# AI Coding 思维：告别学语法，学会下命令板

## 课程逐字稿（讲师讲解词）

第三章

共 35 页幻灯片

---

## 第 1 页

### 【幻灯片标题】AI Coding 思维：告别学语法，学会下命令

**【逐字稿】**

同学们好，欢迎来到第三章《AI Coding 思维：告别学语法，学会下命令》。在第一章我们学习了 AI 编程的发展历程，在第二章学习了 Code Agent 的核心概念和工具箱。这一章我们将聚焦一个更深层的话题——AI 驱动的编程思维跃迁。什么是思维跃迁？简单说就是从「会写代码」到「会指挥 AI 写代码」的能力升级。这不仅仅是工具的更换，更是思维方式的根本转变。过去你需要记住语法、理解 API、手动调试；现在你需要学会描述需求、设计提示词、审核 AI 产出。这一章将带你完成这个思维跃迁，掌握 AI 指挥式编程的全流程。让我们开始吧！

---

## 第 2 页

### 【幻灯片标题】前言

**【逐字稿】**

本章聚焦 AI 驱动的编程思维跃迁，颠覆传统逐行编码模式，完成从「会写代码」到「会指挥 AI 写代码」的核心能力升级。我们会讲解几个核心内容：第一，AI 代码解读——AI 生成的代码你需要能看懂、能审核。第二，Vibe Coding 核心流程与提示词框架——如何用结构化的方式指挥 AI 写代码。第三，码道云端实战——在真实平台上动手操作，包含零代码 API 工具案例。第四，Skill 与 MCP 能力集成——如何给 AI 扩展外部能力。第五，运行环境配置——让 AI 生成的代码能跑起来。通过这些内容，你将掌握 AI 指挥式编

程的全流程，依托 AI 快速完成从需求到部署的完整开发链路，实现想法到产品的高效落地。这一章实操性很强，建议大家边听边动手实践。

---

## 第 3 页

### 【幻灯片标题】课程目标

**【逐字稿】**

我们来看本章的学习目标。学完本课程后，您将能够：第一，了解 AI Coding 模式在智能编程开发中的核心价值与设计思想——知道 AI 编程为什么可行、为什么高效。第二，掌握 AI 编程协作的交互方式，包括基础对话规范与结构化提示词设计——学会怎么跟 AI 有效沟通。第三，学习 AI 辅助开发的完整流程——从需求描述到代码生成到测试验证到部署上线的全链路。第四，实践代码项目的环境部署与问题排查管理，完成实验——动手把 AI 生成的代码部署运行起来。这四个目标从认知到技能到流程到实践，层层递进。本章特别强调实践——理论听懂了不等于会用，一定要动手操作才能真正掌握。

---

## 第 4 页

### 【幻灯片标题】目录

**【逐字稿】**

我们来看本章的目录。本章分为四个部分：第一部分是「思维跃迁：从学语法到学指挥」，建立 AI 编程的正确认知。第二部分是「软件开发基础能力」，补充一些软件开发的预备知识，帮助大家理解 AI 生成的代码。第三部分是「Vibe Coding 核心流程与云端 Vibe Coding 实战」，这是本章的重点，通过实战掌握 AI 编程的核心技能。第四部分是

「码道平台」，介绍华为云的 AI 编程平台。这四部分从思维到基础到实战到平台，形成完整的学习路径。我们逐一展开。

---

## 第 5 页

### 【幻灯片标题】产业趋势：AI 编程成为主流，大量开发者使用 AI 编程工具实现效能跃升

**【逐字稿】**

我们先来看产业趋势，理解为什么 AI 编程已经成为主流。根据谷歌 2025 年 DORA 报告：高达 90%的开发者已经使用 AI，使用时长达到每天中位数两小时。写新代码、修改已有代码、写文档、创建测试用例等都已成为 AI 的主战场。80%的人认为生产力显著提升，59%的人表示代码质量更好。这说明 AI 编程不再是尝鲜，而是主流。再看发展趋势：AI 编程从代码补全，向任务级生成、项目级生成、AI 软件工程师、Agent Team 发展。Gartner 预测到 2028 年，75%的软件工程师会采用编码 Agent 进行日常开发。自主编程 Agent 发展迅猛，AI 编程未来三年对 token 算力消耗对比 25 年增长 70 倍。这些数据告诉我们：AI 编程不是要不要用的问题，而是怎么用好的问题。尽早掌握 AI 编程技能，就是尽早建立职业竞争优势。

