Grades 6–8

wonder workshop

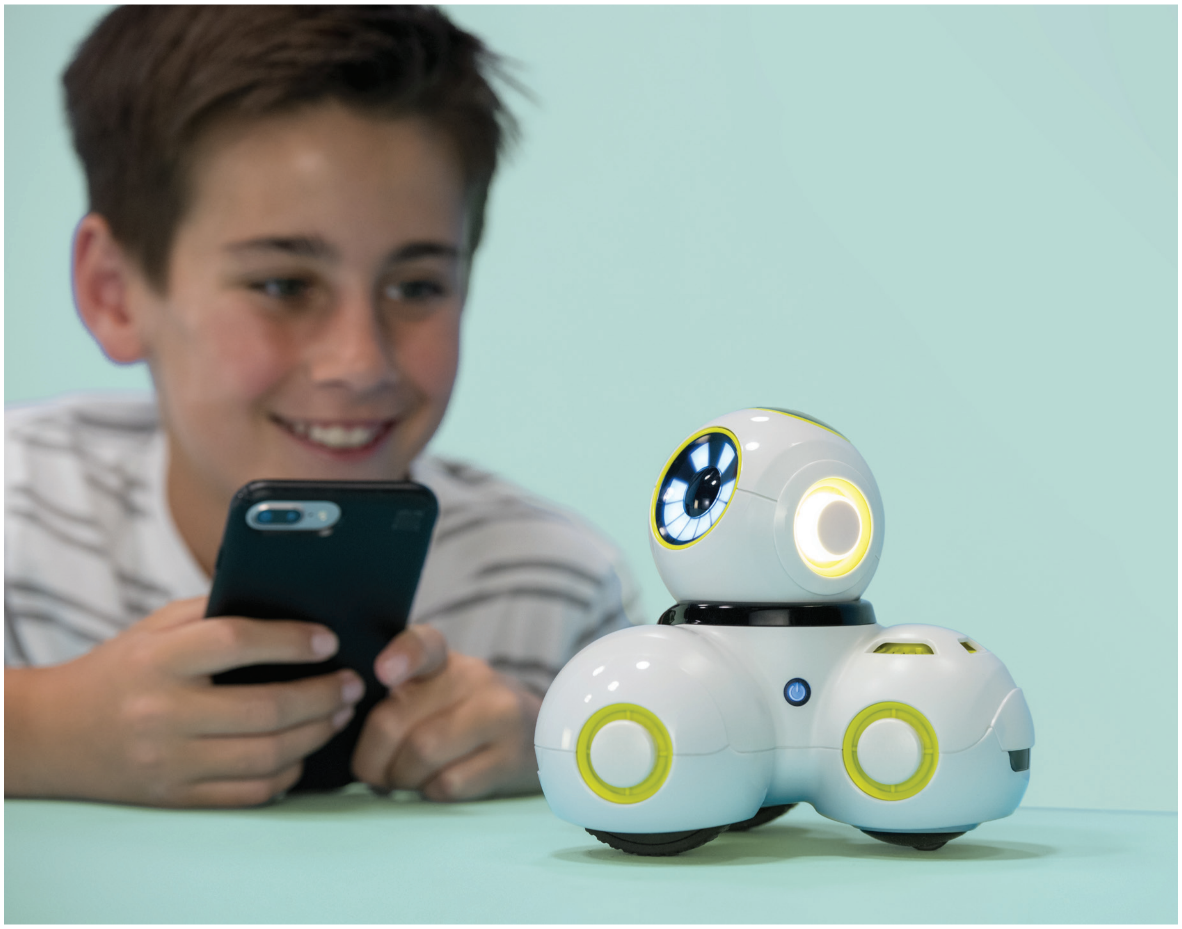
APPLIED ROBOTICS

Curriculum Guide

cue™

Your robot. Your rules.

Everything educators need to know to get their students coding with Cue!



