后续安排:

1. Koa框架
2. 实时通信Socket.IO

总结:

1. Cookie和Session
    - 概念：Cookie客户端、服务端存储少量数据方式

      创建：res.cookie(key, value, opts)

      * *opts: {maxAge: 3600*1000, httpOnly: true, signed: true}

      访问：req.cookies.xx

      req.signedCookies.xx
    - 概念：Session服务端存储数据方式，
    - session存储：内存、数据库
    - 使用session：req.session.xx
2. Angular
    - 路由传参
        - 必选参数

          配置路由需要占位符： {path:'course/:id'}

          传递：

          router.navigate(['course', 1])

          拿参：

          ```
          constructor(private route:ActivatedRoute){
              route.paramMap.subscribe((pm:ParamMap)=>{
                  pm.get('id')
              })
          }
          ```
        - 可选参数

          传递：

          router.navigate(['course', 1, {foo:'foo'}])

          拿参：同上
        - 查询参数

          传递：

          router.navigate(['course', 1], {queryParams: {foo:'foo'}})

          拿参：

```
constructor(private route:ActivatedRoute){
    route.queryParamMap.subscribe((pm:ParamMap)=>{
        pm.get('foo')
    })
}
```

自定义组件使用

1. 创建组件：ng c compName
2. 传值：通过@Input()

   组件中：

   ```
   @Input()
   data: MenuItem[];
   ```

   使用组件：

   ```
   <app-kkb-menu [data]="userMenuData"></app-kkb-menu>
   ```

ngSwitch运用

```
<div *ngFor="let d of data" [ngSwitch]="d.type" class="menu-item">
  <!--外部链接-->
  <a *ngSwitchCase="'link'" [href]="d.url">{{d.label}}</a>
  <!--内部链接-->
  <a *ngSwitchCase="'route'" [routerLink]="d.url">{{d.label}}</a>
  <!--回调函数-->
  <a *ngSwitchCase="'callback'" (click)="d.cb()">{{d.label}}</a>
</div>
```

> 注意*ngSwitchCase后面是表达式，如果是字符串请添加单引号"'link'"

模糊查询

```
const sql = `SELECT * FROM kkb.vip_course where name like ?`;
// 请这样传条件
const results = await query(sql, '%'+req.query.keyword+'%');
```

请求防抖

原理：利用Subject派发事件，利用其pipe过滤事件，利用switchMap转换字符串为服务搜索结果

```
subject: Subject<string> = new Subject<string>();
// 派发事件
search(keyword) {
    this.subject.next(keyword);
  }
constructor () {
  this.subject.pipe(
      debounceTime(300), // 防抖
      distinctUntilChanged(), // 去重：加入紧挨着两次输入值相同则不发送
      switchMap(keyword => { // 将Observable<string>类型转换为Observable<Result<any>>
        // 判空
        if (keyword === '') {
          return of(null);
        }
        return this.ucs.searchCourse(keyword);
      })
    ).subscribe(// 搜索结果处理
      (result: Result<any[]>) => {
        // 结果处理
      }
    );
  }
```