---

## 第 6 页

### 【幻灯片标题】AI 时代开发者的新角色

**【逐字稿】**

我们来看 AI 时代开发者的新角色。传统开发者的角色是手写代码、死记语法、调试 Bug——核心能力是「执行」。AI 时代的开发者变成了 AI 驱动的创造者——定义需求、

指挥 AI、创造价值——核心能力是「决策和创造」。具体来说，AI 时代有五种新角色：需求分析师——将模糊的业务想法转化为 AI 可执行的技术需求。产品经理——对齐技术与业务认知。测试工程师——验证 AI 产出的正确性。架构师——设计系统整体结构。代码编写者——这个角色正在被 AI 接管。注意中间的「思维跃迁」——从机械执行到智慧决策。这不是说编码能力不重要了，而是编码从「核心能力」变成了「基础能力」，更高层的「需求定义、架构设计、结果审核」成为新的核心能力。大家要尽早完成这个思维转变，不要只停留在「写代码」的层面。

---

## 第 7 页

### 【幻灯片标题】思维转变：传统编程 vs Vibe Coding 编程

**【逐字稿】**

我们通过对比来看思维转变的具体内容。传统编程 vs Vibe Coding 编程，从五个维度对比。第一，学习路径——传统编程语法先行，需先精通编程语言语法、数据结构与算法，学习曲线陡峭；Vibe Coding 需求先行，无需深研底层语法，以业务需求为核心，快速上手开发。第二，核心能力——传统编程依赖严密逻辑推理，把控每个分支细节；Vibe Coding 需要沟通与拆解能力，将需求清晰描述给 AI。第三，工作模式——传统编程手动逐行编写；Vibe Coding 用自然语言指挥 AI 生成，聚焦结果审核。第四，知识更新——传统编程新技术学习成本高；Vibe Coding AI 实时同步技术前沿。第五，协作方式——传统编程人与人协作基于代码评审；Vibe Coding 人与 AI 协作更聚焦业务目标。这个对比不是贬低传统编程——传统编程能力仍然是基础，不懂代码你无法审核 AI 的产出。而是在传统编程基础上，增加 AI 协作能力，实现效率倍增。

---

## 第 8 页

### 【幻灯片标题】目录

**【逐字稿】**

我们进入第二部分——软件开发基础能力。在指挥 AI 写代码之前，你需要具备一些基础的软件开发知识，这样才能看懂 AI 生成的代码、审核其正确性。这部分会补充软件开发的核心概念，帮助大家建立代码理解能力。即使你主要用 AI 编程，这些基础知识也是必不可少的——因为你是「指挥官」和「验收者」，必须能看懂「士兵」的执行结果。

---

## 第 9 页

### 【幻灯片标题】拆解"个人日程助手 App"

**【逐字稿】**

我们以「个人日程助手 App」为例来拆解软件结构。整体功能拆解：用户可见的功能包括查看、添加、删除、修改日程、切换日期；用户不可见的功能包括保存、读取数据、验证输入、数据持久化；运行支撑包括页面渲染、网络请求、本地存储。软件有四大核心组成：前端——用户直接看到和操作的界面；后端——处理数据和业务逻辑的服务器程序；数据——存储和管理的所有信息；运行环境——让代码能够执行的基础平台。理解这四大组成很重要——因为 AI 帮你生成的代码也会涉及这四个方面，你需要知道 AI 生成的代码分别对应哪个部分，才能审核其完整性。比如 AI 只生成了前端代码，你就要问：后端 API 呢？数据存储呢？