Applied Robotics Curriculum Guide

Unit 1: Creative Writing
Unit 2: Game Design
Unit 3: Innovation

Grades 6–8

wonder workshop
clever together

Credits & Acknowledgements

Producer: Charlotte Cheng, June Lin
Editors: Charlotte Cheng, April Crane
Curriculum Authors: Fady Andraws, Charlotte Cheng, Matt Hesby, Mina Shenouda
Lesson Authors: Fady Andraws, Matt Hesby, Mina Shenouda
Illustration: Tim Reardon, Dag Haile
Design: Chris Cast, Desiree Jue, Katlin Li
Education Team at Wonder Workshop: Mike Lorion, Arnab Basu, Charlotte Cheng, April Crane, Naomi Harm, Katrina Keene, Jeff Mao, Bryan Miller, Dylan Portelance, Hetav Sanghavi, Darri Stephens, Timothy Tomaso, Amy Wilson

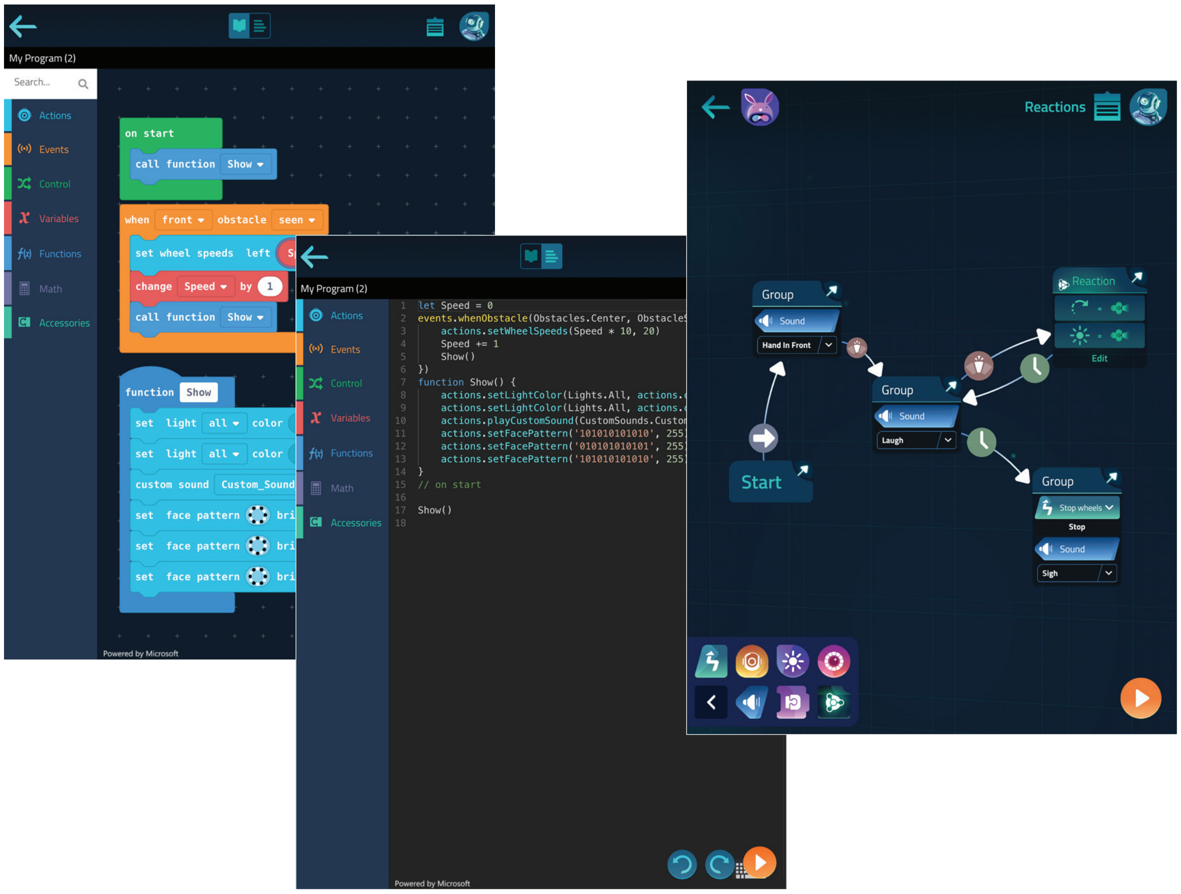
Wonder Workshop, Inc.
1500 Fashion Island Blvd., #200
San Mateo, CA 94404, USA

Support:
help.makewonder.com
support@makewonder.com

Welcome to the world of coding and robotics with Wonder Workshop!

We're excited that you've chosen our robot, Cue, for your classroom. When used with our app, Cue can bring coding and STEAM to life in a collaborative, interactive, and intuitive way!

