HOW TO BUILD A PROJECT

CoderDojo

About this series

I'm learning: Project Design

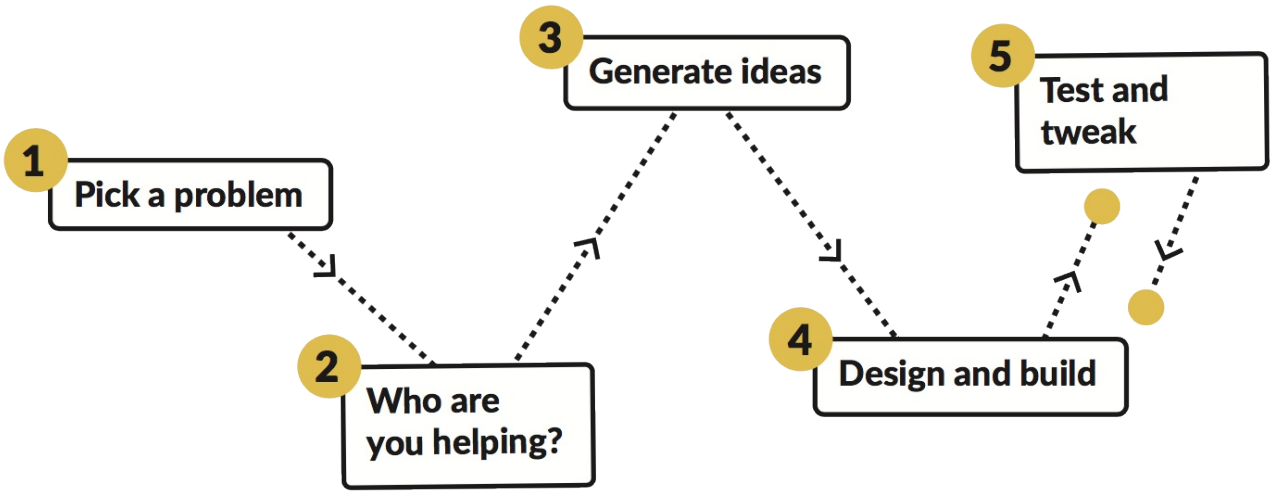
1 You've decided that you want to do a project. That's exciting!

But just how do you get started? It can seem like a daunting task, but don't worry, you've got this. The trick is to break the project down into little pieces and start small.

For example, when someone wants to build a house, they will break that project down into pieces:

- They might first decide what kind of house to build
- Then they draw up plans for what it should look like
- Next, they'll work out how they will build it based on the plans
- Finally, they gather the materials (and maybe builders!) needed and then do the building work

2 These Meta Sushi Cards will take you through the process of designing and building your technology project from start to finish. There are five main steps you will carry out:



Pick a problem: How do you want to change the world with technology? Look up how other people have solved similar issues.

Who are you helping? Think about what kinds of people your project will help and how it will work best for them. How can you adapt what you are building to their needs?

Generate ideas: Collect ideas, no matter how wacky, for how your project might work. Pick your favourite!

Design and build: Draw some designs for the idea you picked, then start making it! Remember: your Dojo's Mentors can help if you have trouble.

Test and tweak: As soon as some of the project works, get people to try it! Ask them what aspects of it you could make better or more useful. Tweak your project, and test it again.

This cycle of testing and tweaking is an **iterative** process. That means you repeat parts of it, maybe lots of times, before you are finished.

1 Pick a problem

The first step is to **define** the purpose of your project.

Who will you be making it for?

The person or people who will use your project are called your **user** or **users**. They could be a specific person or group of people, such as a friend or relative, a teacher, the people in your Dojo or school, or even you yourself. The user doesn't have to be someone you know: it could also be a more general group of people, for example school kids of a certain age, parents, people with a disability, or people who like cats!

What problem will it solve?