---

## 第 10 页

### 【幻灯片标题】软件核心组成的生活化类比（总结下）

**【逐字稿】**

我们用「餐厅」来类比软件架构，帮助大家直观理解。前端对应餐厅前台加服务员——展示菜单、接收点餐、传递菜品，在 App 中就是展示界面、接收操作、展示结果。后端对应后厨加厨师——处理订单、制作菜品、管理食材，在 App 中就是处理数据、执行逻辑、管理存储。数据对应食材仓库加菜单——存储原材料和菜品信息，在 App 中就是存储日程数据、用户设置。运行环境对应餐厅场地加厨房设备——提供工作场所和工具，在 App 中就是提供代码运行的平台和依赖。一个关键认知：用户始终只和「服务员」（前端）直接交互，而「后厨」（后端）的复杂加工、「仓库」（数据）的存取、「场地」（环境）的运转，对用户来说是透明的黑盒。这个类比帮助大家理解为什么前端和后端要分离——各自职责不同，各自优化。

---

## 第 11 页

### 【幻灯片标题】前后端分离架构：谁负责页面，谁负责数据

**【逐字稿】**

我们来看前后端分离架构。现代 Web 开发的标准架构是：前端在浏览器中渲染页面，后端在服务器上处理逻辑，通过 API 接口进行数据交互。用户发起请求→前端渲染页面→后端查询数据库→返回 JSON 数据→前端展示。前端负责页面展示、用户交互、数据渲染，最终产出是美观易用的界面。后端负责数据处理、业务逻辑、接口开发，最终产出是稳定高效的 API 接口。前后端分离的好处是各自独立开发、测试和部署——前端团队和后端团队可以并行工作。理解这个架构对 AI 编程很重要——因为当你让 AI 开发一个 Web 应用时，你需要告诉 AI 前端和后端分别要做什么，否则 AI 可能只生成前端没有后端，或者混在一起。

---

## 第 12 页

### 【幻灯片标题】“个人日程助手 App” 文件结构解析

**【逐字稿】**

我们以「个人日程助手 App」为例来看文件结构。一个项目有很多文件，你需要知道改什么找哪个文件。改页面文字、按钮、图片→找.html 文件；改颜色、字体、布局→找.css 文件；改点击事件、数据请求→找.js 文件；改接口返回数据、业务逻辑→找后端.py 或.js 文件；改接口地址、端口号→找.json 或.env 或.py 配置文件。这个对应关系很重要——因为 AI 编程时你会经常说「帮我改一下 XX」，你需要知道这个 XX 对应哪个文件，才能给 AI 精确的指令。比如你说「帮我改一下页面颜色」，AI 需要知道你要改的是 CSS 文件；如果你说得更精确「帮我修改 style.css 中的主题色为蓝色」，AI 就能更准确地执行。

---

## 第 13 页

### 【幻灯片标题】一眼看懂前端文件（确认用到的内容）

**【逐字稿】**

我们来看前端文件的识别方法。常见前端文件扩展名：.html 是超文本标记语言，定义页面结构（骨架），修改页面元素时找它。.css 是层叠样式表，定义页面样式（外观），修改颜色字体布局时找它。.js 是 JavaScript 脚本，定义页面行为（交互），修改点击事件数据请求时找它。.json 是数据文件，存储配置和静态数据，修改配置时找它。.svg 是矢量图形，页面中的图标。.png/.jpg/.gif 是图片文件。.ico 是网站图标。关键在于快速识别：哪里是页面主标题、哪里是可交互的按钮、哪里是数据展示的列表区域，从而理解页面的信