In this guide, you will learn about the **Applied Robotics Curriculum for Cue**, including how to use our robots and curriculum content, implementation approaches, project-based assessment strategies, and cross-curricular extensions.



```
let Speed = 0
events.whenObstacle(Obstacles.Center, ObstacleS
    actions.setWheelSpeeds(Speed * 10, 20)
    Speed += 1
    Show()
})
function Show() {
    actions.setLightColor(Lights.All, actions.c
    actions.setLightColor(Lights.All, actions.c
    actions.playCustomSound(CustomSounds.Custom
    actions.setFacePattern('101010101010', 255
    actions.setFacePattern('010101010101', 255
    actions.setFacePattern('101010101010', 255
}
// on start

Show()
```

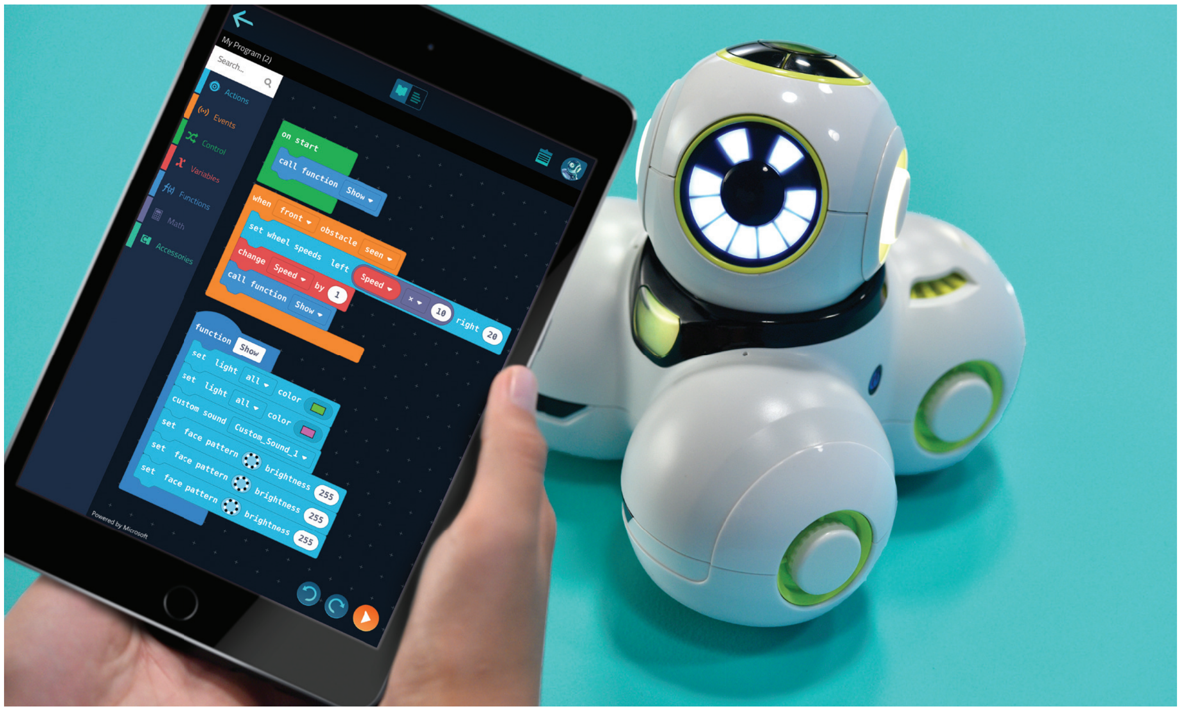
Teaching Coding with Robotics

Why Coding?

Learning to code helps students develop essential 21st-century skills!

Through coding, students learn to see the world through the lens of **computational thinking**. Computational thinking enables students to decompose problems, recognize patterns, and understand abstract concepts. Students can apply these skills to their everyday lives.

Coding is also a new type of literacy that prepares students for the careers of tomorrow. Whether it's a career in graphic arts or financial consulting, more and more occupations require employees to have at least a fundamental understanding of reading and writing code. Coding also prepares students for success in high-demand careers in STEAM (Science, Technology, Engineering, Art, and Mathematics).



Why Robotics?

Robots are an integral part of students' everyday lives. From a robo-vacuum that does chores to the vending machine where they buy snacks, robots are everywhere around them.

A **robot** is a machine that can gather information about its environment and use that information and any pre-programmed instructions to complete tasks. Robots are controlled by code to perform these instructions. These could be very simple and specific, like completing a set of movements that would cut a piece of wood to a certain length, or extremely complex, like teaching the robot how to respond in new situations.

