

Lesson 1.9 – Using XO for Project Design

Duration

50–55 minutes

Learning Objectives

- **Utilize** CreatiCode XO to brainstorm and outline new project ideas.
 - **Develop** skills in crafting specific and manageable project plans with XO's assistance.
 - **Implement** step-by-step project plans by engaging in iterative questioning with XO.
 - **Recognize** and **understand** the limitations of XO in project development.
 - **Apply** best practices in prompt engineering to effectively use XO as a project planning tool.
-

Preparation Steps

Resource Materials:

- Familiarize yourself with the [CreatiCode XO – New Project Design](#) tutorial.
 - Prepare example project ideas to demonstrate XO's capabilities.
-

Lesson Outline

1. Introduction to XO for Project Design (5–10 minutes)

- **Overview:**
 - Introduce the focus of today's lesson: using CreatiCode XO to design and plan new projects.
 - Explain the role of XO as an assistant in project design, emphasizing that XO aids in brainstorming and outlining rather than executing projects.
- **Key Points:**
 - XO can provide project outlines based on student ideas.
 - Importance of clear and specific prompts to receive useful guidance from XO.

- XO's limitations in handling large, complex requests and generating full projects.

2. Brainstorming Project Ideas with XO (10 minutes)

- **Activity:**
 - **Teacher Demonstration:**
 - Show how to initiate a project design session with XO.
 - Example Prompt: "I want to build a simple educational quiz app using ChatGPT blocks. How should I start?"
 - Display XO's response, highlighting the project overview, key components, and logical steps.
 - **Student Practice:**
 - Students think of a basic project idea they are interested in (see examples at end of the [tutorial](#)).
 - Individually, students input their project ideas into XO and review the generated outlines.

3. Customizing Project Plans with Detailed Prompts (10 minutes)

- **Activity:**
 - **Teacher Explanation:**
 - Discuss how adding more details to prompts results in more tailored and unique project outlines.
 - Example Enhanced Prompt: "Design a trivia game where players answer questions about space. The game should have multiple difficulty levels and provide hints using ChatGPT."
 - **Student Practice:**
 - Students refine their initial prompts by adding specific details about their project's functionality, themes, or unique features.
 - Encourage students to compare the outlines generated from basic vs. detailed prompts.

4. Implementing the Project Step-by-Step (10 minutes)

- **Activity:**
 - **Teacher Demonstration:**
 - Show how to break down the project outline into manageable steps by asking follow-up questions.
 - Example Follow-Up Prompt: "What are the key components needed for a trivia game about space?"
 - Display XO's detailed response outlining components such as UI elements, question database, scoring system, etc.
 - **Student Practice:**
 - Students select one step from their project outline and ask XO for detailed instructions or suggestions on implementing that step.

- Encourage iterative questioning to flesh out each component of their project.

5. Understanding XO's Limitations (10 minutes)

- **Discussion:**
 - **Teacher Explanation:**
 - Highlight that XO cannot write entire projects or generate extensive code automatically.
 - Discuss scenarios where XO's responses may be generic or incomplete if prompts are too broad.
 - Emphasize the importance of keeping project scopes manageable and specific.
 - **Student Reflection:**
 - Students share their experiences where XO provided limited assistance.
 - Discuss strategies to refine prompts to overcome these limitations.

6. Wrap-Up and Q&A (5 minutes)

- **Summary:**
 - Recap the key points covered in the lesson:
 - Using XO to brainstorm and outline projects.
 - Importance of detailed and specific prompting.
 - Recognizing XO's limitations and managing project scope.
- **Questions:**
 - Open the floor for any questions or clarifications.
 - Address any challenges students faced during the activities.

Extensions & Differentiation

For Advanced Students:

- Challenge them to create more complex project outlines by incorporating multiple categories of blocks (e.g. 3D, AI)
- Encourage experimentation with varied prompt styles to achieve more creative and unique project plans.

For Students Needing Support:

- Pair them with peers for collaborative prompting and project planning.

