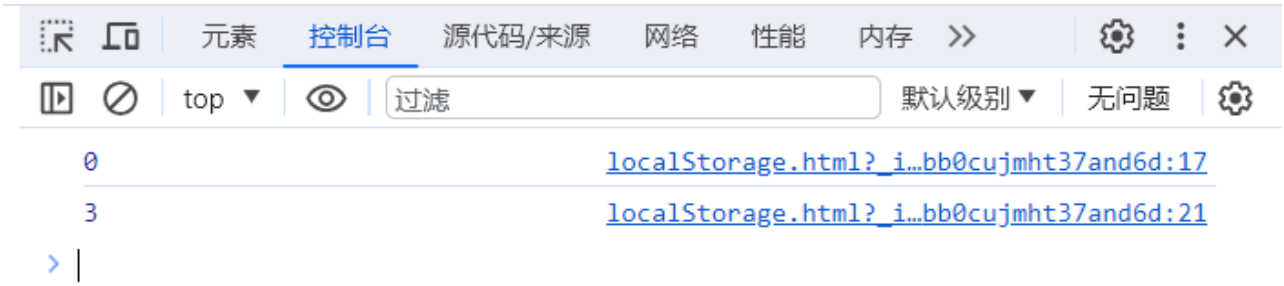
1. 本地储存（localStorage）
    - Cookie: 有容量限制且值只能是字符串类型
    - Storage: 分为会话级别和本地级别，键值对类型，值只能是字符串类型
        - setItem("key","value"): 添加或替换键值对（不存在则添加）
        - getItem(key): 返回特定键的**值**
        - 会话级别: 数据存在于**页面浏览时**，但**关闭页面后消失**
        - 本地级别: 数据可以在**不同页面间共享**，即使页面关闭后再次访问仍可以取出数据
    - Indexed DB: 基于事务的本地存储方式
- 功能演示

新建页面

输入以下代码：

```
console.log(localStorage.length);
localStorage.setItem("name","Alice");
localStorage.setItem("age","20");
localStorage.setItem("tall","165.5");
console.log(localStorage.length);
```

当我们打开控制台，会发现控制台输入如下图：



localStorage的数据长度为0，但我们向其写入了三条数据，因此当前长度为3。即使刷新页面，控制台两个都变为输出3，因为数据已存储在本地，除非手动清除，否则将一直存在。（如下图）

- 遍历数组的键，并取键值

在此实例中，如果我们并不知道localStorage所对应的键名，我们可以通过key[索引]来取键名，并通过**getItem方法**来取对应的键值。

思路：通过for循环遍历，长度为localStorage.length，k的末位索引为localStorage.length-1。

代码如下：

```
for(let i=0 , len=localStorage.length;i<len;i++){
    let k=localStorage.key(i);    //遍历键，并赋给k
    let v=localStorage.getItem(k);    //通过键名取键值
    console.log(k+":"+v);            //打印“键名：键值”
}
```

- 演示实例：完善register页面的保存按钮功能

我们自Chapter2、3中完成注册页面后，保存按钮并未生效，下面对保存按钮进行完善。

首先，捕获按钮，通过按钮的name值来捕获按钮。

代码如下：

```
let save_btn = form["save"];
```

之后，通过监听save按钮的click（被点击）事件，会弹出一个确认框，询问用户是否确认保存。如果用户点击了取消按钮，则**函数会立即返回**，不会执行后续的代码。如果用户点击了确认按钮，则会获取四个文本框的值，**并将这些值分别存储到本地存储（localStorage）**中，对应的**键名分别为"user_name"、"pwd"、"pwd_confirm"和"email"**。代码如下：

```
//将四个文本框内容写到本地存储中
save_btn.addEventListener("click",function(){
    //提示用户是否需要保存
    if(!confirm("你确认要保存吗？")){
        return;
    }

    let user_name=user_name_field.value;
    let pwd=pwd_field.value;
    let pwd_confirm=pwd_confirm_field.value;
    let email=email_field.value;
```

```
12
13      localStorage.setItem("user_name",user_name);
14      localStorage.setItem("pwd",pwd);
15      localStorage.setItem("pwd_confirm",pwd_confirm);
16      localStorage.setItem("email",email);
17  })
```

2. 两种Map的比较（Java中的Map可见笔记 Java-Chapter17(2023.6.14)）

Java中的Map

- Map接口: 表示键值对映射关系，包括键值对的增删改查等操作
- 实现类: HashMap、TreeMap、LinkedHashMap等
- 常用方法: put(), get(), remove(), containsKey(), containsValue(), size()