Robotics brings coding and STEAM to life. It provides great opportunities for students to develop and apply their coding skills through hands-on experiences. Using our robot, Cue, provides students with instant feedback and engages their spatial awareness. With complex sensors, sounds, movements, and even the ability to program in JavaScript, Cue helps students see how they can solve problems and express ideas through the power of code!

Meet Cue the Robot

Cue is the newest member of Wonder Workshop's robot family. Like Dash and Dot, Cue is a robot that comes alive with animations, sounds, and lights to engage students' creativity.

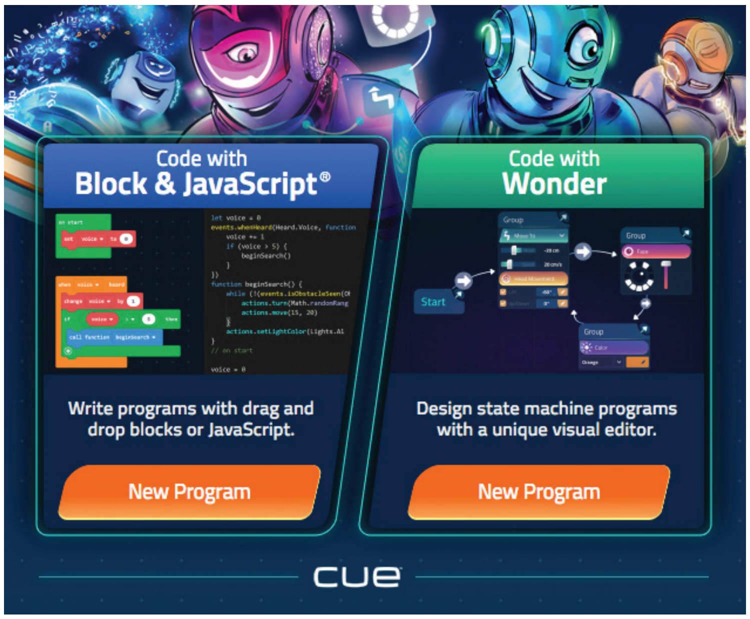
Cue is a level up from Dash and Dot in both robotic and programming capabilities. It comes equipped with more advanced sensors, motors, and processing power. Furthermore, students can program Cue to perform multiple commands at once and even program Cue using JavaScript!

Here are a few more similarities and differences between Cue, Dash, and Dot:

wonder workshop clever together	cue	dash	dot
Recommended Age Range	11+	6+	6+
Robot Capabilities			
Personality	Customizable	Unique for Dash	Unique for Dot
Voice Recording & Playback	●	●	●
Detects Voice Direction	●	●	
Robot Control	●	●	●
Drives, Turns, & Moves Head	●	●	
Detects Objects (Front & Behind)	●	●	
Volume Control	●		
Light Brightness Control	●		
Accelerometer	●	●	●
Gyroscope	●	●	
Play Time (battery fully charged)	Several Hours	Several Hours	Several Hours
Coding/User Experiences			
Fast Sensor Data Response	●		
State Machine Programming	Wonder (in the *Cue* App)	*Wonder* App	*Wonder* App
Parallel Programming (performs more than one action at a time)	●		
Reactive Behaviors	●		
Block-Based Coding	●	●	●
JavaScript® Coding	●		
Pre-reader Apps		*Go, Path, Xylo*	*Go*
Accessories			
Sketch Kit	●	●	
Launcher		●	
Xylo		●	
Building Brick Connectors	●	●	●
Creativity Pack			●

The *Cue* App

The *Cue* app is full of voice and choice! Students can choose to code with either **Block & JavaScript**®or **Wonder**.



```
let voice = 0
events.whenHeard(Heard.Voice, function
    voice += 1
    if (voice > 5) {
        beginSearch()
    }
})
function beginSearch() {
    while (!(events.isObstacleSeen(O
        actions.turn(Math.randomRang
        actions.move(15, 20)
    }
    actions.setLightColor(Lights.Al
}
// on start

voice = 0
```

Write programs with drag and drop blocks or JavaScript.

Design state machine programs with a unique visual editor.

The **Code with Block & JavaScript** section is where students can transition back and forth from block-based programming—similar to what's used in the *Blockly* app for Dash and Dot—to text-based programming using JavaScript. Students get to decide when to try more advanced robot capabilities with text-based programming.