What do you want to help your users with? Why do you want to help them? Maybe your neighbour has a dog that keeps getting lost, and you want to help them keep track of it. Perhaps you want to help people to prepare for school or hospital, or to learn more about the environment. Or maybe you want to help somebody to have fun!

Has it been done before?

Do some research to find out if anybody has done something similar to your project before, or has tried to solve a similar problem.

Be specific!

Define a **problem statement** by writing down specifically what you want to achieve. Your project will be much more useful if you have a clear idea of what you want it to do and why before you start building it.

2 Who are you helping?

In this step you will try to gain **empathy** for the person or people you are making the project for. That means you will put yourself in their shoes to try to **understand** their needs.

A great way to **empathise** with your user(s) is by asking them questions. Get them to tell you about their experiences with the thing you are helping them with. **Investigate** ways in which your project could help them. Keep asking them "Why?" to get as much detail as you can! Be sure to listen carefully to their answers, and write down lots of notes to keep track of all the little details. Or you could even ask their permission to make a recording of the conversation.

- If you can't talk to your user directly, you could make **observations** based on things you've read or seen on TV. Be sure to base your observations on more than one text/video.

It's OK if you have some ideas already about what your user(s) might say, but make sure you ask questions to test whether these ideas — your **assumptions** — are correct! Find out what's important to them. You are trying to see things from their point of view, so you that you can understand **what they need or want**.

Another thing to think about is whether your user is comfortable with technology! Will they be able to use your project, or will they need help?

Empathise

This step is all about **caring** about your user and considering what they need or want. After all, they are the person who will be using your finished project!

3 Generate ideas

What will you make?

With a purpose defined for your project, you can now get the creative juices flowing and start **brainstorming ideas** for how you could solve the problem you picked. Pens, pencils, and lots of paper at the ready!

No writing here: sketch out your ideas with **pictures** instead of using words. Don't worry about your drawing skills! Since the aim is to describe **ideas** and not to create a work of art, stick figures and squiggly lines are perfect.

Make sure you put down **every** idea that pops into your head: you are going for quantity, not quality. Let your imagination run wild. Who cares if an idea is good or bad? You can think about that later. If you only have one or two ideas, that's OK, too! Right now, you just want to get everything out of your head and onto paper.

How will you make it?

After you've sketched out your ideas, look them over and think about which ones you like. Pick your favourite, and then start jotting down some thoughts on how you might make this idea a reality.

List some of the things you want your project to do, as well as some of the things it won't do. Decide what technology you will use. For example, will you be building a mobile phone app, or a Scratch project, or perhaps a machine or device, or something else?

Write everything down

Not every idea has to be amazing! In fact, during building almost every brilliant invention, the chances are that the inventor had a tonne of ideas that turned out to be silly or terrible. That's how brainstorming works!

4 Design and build

Before you jump in and start coding or constructing circuits, it's a good idea to plan how your project will work and how it should look, so that know you what you'll need to do.

Work out what **information** you'll need, if any. This could be information your project presents to the user, or answers the user needs to **input** when using the project, for example by choosing between different buttons or options, or by entering text. How will you use the information in your project?