息架构。这个能力对审核 AI 生成的前端代码很重要——你需要能看懂 AI 生成的 HTML 结构是否合理。

---

## 第 14 页

### 【幻灯片标题】前端代码分解实战：日程列表页面

**【逐字稿】**

我们通过一个实际例子来看前端代码的三大组成部分。以日程列表页面为例：HTML 负责结构——用标签定义页面骨架，比如<h1>我的日程</h1>是标题，<input>是输入框，<button>是按钮，<ul>是列表容器。CSS 负责样式——body 设置字体和布局，h1 设置标题颜色和居中，#add-btn 设置按钮的背景色和悬停效果。JavaScript 负责行为——获取页面元素、添加点击事件、创建新元素、删除日程。HTML、CSS、JS 三者分离是前端开发的最佳实践——结构、样式、行为各司其职。当你让 AI 生成前端代码时，AI 通常会同时生成这三个部分。你需要能看懂每个部分做什么，才能审核代码是否正确。理解语义化标签很重要——<header>、<nav>、<main>、<footer>等标签不仅定义结构，还表达语义，有利于 SEO 和无障碍访问。

---

## 第 15 页

### 【幻灯片标题】前端代码分解实战：CSS 样式

**【逐字稿】**

我们继续深入看 CSS 样式部分。CSS 的核心逻辑是「从骨架到外衣」——它不改变 HTML 结构，而是通过规则告诉浏览器如何渲染元素。比如 body { font-family: Arial; max-width: 600px; }设置整体字体和宽度；h1 { color: #333; text-align: center; }设置标题颜色和居

中；#add-btn { background-color: #4CAF50; }设置按钮为绿色。CSS 通过选择器精准定位元素——#add-btn 选中 id 为 add-btn 的元素，.schedule-item 选中 class 为 schedule-item 的所有元素。当你让 AI 修改页面样式时，AI 会修改 CSS 文件中的规则。你需要能看懂 CSS 选择器和属性，才能判断 AI 的修改是否正确。比如 AI 把按钮颜色改成了你想要的颜色吗？布局是否如你预期？

---

## 第 16 页

### 【幻灯片标题】前端：JavaScript 逻辑（main.js）

**【逐字稿】**

我们来看 JavaScript 逻辑部分。JavaScript 负责页面的动态行为。以日程添加功能为例：首先用 document.getElementById 获取页面上的按钮和输入框元素，然后用 addEventListener 给按钮添加点击事件，点击时获取输入框的值，用 document.createElement 创建新的列表项，设置文本内容，添加删除按钮，最后用 appendChild 将新元素添加到页面上。JavaScript 的核心能力是 DOM 操作——动态创建、修改、删除页面元素，实现内容的实时更新。当你让 AI 添加交互功能时，AI 会生成 JavaScript 代码。你需要理解基本的 DOM 操作——获取元素、添加事件、创建元素、修改内容——才能审核 AI 的代码是否正确实现了你想要的功能。

---

## 第 17 页

### 【幻灯片标题】门户网站应用的云端形态

**【逐字稿】**

我们来看一个门户网站应用的云端形态案例。把本地开发完成的门户网站应用部署到华为云 ECS，并接入 GaussDB。应用类型是门户网站，核心组成包括门户前台、管理后台、后端服务。部署环境是华为云 ECS 加 GaussDB。部署方式是码道智能体自动打包、上传、部署。一句话理解整个架构：码道负责开发与联调，码道智能体负责自动化部署动作，ECS 负责应用运行，GaussDB 负责数据持久化。这个案例展示了 AI 编程的完整链路——不仅用 AI 写代码，还用 AI 部署上线。后续实操环节我们会动手体验这个流程。理解部署架构很重要——因为 AI 编程不只是生成代码，还要让代码跑起来，部署是最后一公里。

---

## 第 18 页

### 【幻灯片标题】目录

**【逐字稿】**

我们进入第三部分——Vibe Coding 核心流程与云端实战。这是本章的重点部分。前面讲了思维跃迁和基础能力，现在进入实操环节。我们会通过一个完整的案例——校园美食地图助手——来演示 Vibe Coding 的全流程，包括需求描述、AI 生成、能力补充、测试验证、反馈修复五个环节。通过这个案例，你将亲身体验 Vibe Coding 的威力和正确使用方法。