Similar to the *Wonder* app for Dash, the **Code with Wonder** section provides a revolutionary, flow-based programming language where students can arrange their code into patterns that are similar to flowcharts or de cision trees. With this type of programming, students can create state machines like what robotics engineers use in their work. State machines are a series of robot states and conditions that help the robot process information and make decisions.

There are also four different avatar personalities for students to select in the app. Each personality has unique sounds and animations that can infuse students' programs with humor and inspire their storytelling. Students can also make their own personalized recordings with the robot, turning the app into a true tool of expression.

Setting Up Your Classroom

Robots, Apps, and Tablets

For our **Applied Robotics Curriculum**, we recommend assigning two students per robot. In pairs, students can share their ideas, take turns, and work together to complete activities and projects with Cue.

Before introducing Cue to the classroom, be sure to:

- ☐ Download the *Cue* app onto each tablet or device.
- ☐ Connect each robot to the *Cue* app and give each robot a name in the app.
- ☐ Use masking tape and/or a marker to label the robots with their names.
- ☐ Fully charge each robot.
- ☐ Fully charge and label each tablet or compatible device (e.g., Tablet #1, Tablet #2).
- ☐ Set up student accounts for each group that they can use to log into the *Cue* app.
- ☐ Create floor space for each group's robot and materials. For our activities and projects, we recommend providing at least a 4 x 4 sq. ft. space for each robot.

To help student groups keep track of their work, make sure each group uses their assigned username and password to log in to the *Cue* app.

You can find setup, care, and maintenance tutorials for our robots here:
www.education.makewonder.com/professional-development.

Additional Materials

Some of our activities and projects require students to use additional materials.

To complete the challenges, we recommend that you have the following materials available in the classroom:

- **Design Process Notebooks** (1 per student)
- **Sketch Kit** marker attachments (1 per group)
- **Sketch Kit** dry erase markers (1 set per group)
- **Wonder Whiteboard Mat** (1 per 2–3 groups)
- painter's or masking tape (1 roll per group)
- cardboard, construction paper, and/or other recycled materials to create props and sets

For additional robots, accessories, **Whiteboard Mats**, and **Notebooks**, you can shop at our online store: **https://store.makewonder.com**.

Making It Work

We recognize that classrooms have different limitations and encourage you to be creative with the resources you have!

If you are missing some of the materials listed, provide alternative resources. For example, instead of using the **Sketch Kit**, students can create their own drawing accessory using LEGO™ bricks and our **Building Block Connectors**. As an alternative to using the **Wonder Whiteboard Mat** for their robots' drawings, students can use butcher paper.

Curriculum Design Philosophies

Older and more advanced students are ready to be more independent with their learning. They look for more agency in their activities. They also value exploring concepts that are tied to real-world applications.

That is why the **Applied Robotics Curriculum** is built upon the foundations of **project-based learning** and **design thinking**. These two philosophies help students take control of their own learning while collaboratively applying concepts to projects and problems that they care about.

Project-Based Learning

Project-based learning is a teaching philosophy in which students work on projects that matter to them. While defined around an educational purpose, students still have choice in the content of their project. They are invested in their project's results and can apply what they've learned to something that is exciting and engaging to them.

The Design Thinking Process

The **Design Thinking Process** is a problem-solving and design method that many industries use to develop their ideas. From authors to architects and engineers to game designers, industry professionals understand that iteration and revision is crucial to creating a compelling product. The **Design Thinking Process** helps people design, develop, and iterate on their ideas in an efficient and effective way.

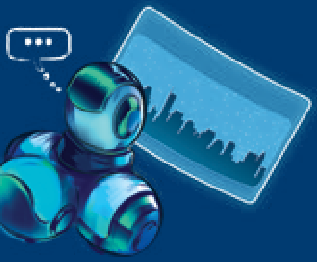
First, designers must **understand** the audience they are serving. Then, they **define** the focus of their project by selecting the specific problem they're trying to solve or message they want to communicate to their audience. Next, designers **ideate** by brainstorming many solutions or ideas with that defined focus in mind. Finally, they **plan** and **build** a prototype based on one of their ideas and **test** it with their audience. Designers then repeat these steps again and again to refine their idea until it is ready to be shared with the world.

By having students experience the **Design Thinking Process** as they work on projects with Cue, they gain real-world problem-solving and design skills that can be used in any industry or profession they choose. Students also learn the value of developing and refining an idea that they're invested in.