JavaScript中的Map

- Map对象: **保存键值对**，键可以是任何数据类型，记住键的**原始插入顺序**
- 提供映射大小的属性

3. 实现在关闭网页窗口前对注册页面框内数据实现自动保存

事件触发

- 页面加载完后会触发load事件
- 关闭浏览器时会触发window对象的beforeunload和unload事件
- 修改storage数据不会引发storage事件，但在同源网站的其它页面会引发storage事件
- 演示实例：在注册页面关闭前自动保存框内数据

添加代码如下：

```
1  //判断输入框是否有值，有则在关闭窗口前自动保存，没有则不写
2  window.addEventListener("beforeunload",(event)=>{
3      let user_name=user_name_field.value;
4      let pwd=pwd_field.value;
5      let pwd_confirm=pwd_confirm_field.value;
6      let email=email_field.value;
7
8      let ls=localStorage;
9
10     //分四个框判断框内是否有值，有则保存，否则不存储
11     if(user_name_field){
12         ls.setItem("user_name",user_name);
13     }
14     if(pwd_field){
```

```
        ls.setItem("pwd",pwd);
    }
    if(pwd_confirm_field){
        ls.setItem("pwd_confirm",pwd_confirm);
    }
    if(email_field){
        ls.setItem("email",email);
    }

})
```

这里代码，如果框内有值，则user_name_field、pwd_field、pwd_confirm_field和email_field为非0，则为true，执行if后的setItem语句。如果框内为空，则为false，不会执行语句。

- storage数据的修改

在A页面修改storage数据，本身不会引发storage事件，但在同源网站的**其它页面**会引发**storage事件**

演示实例：

首先我们创建一个another.html页面，附代码：

```
//创建event事件监听修改storage打印修改的键名，旧值和新值
window.addEventListener("storage",function(event){
    console.log(event.key,event.oldValue,event.newValue);
});
```

并在localStorage页面中添加修改按钮

```
<input type="button" value="修改">
```

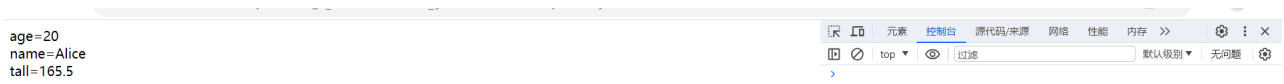
为按钮附上修改localStorage的事件

```
<script>
    let btn=document.querySelector("input");
    btn.addEventListener("click",(event)=>{
        localStorage.setItem("name","Bob");
    })
```

```
6
7       //创建localStorage对象
8       // let localStorage=window.localStorage;
9       console.log(localStorage.length);
10      localStorage.setItem("name","Alice");
11      localStorage.setItem("age","20");
12      localStorage.setItem("tall","165.5");
13      console.log(localStorage.length);
14      for(let i=0 , len=localStorage.length;i<len;i++){
15          let k=localStorage.key(i);
16          let v=localStorage.getItem(k);
17          console.log(k+":"+v);
18      }
19  </script>
```

这个实例中，当我们没有按按钮前，localStorage中的键名为name的值为Alice，当点击按钮，值改为Bob，此时another.html中的控制台中会打印修改的键名，旧值和新值。

未单击按钮前：



单击“修改”按钮后：

刷新后页面变为

```
age=20
name=Bob
tall=165.5
```

4. Socket、ServerSocket和Websocket

1. Socket：Socket是在网络中进行通信的**两个端点**之间的一种**抽象**。它包含了**IP地址和端口号**，通过它们可以实现**不同计算机上的进程之间的通信**。通常，Socket是**在客户端和服务器**之间建立的连接的一部分，允许它们**相互发送和接收数据**。在Java中，可以使用Socket类创建客户端套接字。

2. ServerSocket：ServerSocket是Java中用于创建**服务器端**的类。它通过在**特定的IP地址和端口**上**监听**来自客户端的连接**请求**来实现。一旦有客户端发起连接，ServerSocket将创建一个新的Socket实例来处理通信。ServerSocket在内部会**维护一个连接队列**，以处理传入的连接请求。它按照**先来先服务**的原则来处理队列中的连接请求。

3. Websocket：Websocket是一种在**Web浏览器和服务器**之间进行**全双工通信**的通信协议。它允许在单个TCP连接上进行**双向通信**。Websocket与HTTP不同，因为在建立连接之后，客户端和服务器均可**通过发送数据来触发事件**。在Websocket中，客户端和服务器之间可以**传输文本和二进制数据**。