---

## 第 19 页

### 【幻灯片标题】Vibe Coding(氛围编程)的核心开发闭环（精简）

**【逐字稿】**

我们先来看 Vibe Coding 的核心开发闭环。共五个环节：第一，描述需求——梳理清晰需求和指标，给 AI 精准指令。这是起点，需求描述的质量直接决定后续所有环节的质量。第二，AI 生成——AI 依据需求快速生成完整代码。这是 Vibe Coding 的核心环节，AI 一次性生成最小可用版本。第三，能力补充——接入外部工具和组件拓展程序能力。AI 生成的代码可能需要调用外部 API，比如地图、天气等，需要补充这些能力。第四，测试验证——多维度测试校验代码功能、性能与安全。AI 生成的代码需要人工测试，确保功能正确。第五，反馈修复——收集反馈，和 AI 迭代修复优化代码。发现问题后跟 AI 对话修复，循环迭代。这五个环节构成一个闭环——不是一次走完就结束，而是可以反复迭代优化。

---

## 第 20 页

### 【幻灯片标题】环节 1 结构化需求：提示词心法

**【逐字稿】**

我们来看第一个环节——结构化需求提示词心法。以校园美食地图助手为例，一个好的提示词应该包含四个部分。技术要点：限定技术栈和约束——比如用 fetch 直接调用高德 REST API，地图展示用高德地图 JS API，准备 Mock 数据作为降级备选。功能要求：梳理业务流程和功能清单——比如 IP 定位、逆地理编码、POI 搜索、地图展示。界面风格要求：给定参考产品和风格配色——比如移动端适配、卡片布局、暖色调。输出格式要求：规定交付文件类型——比如输出完整 HTML 代码，确保复制即可运行。对比一下：如果你只说「帮我做个美食地图」，AI 可能给你一个很简陋的页面；但用结构化提示词，AI 能一次性生成一个功能完整、界面美观的应用。这就是结构化提示词的价值——把模糊的想法转化为精确的指令。

---

## 第 21 页

### 【幻灯片标题】环节 2 AI 生成：一键产出最小可用版本

**【逐字稿】**

我们来看第二个环节——AI 生成。在码道平台中，AI 会根据你的提示词生成代码。这里要理解码道平台的项目结构：顶层配置文件夹包括.arts/settings.json（全局配置）和.codeartsdoer 核心智能体工作目录。.codeartsdoer 下有 agents/（智能体定义）、mcp/（外部能力扩展配置）、rule/（编码规范）、skills/（可复用流程技能）。需求规范目录 specs/ 下有 spec.md（需求说明）、design.md（UI 规范）、tasks.md（任务清单）。理解这个结构很重要——因为 AI 生成的代码会放在这些目录中，你需要知道每个目录的作用。MCP 负责外部工具调用，Skill 负责固定业务流程模板，二者是 AI 拓展能力的两大核心。在实际操作中，你只需要描述需求，AI 会自动在对应目录生成文件。

---

## 第 22 页

### 【幻灯片标题】环节 3 外部能力补充：SKILL 固化规则并添加天气 MCP

**【逐字稿】**

我们来看第三个环节——外部能力补充。AI 生成的代码可能需要调用外部 API，比如高德地图。这时需要配置 MCP 和创建 Skill。以校园美食推荐为例：在码道的 SKILL 配置下创建 food-recommend-skill，包含 name（food-recommender）、description（校园周边美食推荐专家），以及推荐规则——距离优先级、价格过滤、口味分类、评分参考。同时配置高德地图 MCP——在聊天窗口输入「我获取到了高德开放平台的 API Key，请帮我配置高德地图 MCP」，码道会自动帮你配置。这个环节展示了 Vibe Coding 不只是让 AI 写

