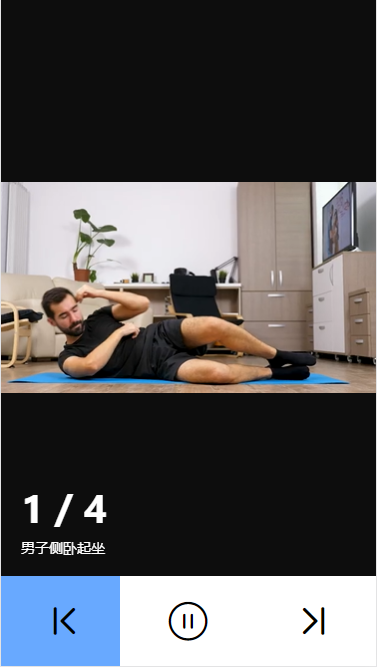
**源码健身Fitness移动Web站**

**课程训练&课程详情&播放器Day05**

# 今日任务概览

功能预览：（课程训练列表、课程详情、播放器）完成： 布局 + JS交互 + 接口联调。



附：

设计稿地址：[设计稿链接](https://modao.cc/app/629ff0afd82a66bebb7e0d87b726d618140faf20?simulator_type=device&sticky)

# 修炼技术点

* Flex布局、复杂图文混排
* 列表数据渲染、图文数据动态渲染、视频动态渲染。
* H5自定义属性、跨页面传递数据
* H5视频video高级api
* 视频播放器内核、复杂dom交互实现
* Ajax请求接口联调、PostMan的使用

# 课程训练列表

动态添加底部tabs栏目

*// 添加tab*

  addTabs('sports')

## 3.1课程训练导航实现

1. 导航核心代码html：实现导航核心布局， 课程训练激活，激活下面有一根蓝色横线，居中。

*<!-- 顶部导航 -->*

  <div class="top-nav">

    <ul class="top-nav-list dpflex">

      <li class="top-nav-item flex1"><a class="top-nav-link" href="#">跑步</a></li>

      <li class="top-nav-item flex1"><a class="top-nav-link" href="#">骑行</a></li>

      <li class="top-nav-item flex1"><a class="top-nav-link active" href="#">课程训练</a><span class="line"></span></li>

    </ul>

  </div>

1. 导航核心代码css

*/\* 顶部导航 \*/*

.top-nav {

  height: 50px;

  line-height: 50px;

  .top-nav-list {

    width: 90%;

    margin: 0 auto;

    border-bottom: 1px solid #eee;;

    .top-nav-item {

      text-align: center;

      position: relative;

    }

    .top-nav-link {

      color: #9A9AAA;

      &.active {

        font-weight: 700;

        color: #429CFF;

      }

    }

    .line {

      display: inline-block;

      position: absolute;

      left: 50%;

      transform: translateX(-50%);

      bottom: 0;

      width: 50px;

      height: 2px;

      background-color: #AAD9F7;

    }

  }

}

## 3.2课程列表数据实现

1. 实现列表html结构占位：

*<!-- 最新课程 -->*

  <h4 class="newcourse-title">最新课程</h4>

  <div class="card-newcourse" id="cardNewcourse"></div>

*<!-- 课程 -->*

  <h4 class="course-title">课程</h4>

  <div class="card-list" id="cardList">

  </div>

1. 数据动态渲染核心代码。

*/\* 获取dom \*/*

  const cardNewcourse = document.querySelector('#cardNewcourse')

  const cardList = document.querySelector('#cardList')

*/\* 渲染函数 \*/*

  function renderCourse(courseList) {

*// 取出新课程*

    const newCourse = courseList.filter(function (course) {

      return course.new === 1;

    })[0]

*// 取出其他课程*

    const courses = courseList.filter(function (course) {

      return course.new === 0;

    })

*// 渲染新课程*

    let newCourseHtml = `<div id="newcourse-container" data-id="${newCourse.id}" class="newcourse-container">