THE DESIGN THINKING PROCESS

1. UNDERSTAND

Who is your audience? Spend time interviewing and researching your audience to understand their needs and interests.



2. DEFINE

Define the problem you want to solve or the message you want to share.

What have others done to try to solve that problem or share a similar message?

3. IDEATE

Brainstorm solutions and ideas.

Think outside the box!

4. PLAN

Select a few ideas and develop designs.

Decide what materials you'll need.

5. BUILD

Build a prototype of your design that represents your ideas.

6. TEST

Test your prototype.

Get feedback from users to improve your design.

Applied Robotics Curriculum for Cue

In our **Applied Robotics Curriculum for Cue**, we outline a recommended action plan for implementing robotics and coding through the power of design thinking and project-based learning.

This plan offers a variety of teaching tools that include lesson plans, in-app content, student-facing notebooks, a solution guide, and authentic assessments.

This curriculum is designed for students who have had some exposure to block-based coding languages, like those found in *Scratch* or *Blockly*. These students are ready to explore coding concepts and robot capabilities in more depth.

Our **Applied Robotics Curriculum** is organized into units. Each unit introduces new robot capabilities that are tied to fundamental coding concepts. Students explore these capabilities in depth, using different programming paradigms. They then apply these capabilities to their long-term project for that unit.

The themes of the **Applied Robotics Curriculum** are as follows: **Unit 1: Creative Writing, Unit 2: Game Design,** and **Unit 3, Innovation**. The curriculum also covers the following robot capabilities and coding concepts:

Our Scope and Sequence

Our Scope & Sequence progresses students through these fundamental coding concepts:

	Unit 1	Unit 2	Unit 3
Theme	Creative Writing	Game Design	Innovation
Recommended Grade Level	6th Grade	7th Grade	8th Grade
Design Thinking Process	•	•	•
Block-Based Programming	•	•	
State-Machine Based Programming	•	•	•
JavaScript Programming	•	•	•
Computing Systems	•	•	•
Data & Analysis		•	•
Sequences	•	•	•
Events & Sensors	•	•	•
Loops	•	•	•
Functions		•	•
Variables		•	•
Conditionals		•	•
Arrays			•
Booleans		•	•

The curriculum also covers the following cross-curricular concepts:

	Unit 1	Unit 2	Unit 3
Theme	Creative Writing	Game Design	Innovation
Recommended Grade Level	6th Grade	7th Grade	8th Grade
Language Arts			
Reading	•	•	•
Writing	•	•	•
Speaking & Listening	•	•	•
Math			
Geometry	•	•	•
Ratios & Proportions		•	•
Algebra		•	•
Statistics & Probability		•	•

Lesson Plans

Each unit is paired with free lessons that are available online at:
https://education.makewonder.com/curriculum/appliedrobotics

These lesson plans include the following elements:

- **Warm Up:** Connect to students' prior coding and robotics knowledge with introductory exercises and discussion questions.

- **Direct Instruction:** Introduce new coding and robotics concepts through modeling and demonstrations.

- **Guided Practice:** Guide students through a group activity involving in-app content and/or unplugged exercises.

- **Independent Practice:** Have students complete **in-app challenges** and work with their **Design Process Notebooks** to practice coding concepts and apply them to their long-term project.

- **Wrap Up:** Close the lesson with student presentations and/or wrap-up discussions.

These lessons incorporate connections to *Code.org's* **CS Discoveries** course and are designed to meet *Computer Science Teacher Association* (CSTA) and *International Society for Technology in Education* (ISTE) standards. More information about these standard correlations can be found within each lesson plan.

We recognize that each classroom's schedules and needs differ. Thus, we recommend that you view these lesson plans as a resource from which you can pick and choose the content that works best for your students.