代码，还需要给 AI 配备外部工具。MCP 连接外部服务（如地图 API），Skill 封装业务逻辑（如推荐规则），两者协同让 AI 生成的应用具有真正的业务能力。

---

## 第 23 页

### 【幻灯片标题】环节 3 外部能力补充：MCP 与 SKILL 的协同应用场景

**【逐字稿】**

我们来看 MCP 与 Skill 的协同应用场景。以自动生成业务周报为例：Skill 制定完整执行流程——先拉取业务数据，再分类汇总，最后按固定模板生成报表，同时约束文案风格和必填板块。MCP 承接数据调取动作——对接业务数据库、项目管理系统、考勤系统等，批量查询原始数据。对比两者的定位：MCP 负责连接外部世界（数据加动作），粒度通常是原子操作（如 read_file），必须涉及外部，需要编程实现 MCP 协议接口。Skill 负责封装复杂任务流程，粒度是多步工作流，不一定涉及外部，可以用声明式配置。二者协同就是：由 Skill 编排完整工作流，流程中需要对接外部资源的环节统一通过 MCP 实现。理解这个协同模式对实际开发很重要——它让你能构建复杂的自动化工作流。

---

## 第 24 页

### 【幻灯片标题】环节 4 测试验证

**【逐字稿】**

我们来看第四个环节——测试验证。AI 生成的代码需要人工测试，不能直接信任。以校园美食地图助手为例，测试分两步：第一步，测试智能推荐模块——在搜索框输入「火锅」并搜索，观察每个餐厅卡片是否自动生成推荐理由，比如「这家重庆火锅人均 65 元，适合宿舍聚餐，辣度可选」。推荐理由应结合价格、口味、距离等因素自动生成。第

二步，测试价格档次筛选器——依次点击平价、中档、高档选项，验证餐厅列表是否按规则过滤：平价人均小于 30 元，中档 30 到 80 元，高档大于 80 元。测试验证是 Vibe Coding 中不可跳过的环节——AI 生成的代码可能有逻辑错误，只有通过测试才能确认功能正确。建议每次 AI 生成代码后都做功能测试，不要等到最后才发现问题。

---

## 第 25 页

### 【幻灯片标题】环节 5 反馈修复：体验与质量优化

**【逐字稿】**

我们来看第五个环节——反馈修复。测试发现问题后，需要跟 AI 对话修复。以校园美食地图助手为例，修复分两类：第一类是 UI 细节体验升级——交互视觉区分、加载状态反馈、全端响应适配、智能模糊检索，优化后界面交互效率提升 40%。第二类是代码质量深度重构——安全漏洞封堵、规范化文档注释、样式变量统一管理、项目文档完善，优化后代码冗余减少 30%，漏洞率降至 0。反馈修复的关键是给 AI 精确的反馈——不要说「有问题」，要说「点击餐厅后距离显示为 0m，应该显示实际距离，超过 1000 米转换为公里」。反馈越精确，AI 修复越准确。这个环节是 Vibe Coding 闭环的最后一环——发现问题、反馈 AI、AI 修复、再测试，循环迭代直到满意。

---

## 第 26 页

### 【幻灯片标题】环节 5 反馈修复：解决 BUG

**【逐字稿】**

我们来看一个具体的 BUG 修复案例。问题是：点击任意餐厅后，详情卡片中显示的距离都是 0m，与实际距离不符。修复要求：检查 MCP 返回数据中的距离字段，确保每个

餐厅正确显示从用户位置到该餐厅的实际距离，如果距离超过 1000 米自动转换为公里。在聊天窗口中输入自然语言提示词描述这个问题和修复要求，AI 就能帮你修复。这个案例展示了 Vibe Coding 的一个核心优势——修 BUG 不需要你自己找代码改代码，只需要描述问题，AI 帮你定位和修复。但注意，你需要能描述清楚问题是什么、预期行为是什么，这需要你理解业务逻辑。所以即使有 AI 帮忙，你仍然需要理解代码在做什么。