      <img class="newcourse-img" src="${newCourse.imgUrl}" alt="最新课程图片">

      <h5 class="title">${newCourse.name}</h5>

      <p class="desc">${newCourse.desc}</p>

    </div>`

    cardNewcourse.innerHTML = newCourseHtml

*// 渲染列表数据*

    let retHtml = ''

    for (let i = 0; i < courses.length; i++) {

      let course = courses[i]

      retHtml += `

          <div class="card-item" data-id="${course.id}">

          <img class="course-img" src="${course.imgUrl}" alt="课程图片">

          <h5 class="title">${course.name}</h5>

          <p class="desc">${course.desc}</p>

        </div>

      `;

      cardList.innerHTML = retHtml  *// 放到页面渲染*

    }

  }

1. 点击新课程或点击课程列表中的每一项，都需要跳转到课程详情页面，并且把id带过去。

1）首先，实现点击新课程，跳转携带id

*/\* 跳转到课程详情 \*/*

  cardNewcourse.addEventListener('click', function (e) {

    let parentNode = e.target.parentNode

    if (parentNode.className === 'newcourse-container') {

      let courseid = parentNode.dataset.id

      location.href = './course-detail.html?id='+courseid

    }

  })

2）然后，再实现点击课程列表中每一项，跳转携带id

*/\* 跳转到课程详情 \*/*

  cardList.addEventListener('click', function (e) {

    let parentNode = e.target.parentNode

    if (parentNode.className === 'card-item') {

      let courseid = parentNode.dataset.id

      location.href = './course-detail.html?id=' + courseid

    }

  })

# 课程训练详情

## 详情页面实现

1. 实现核心布局代码

*<!-- 视频图片 -->*

  <div class="poster">

    <img id="posterImg" class="poster-img" src="../../images/advertisement-bg.jpg" alt="">

    <i class="iconfont iconicon\_play"></i>

    <i class="iconfont iconleftarrow"></i>

  </div>

*<!-- 信息 -->*

  <div class="info">

    <h4 class="name" id="name">标题xxxx</h4>

    <div class="info-content">

      <span class="calorie" id="calorie">45</span>千卡

      <span class="time" id="time">15</span>分钟

      <div class="num-wrapper">

        <span class="peopleNum" id="peopleNum">99.99</span>万人练过

        <i class="iconfont iconarrow-right"></i>

      </div>

    </div>

  </div>

*<!-- 描述 -->*

  <div class="desc-wrapper">

    <div class="desc" id="desc">练习频次练习频次练习频次练习频次练习频次练习频次练习频次练习频次练习频次练习频次练习频次练习频次练习频次练习频次</div>

    <div class="info-wrapper dpflex">

      <div class="flex1">

        <h6 class="title"><i class="iconfont iconpinci"></i> 练习频次</h6>

        <span class="text">每周 <span id="frequency"></span> 次</span>

      </div>

      <div class="flex1">

        <h6 class="title"><i class="iconfont iconqixieshebei"></i> 器械</h6>

        <span class="text" id="type">无</span>

      </div>

    </div>

  </div>

*<!-- 按钮 -->*

  <div class="btn-start operation-btn-lg" id="startBtn">开始训练</div>

## Ajax获取接口动态数据渲染

1）发送ajax，根据id获取动态的详情数据

2）渲染详情页面核心代码

*/\* 获取dom \*/*

  const posterImg = document.querySelector('#posterImg')

*/\* 渲染详情 \*/*

  function renderCourseDetail(data) {

    let keyArr = ['posterImg', 'name', 'calorie', 'time', 'peopleNum', 'desc', 'frequency', 'type']

    let domArr = [posterImg, name, calorie, time, peopleNum, desc, frequency, type]

    posterImg.src = data.imgUrl *// 渲染图片*

    for (let key in data) {

      if (keyArr.includes(key)) {

        document.querySelector(`#${key}`).textContent = data[key]

      }

    }

  }

  renderCourseDetail(courseDetail)

## 4.3开始训练

点击开始训练，跳转到视频播放器界面

*/\* 开始训练 \*/*

  const startBtn = document.querySelector('#startBtn')

  startBtn.addEventListener('click', function () {

*// 把数据存入本地*

    const fragments = courseDetail.fragments

    window.localStorage.setItem('fragments', JSON.stringify(fragments))

    location.href = './player.html'