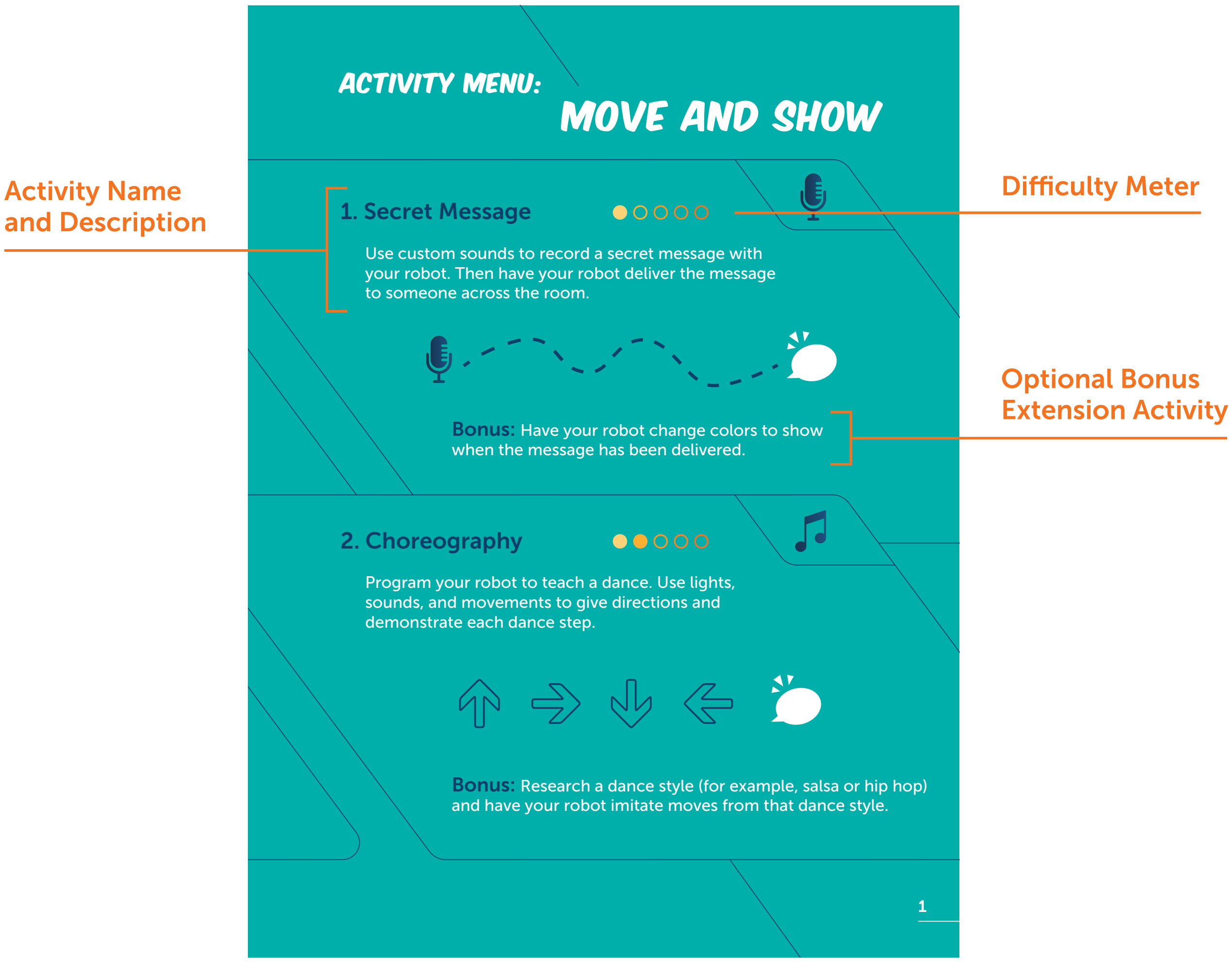
Design Process Notebook

The **Design Process Notebook** guides students through activities as they learn about Cue's robotic and coding capabilities. They then apply what they've learned to create a long-term project of their choice using the **Design Thinking Process**.

Activity Menus

After students are introduced to new coding concepts and robot capabilities through our **in-app challenges**, they choose one of four different open-ended challenges from an **Activity Menu**. The activities in each of these menus are focused on a specific coding concept and robot capability.

For example, after students learn how to program Cue to move, show lights, and play sounds, they select a challenge from **Activity Menu: Move and Show** in their **Design Process Notebook**.


Activity Name and Description
Difficulty Meter
Optional Bonus Extension Activity

ACTIVITY MENU: MOVE AND SHOW

1. Secret Message ●○○○○

Use custom sounds to record a secret message with your robot. Then have your robot deliver the message to someone across the room.

Bonus: Have your robot change colors to show when the message has been delivered.