---

## 第 27 页

### 【幻灯片标题】环节 5 反馈修复：AI 辅助调试

**【逐字稿】**

我们来看 AI 辅助调试的思维转变。从「我找 bug」到「让 AI 找 bug」——借助 AI 的逻辑推理与代码理解能力，将繁琐的人工排查转化为高效的智能协作。调试流程分五步：遇到报错→复制信息→描述问题→AI 修复→验证结果。你只需要做「报案人」而非「侦探」——不需要自己大海捞针找问题，只需要准确提供「案发现场」（错误信息、上下文、预期结果），让 AI 成为你的技术专家。常见错误类型：语法错误——代码不符合语言规范，有明确报错位置；逻辑错误——代码能跑但结果不符预期，需结合业务逻辑分析；环境错误——依赖缺失、版本不兼容、权限不足。向 AI 有效反馈的方法：精准描述「输入 X 时预期得到 Y 但实际返回 Z」，附带报错截图和关键代码片段，如果方案不完美可以追问「此方案导致 A 问题，如何调整」。这个思维转变能极大提升调试效率——你不再需要是 debug 专家，AI 帮你 debug，你负责描述问题和验证结果。

---

## 第 28 页

### 【幻灯片标题】Vibe Coding 的实操技巧

**【逐字稿】**

我们来总结 Vibe Coding 的实操技巧。第一，先搭骨架再迭代——首轮指令仅生成最小可运行版本，再通过多轮对话逐步叠加功能。不要期望一次生成完美代码。第二，聚焦目标而非实现——明确最终交互效果与验收标准，只定义「要实现什么」不约束「如何实现」。第三，直接复制报错日志——功能异常时直接将控制台错误信息发给 AI，快速定位修复。第四，复用成熟 UI 风格——以行业主流产品为参考，一句话描述即可获得贴合预期的界面。第五，用规范统一代码风格——创建.coding-rules.md 文件约定编码准则，AI 自动遵循。同时要了解 Vibe Coding 的优劣势：前端 UI 开发、CRUD 业务逻辑、Mock 与原型是强项；安全敏感逻辑、性能优化、架构决策仍需要人类判断。记住：AI 是副驾驶，你才是驾驶员。

---

## 第 29 页

**【幻灯片标题】目录**

**【逐字稿】**

我们进入第四部分——码道平台。前面讲了 Vibe Coding 的理论和实战技巧，现在介绍华为云码道平台——这是实践 AI 编程的利器。我们会介绍几种 AI 编程工具的对比，以及码道平台的技术架构，帮助大家选择合适的工具并理解平台能力。

---

## 第 30 页

**【幻灯片标题】AI Coding 工具基础**

**【逐字稿】**

我们来看 AI Coding 工具的基础对比。Cursor 是本地 AI 原生 IDE，基于 VSCode 改造，支持整文件全项目上下文、对话式修改代码、报错自动修复。GitHub Copilot 是 IDE 插件，实时代码补全、单行函数快速生成。Replit AI 是国外云端在线 IDE，适合前端和学生快速做 Demo。码道 CodeArts 是华为云面向企业教学和团队工程交付的云端 AI 全栈开发平台，支持项目级代码生成、代码续写、研发知识问答、单元测试用例生成等。核心认知：把 AI 当成你的编程合伙人——清晰的指令加耐心的互动等于高效的产出。不要期望 AI 一次完美，但通过有效沟通，它能成为你最得力的副驾驶。不同工具有不同定位，选择时要考虑你的场景——个人快速开发用 Cursor 或 Replit，企业团队工程用码道。