  })

# 视频播放器

## 布局实现

首先实现视频播放器 和 暂停弹出蒙层的布局

*<!-- 视频容器 -->*

  <div class="video-wrapper flex1">

    <video id="video" muted style="width: 100%; height: 100%;"></video>

    <div class="part">

      <div class="progress">

        <span id="curPart">1</span> / <span id="totalPart">8</span>

      </div>

      <p class="title" id="title">膝关节环绕</p>

    </div>

  </div>

*<!-- 控制按钮 -->*

  <div class="controls dpflex">

    <i id="preBtn" class="flex1 iconfont iconxueyuan-shangyijie"></i>

    <i id="stopBtn" class="flex1 iconfont iconbofang"></i>

    <i id="nextBtn" class="flex1 iconfont iconxueyuan-xiayijie"></i>

    <div class="progress" id="progress"></div>

  </div>

*<!-- 蒙侧 -->*

  <div class="mask" id="mask">

    <h4 class="title">放弃可以找到一万个理由，坚持只需要一个信念</h4>

    <div class="btn-group dpflex">

      <div id="btnContinue" class="btn-continue dpflex fd-column">

        <i class="iconfont iconicon\_play"></i>

        <span>继续训练</span>

      </div>

      <div id="btnOver" class="btn-over dpflex fd-column">

        <i class="iconfont iconjieshu"></i>

        结束训练

      </div>

    </div>

*<!-- 当前 -->*

    <div class="current dpflex">

      <img id="currentImg" width="100px" height="98" src="" alt="" class="flex1">

      <div class="current-text flex1 dpflex fd-column" >

        <h6 class="title">当前:</h6>

        <span class="text" id="currentTitle">xxxxx</span>

      </div>

    </div>

  </div>

## 播放器实现&自动播放下一节

1. 首先获取本地数据（视频播放片段）

*// 获取本地数据*

  let fragments = JSON.parse(window.localStorage.getItem('fragments'))

1. 渲染视频总节数

*/\* 渲染节数 \*/*

  const totalPart = document.querySelector('#totalPart')

  const total = fragments.length;

  totalPart.textContent = total

1. 实现一个自动播放视频的函数

*/\* 获取视频dom \*/*

  const video = document.querySelector('#video')

  const curPart = document.querySelector('#curPart')

  const title = document.querySelector('#title')

  let progress = document.querySelector('#progress')

  let intervalId = null

*/\* 播放 \*/*

  let index = 0

  function play() {

    curPart.textContent = index + 1;

    title.textContent = fragments[index].title

    video.src = fragments[index].videoUrl

    video.play()

    calcProgress() *// 计算进度条*

  }

  play() *// 初始化调用*

1. 监听视频播放结束事件，自动播放下一节

*// 监听视频播放结束*

  video.addEventListener('ended', function () {

    index++

    if (index > total - 1) {

      return;

    } else {

      play()

    }

  })

## 上一节&下一节

*/\* 上一节 &  下一节 \*/*

  const preBtn = document.querySelector('#preBtn')

  const nextBtn = document.querySelector('#nextBtn')

  preBtn.addEventListener('click', function () {

    index--

    if (index < 0) index = 0;

    play()

  })

  nextBtn.addEventListener('click', function () {

    index++

    if (index > total - 1) {

      index = total - 1

    }

    play()

  })

## 暂停&继续&结束

*/\* 获取dom \*/*

  const mask = document.querySelector('#mask')

  const stopBtn = document.querySelector('#stopBtn')

  const btnContinue = document.querySelector('#btnContinue')

  const btnOver = document.querySelector('#btnOver')

  const currentImg = document.querySelector('#currentImg')

  const currentTitle = document.querySelector('#currentTitle')

*/\* 暂停 \*/*

  stopBtn.addEventListener('click', function () {

    video.pause()

    currentTitle.textContent = fragments[index].title  *// 当前标题*

*// 当前图片*

    currentImg.src = fragments[index].imgUrl

    mask.style.display = 'block'