- Provide additional examples of detailed prompts and step-by-step guidance to help them formulate effective questions for XO.
-

Notes for Teachers

Understanding XO's Role:

- Reinforce that XO is an assistant designed to support the learning process, not to replace the need for student creativity and problem-solving.
- Emphasize the importance of guiding students to ask specific, manageable questions to get the most out of XO.

Technical Readiness:

- Ensure that all students can access and navigate the CreatiCode Playground and effectively use XO.
- Have backup project ideas and example prompts ready to assist students who may struggle with brainstorming.

Encouraging Creativity:

- Foster a classroom environment where students feel comfortable experimenting with different project ideas and prompt styles.
- Highlight the value of iterative design and the importance of refining prompts to improve project outcomes.

Assessment

Wayground format:

https://wayground.com/admin/assessment/68c0e5a7b7f4a50a23c50506?source=lesson_share

Questions (1 point each)

1. Which prompt is **best** for getting a useful **project outline** from XO?
 - a) "Give me all the code for a platformer."
 - b) "Outline a 2D maze game. Include: player movement, keys/doors, timer, win/lose"

states. List components and the first 3 implementation steps.”

c) “Game please.”

d) “Write a huge project for me.”

2. You ask XO: “Make **Minecraft in 3D**.” XO’s reply is generic. What’s the **best next move**?
 - a) Repeat the same request until it works.
 - b) Narrow the scope: “Start a flat world with one place/destroy block mechanic. List assets, variables, and step-by-step setup.”
 - c) Ask XO to paste the full finished code.
 - d) Give up—XO can’t help with design.
 3. **Short answer:** Write a **detailed prompt** that asks XO for a **project outline**. Your prompt must include:
 - project type/topic,
 - **2–3 concrete features**,
 - **any constraint** (e.g., blocks you’re allowed to use, difficulty, time limit),
 - a request for **components + first steps**.
 4. **Short answer:** Suppose XO says a project will have a **key and a door** – the player must collect the key to open the door and win. Write a **follow-up prompt** you would send to XO to get help coding this.
 5. **Short answer:** Name **one limitation** of XO for project design **and one strategy** to mitigate it so you still make progress.
-

Answers & Rubrics (1 point each)

1. **b** — It requests an **outline**, names **features**, and asks for **components/steps**.
2. **b** — **Scope down** to a small slice and ask for steps/assets/variables.
3. **Rubric (1 pt total)**
 - **1.0:** Includes **all four**: topic, 2–3 concrete features, one explicit constraint, and a request for components + first steps.

- Example:

Design a **quiz game** about space. Features: **difficulty levels, hints via ChatGPT, score with streak bonus**. **Constraint:** level 1 only; use **events + variables** (no 3D). ****Please list components** (sprites/backdrops, variables, messages) and the **first 3 steps** to build.”

- **0.5:** Missing exactly **one** required element.
- **0.0:** Missing **two or more** elements or off-topic.

4. Rubric (1 pt total)

- **Example Prompts:**
 - “How do I code it so the door opens when the player has the key?”
 - “How do I use a variable to check if the player has the key?”
 - “What blocks do I need to make the door open after getting the key?”
- **Scoring (1 point total):**
 - **1.0** – Clear and specific prompt asking how to code the key-door logic.
 - **0.0** – Prompt is vague or unrelated to the key-door mechanic.

5. Rubric (1 pt total)

- **1.0:** States **one real limitation and one practical mitigation** that fits design use.
Examples:
 - Limitation: XO **can’t produce full large projects** or long code reliably. Mitigation: **iterate:** ask for **components + first steps**, then drill down feature-by-feature.
 - Limitation: Responses get **generic** if prompts are broad. Mitigation: add **specific features/constraints** (blocks allowed, target mechanics, time/difficulty).
 - Limitation: XO **can’t test or run** designs. Mitigation: **prototype in the editor**, then refine prompts with what worked/didn’t.
- **0.5:** Gives only a limitation or only a mitigation, or both are vague.

- **0.0:** Incorrect (e.g., claims XO can execute projects) or irrelevant.