---

## 第 31 页

### 【幻灯片标题】华为云码道（CodeArts）代码智能体技术架构图

**【逐字稿】**

我们来看华为云码道的技术架构图。整体分为端侧和云侧。端侧包括用户使用终端（AI IDE、VSCode Plugin、CLI、Mobile App、Web、Jetbrains Plugin）和端侧 Agent（MainAgent 研发主智能体、SubAgent 子智能体、MCP/Tools、Skill 研发经验、LocalTurboContext 本地上下文加速、LSP 语言服务接入）。云侧包括 AI Infra 昇腾云算力基础设施、研发上下文工程（TurboContext 上下文加速、Codebase 代码仓理解、KnowledgeRepo 研发知识库）、推理调度（InferHub）、运营运维（Portal 租户运营看板、AOM 指标监控、APM/LTS 故障定界定位）、Agent 调度（物理多租 Agent、逻辑多租 Agent、环境管理）。底层是 MaaS 代码大模型。这个架构展示了码道平台的全栈能力——从端到云，从推理到上下文到运维，是一个企业级的 AI 编程平台。理解这个架构有助于你充分利用平台能力。

---

## 第 32 页

### 【幻灯片标题】本章小结

**【逐字稿】**

我们来做一个本章小结。第一，阐述了 AI 编程的思维跃迁逻辑，建立了「指挥 AI 写代码」的全新开发认知——从执行者变成指挥者。第二，掌握了 Vibe Coding 核心开发闭环与结构化提示词框架，学会与 AI 高效协作——描述需求、AI 生成、能力补充、测试验证、反馈修复。第三，通过零代码实战，熟练运用码道云端平台完成项目开发——从想法到可运行的应用。第四，理解集成外部能力的基础能力——MCP 连接外部服务，Skill 封装业务流程。第五，掌握了 AI 生成代码的快速识别与核验方法，能够验证并迭代优化代码——看懂 AI 的代码，测试其正确性，反馈修复问题。这五点构成了 AI 编程思维跃迁的完整能力图谱。

---

## 第 33 页

### 【幻灯片标题】思考题

**【逐字稿】**

我们来看一道思考题。Vibe Coding 的核心原则是聚焦于什么而非纠结于什么？选项 A 说聚焦实现方式 How 而非目标 What——这说反了，Vibe Coding 是聚焦目标不纠结实现。选项 B 说聚焦目标 What 而非实现方式 How——这是正确的，Vibe Coding 的核心理念就是描述你要什么，不约束怎么做。选项 C 说聚焦代码语法而非业务逻辑——也反了，Vibe Coding 不关心语法。选项 D 说聚焦代码质量而非开发速度——不对，Vibe Coding 追求速

度。正确答案是 B。这道题的核心是帮助大家理解 Vibe Coding 的思维本质——你负责定义目标，AI 负责实现细节。不要纠结于代码怎么写，而要聚焦于你要什么效果。

---

## 第 34 页

### 【幻灯片标题】学习推荐

**【逐字稿】**

给大家一些学习推荐资源。华为云官网是 https://www.huaweicloud.com/，可以了解华为云的 AI 编程产品和服务。华为云学堂是 https://edu.huaweicloud.com/，有丰富的 AI 编程课程和实验环境。建议大家课后访问这些平台，动手实践本章讲的内容。特别是码道平台，可以注册体验版，亲自体验 Vibe Coding 的全流程。本章的内容实操性很强，光听不练只能理解概念，真正动手做了才能掌握技能。建议大家在码道平台上完成校园美食地图助手的实战练习，把理论转化为实操经验。后续课程会有更多实战环节，带大家深入每个技术细节。好，本章内容就到这里，感谢大家的聆听！

---

## 第 35 页

### 【幻灯片标题】版权所有©2026，华为技术有限公司，保留所有权利。

**【逐字稿】**

这是版权页。本资料版权归华为技术有限公司所有，保留所有权利。本资料是华为的保密信息，所有内容仅供华为授权的培训客户内部使用，禁止用于任何其他用途。未经许可，任何人不得对本资料进行复制、修改、改编，也不得将本资料或其任何部分或基于本资料的衍生作品提供给他人。请大家遵守保密规定。感谢大家的聆听，我们下节课再见！