  })

*/\* 继续 \*/*

  btnContinue.addEventListener('click', function () {

    video.play()

    mask.style.display = 'none'

  })

*/\* 结束 \*/*

  btnOver.addEventListener('click', function () {

    location.href = './course.html'

  })

## 进度条实现

*/\* 播放进度条实现 \*/*

  function calcProgress() {

    intervalId =  setInterval(function () {

      let duration = video.duration

      let currentTime = video.currentTime

      if (duration && currentTime) {

        progress.style.width = (currentTime / duration) \* document.body.clientWidth + 'px'

      }

    }, 30)

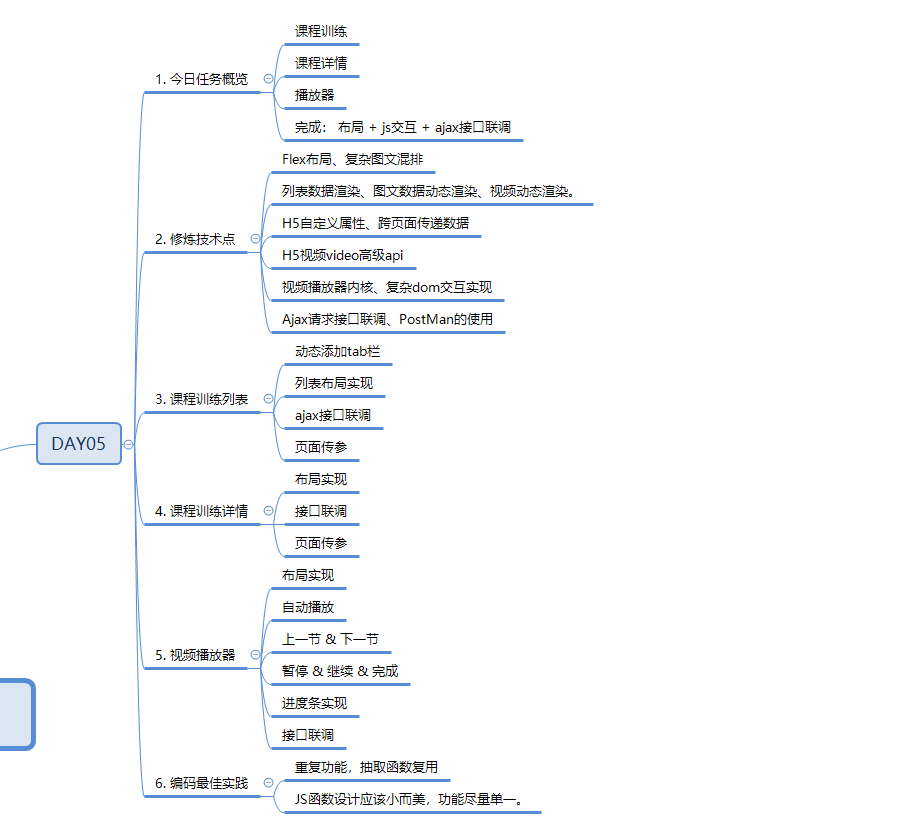
  }

# 今日研发任务CheckList

* 完成-课程训练列表布局
* 完成-课程训练列表ajax数据动态获取&渲染
* 完成-课程列表动态传递参数到课程详情
* 完成-课程详情界面布局
* 完成-课程详情界面ajax数据动态获取&渲染
* 完成-开始训练页面跳转&参数传递
* 完成-视频播放器自动播放
* 完成-视频播放器上一节\*下一节
* 完成-视频播放器暂停、继续、结束
* 完成-视频播放器进度条

PS：使用Git进行代码版本控制，commit格式：git commit -m “feat: 完成注册页背景图适配”

# 今日小结



## 编码最佳实践

* 重复功能，抽取函数复用
* JS函数设计应该小而美，功能尽量单一

## 编码意识

* 遇到复杂的技术场景，先分析，再编码，必要时候甚至可以写一些伪代码；
* 复杂场景下，更需要注意命名以及写注释；