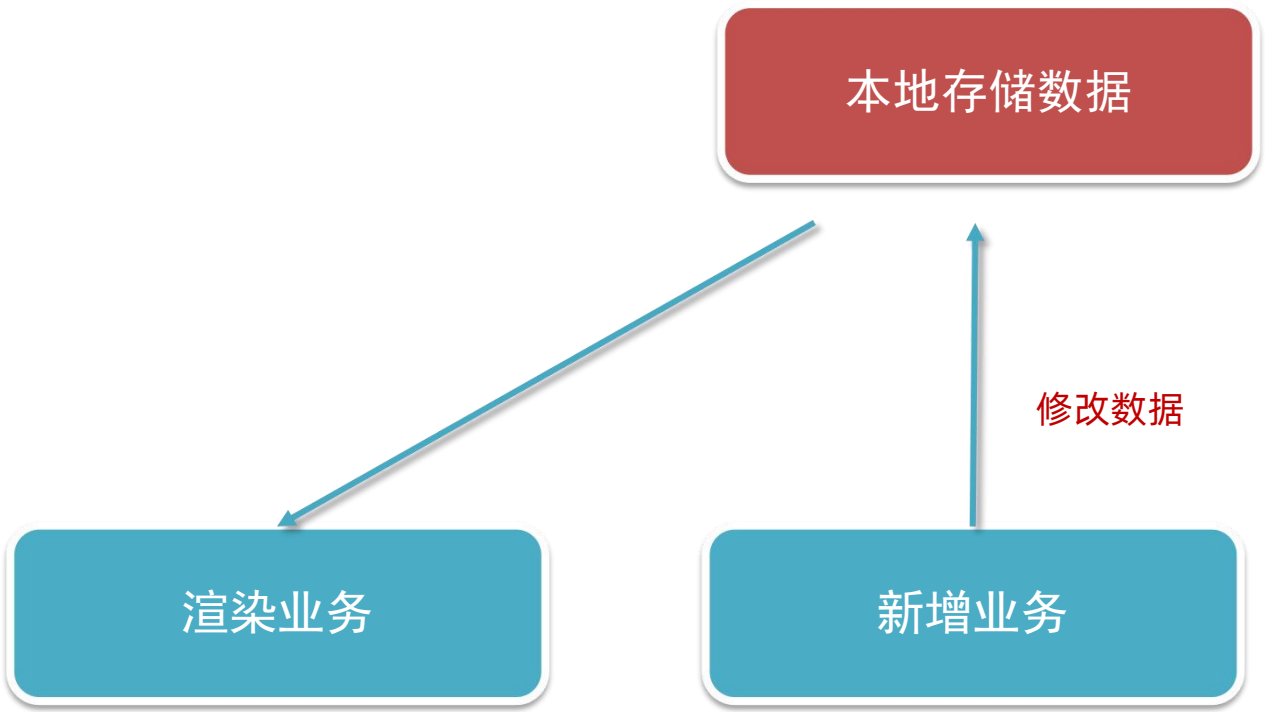
```
事件对象.preventDefault()  // 阻止默认行为
```

②：非空判断

如果年龄、性别、薪资有一个值为空，则 return 返回 '输入不能为空' 中断程序

步骤

学生就业统计表



步骤

学生就业统计表

新增业务

```
{
  stuId: 1,
  uname: '迪丽热巴',
  age: 22,
  salary: '12000',
  gender: '女',
  city: '北京',
  time: '2099/9/9 08:08:08'
}
```

点击新增按钮，页面显示新的数据

核心步骤：

③：给 arr 数组追加对象，里面存储表单获取过来的数据，格式如左图：

④：利用本地存储最新数据渲染页面和重置表单（reset()方法）

⑤：把数组数据存储到本地存储里面， 利用 JSON.stringify() 存储为JSON字符串

步骤

学生就业统计表

- 渲染业务
- 新增业务
- 删除业务

步骤

学生就业统计表

删除业务

点击删除按钮，可以删除对应的数据

核心步骤：

①：采用事件委托形式，给 tbody 注册点击事件

②：得到当前点击的索引号。渲染数据的时候，动态给a链接添加自定义属性 data-

步骤

学生就业统计表

删除业务

点击删除按钮，可以删除对应的数据

核心步骤：

①：采用事件委托形式，给 tbody 注册点击事件

②：得到当前点击的索引号。渲染数据的时候，动态给a链接添加自定义属性 data-id=“0” ③：根据索引号，利用 splice 删除数组这条数据

④：重新渲染页面

⑤：把最新 arr 数组存入本地存储

步骤

学生就业统计表

关于stuId 的处理

核心思路：

①：新增加序号应该是最后一条数据的stuId + 1

- 数组[数组的长度-1].stuId + 1

②：但是要判断， 如果没有数据则是直接赋值为1，否则就采用上面的做法

共有数据 1 条

ID	姓名	年龄	性别	薪资	就业城市	录入时间	操作
2	佟丽丫丫	24	男	12000	北京	2023/2/3 11:07:08	删除

传智教育旗下高端IT教育品牌