请求IP和监听端口号通常是在创建Socket或ServerSocket时指定的。IP可以是服务器的IP地址，用于标识特定的计算机或服务器，端口号则用于标识特定的应用程序或服务。客户端将使用此IP和端口号连接到服务器，而服务器将在此IP和端口号上监听传入的连接请求。

- 实例演示：模拟简单的选课系统

首先制作一个简单的选课单选框，3个课程。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Course</title>
</head>
<body>
        <form name="course" action="process-course">
            <select name="sel">
                <option>Chinese</option>
                <option>English</option>
                <option>Math</option>
            </select>
            <input type="submit" value="选课">
```

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

此处有三个课程，为Chinese，English和Math，当单击选课按钮，触发submit事件，并将选取的课程发送GET请求到process-course

创建的Servlet类：ProcessCourse

```
package com.example.dispatch;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ProcessCourse" ,urlPatterns = "/process-course")
public class ProcessCourse extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String course = request.getParameter("sel");
        System.out.println(course);
        ServletContext context = request.getServletContext();
        switch (course){
            case "English":
                context.getRequestDispatcher("/process-english").forward(request, response);
                break;
            case "Math":
                context.getRequestDispatcher("/process-math").forward(request, response);
```

```
                break;
            case "Chinese":
                context.getRequestDispatcher("/process-chinese").forward(request, response);
                break;
            default:
                context.getRequestDispatcher("/no-process").forward(request, response);
            }
        }
    }


```

在这里，根据请求中传递的参数，选择不同的处理逻辑，然后通过Servlet上下文的dispatchRequest()方法将请求转发给对应课程的处理程序。最后，将结果返回给客户端。
我们简单的写出针对于Chinese课程的Servlet处理类ProcessChinese：

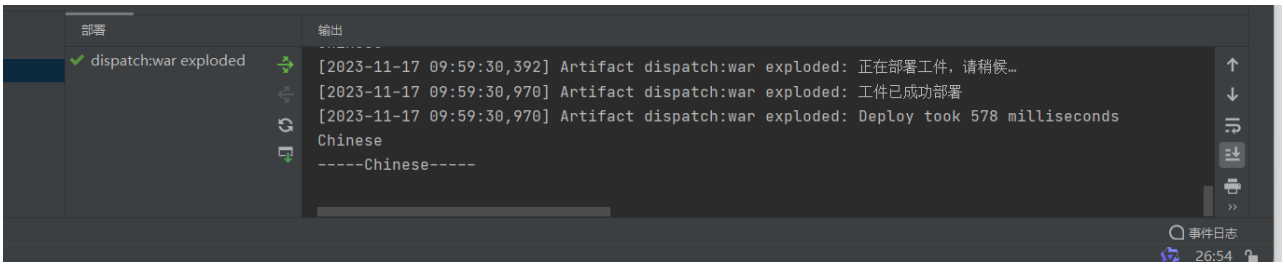
```
package com.example.dispatch;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ProcessChinese" , value = "/process-chinese")
public class ProcessChinese extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("-----Chinese-----");
        request.getRequestDispatcher("/WEB-INF/chinese.jsp").forward(request, response);
```

```
    }
}

```

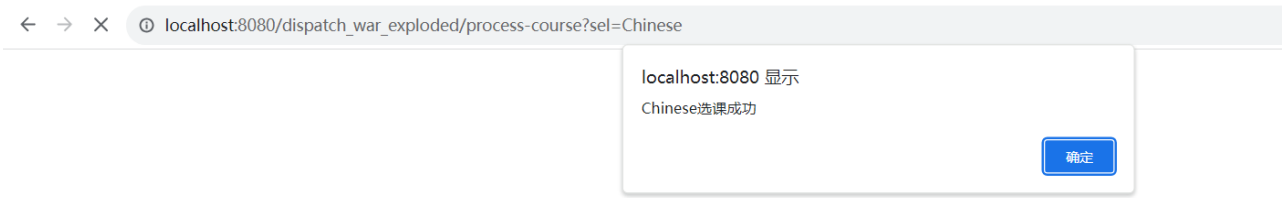
这里我们还需要在webapp文件夹下的WEB-INF文件夹下新建一个chinese.jsp，用来弹窗提示用户选课成功并返回选课页面
代码为：

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2023/11/17
  Time: 21:52
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<%--弹窗，选课Chinese成功--%>
<script type="text/javascript">
    alert("Chinese选课成功");
    window.location.href="course.html";
</script>
<%--跳转到页面--%>
</body>
</html>

```

第16行的意思是告知用户Chinese已经选课成功，第17行的意思是返回选课界面。
附效果图：
IDEA的Tomcat控制台会提示：



浏览器弹窗提示：



单击确定后（返回选课界面）：