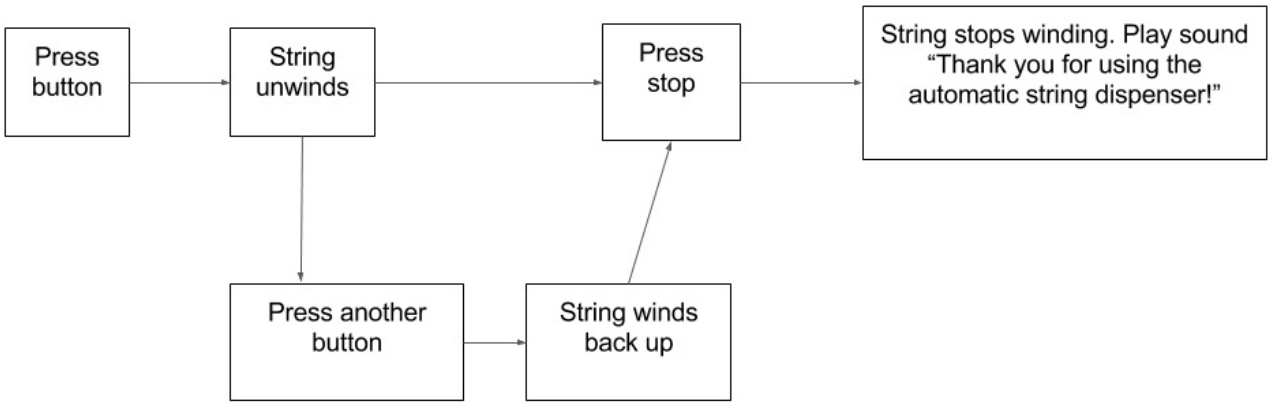
Try to draw a **flow chart** showing step by step how the project will work, or how a person will interact with it. This will help you decide how different parts of your project should be organised and connected. What is the first thing that the user will see? What will happen when they start using your project? Will they click on something? Will your project use feedback to tell the user things, such as sounds, flashing lights, or messages? What happens when the person has finished using it?

For example, the flow chart for an automatic string dispenser might look like this:

CoderDojo

Design a project

I'm learning: Project Design



Next, sketch out what your project will look like in action.

If you're making a software project, such as an app, a website, or something in Scratch, then draw the screens you'll create, or some possible layouts of a web page. Will your user need to **navigate** (find their way around different screens), and if so, how will they do that? For example, maybe you'll have a **navigation** menu, or things that the user will click or swipe, or perhaps the person will press keys on their keyboard, or even use voice commands.

For a hardware project, draw sketches of all the parts you'll be building and the circuits you need to make. Are you designing a robot? What will it look like?

Finally comes the part where you actually make something! Based on your designs, create the first **prototype** of your project.

- A **prototype** is an early version of a project that is made for testing out an idea.

You don't have to build the whole project at this stage. You might only make one tiny part or a simple version of it for now. For example, if you are making an animation or an app, you might just create one screen that does one little thing. The idea is to do just enough so that someone can test it out and you can see whether you're on the right track.

Why not just build the whole thing at once?

It would be an awful shame if you spent a lot of time developing a whole project, and then found out after you were done that it didn't solve the user's problem or they couldn't use it! By building your project bit by bit and testing it as you go along, you find out early on if you need to change something.

5 Test and tweak

Once you have a working prototype, get a user to **try it out**, or try it out yourself.

The aim of testing is to get **feedback** — opinions and suggestions. Are you on the right track with the project? Is it doing what it is supposed to do? So far maybe it only does a part of its job — does it do that correctly? Does it look how you want it to look?

Is the person you've asked to test your project using it the way you expected them to? If not, why not? What should you change?

If the person testing your project has problems or gets stuck, don't help them straight away. You won't be there to explain things every time someone uses your project. Instead, make a note of it – that could be a place where you can make improvements!

Try to write down answers to some of the questions above and anything else you learned from the test. Then plan what you need to do next. If you go back and do more building, be sure to test again afterwards!

You're doing great!

Remember, the user is the person you are making this project for. Don't be upset if they don't like something. All feedback is good feedback, because it will help you to come up with the best possible solution.

Iterate!

Keep doing a small chunk of work at a time and then testing your project again. For example, add a new screen or button, work on a bit of movement or sound, or whatever else makes up your project. This is called **iterating**. Gradually, you will build up all the pieces and end up with a finished product. Test out the finished product, too! Keep making changes and testing again until you and your user are satisfied.

- Coders and designers all over the world work like this, too. Many of the applications you use probably went through lots of **iterations** before they were done. In fact, many projects are never fully complete! Think about applications or websites that you've seen: they sometimes get updates, right? Each update is the result of another iteration of testing and building.