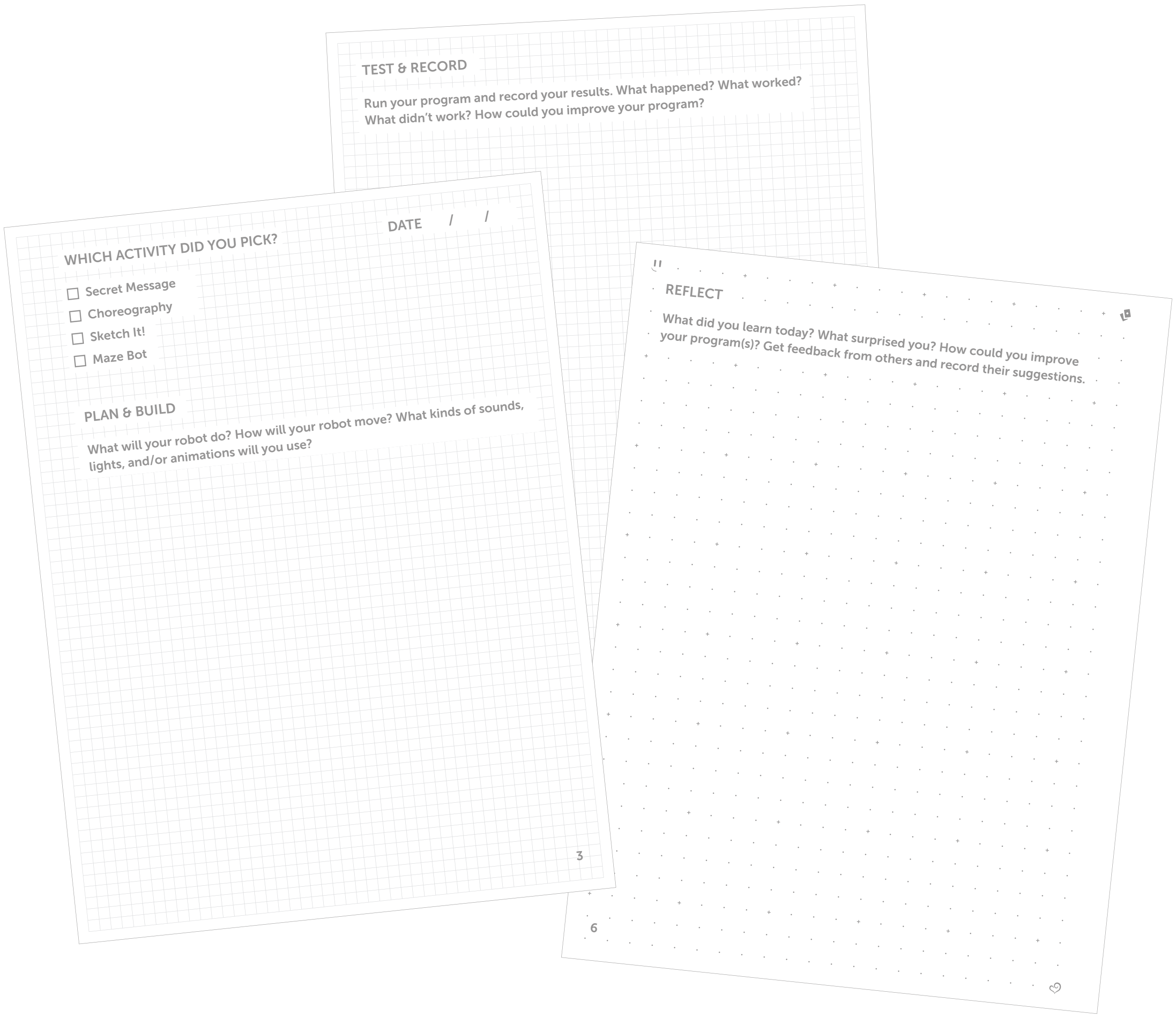
2. Choreography ●●○○○

Program your robot to teach a dance. Use lights, sounds, and movements to give directions and demonstrate each dance step.

Bonus: Research a dance style (for example, salsa or hip hop) and have your robot imitate moves from that dance style.

1

Each **Activity Menu** is followed by graph pages with prompts to help students plan out, test, record, and reflect on the programs they develop to complete each activity.



TEST & RECORD

Run your program and record your results. What happened? What worked? What didn't work? How could you improve your program?

WHICH ACTIVITY DID YOU PICK?

DATE / /

☐ Secret Message
☐ Choreography
☐ Sketch It!
☐ Maze Bot

PLAN & BUILD

What will your robot do? How will your robot move? What kinds of sounds, lights, and/or animations will you use?

3

REFLECT

What did you learn today? What surprised you? How could you improve your program(s)? Get feedback from others and record their suggestions.

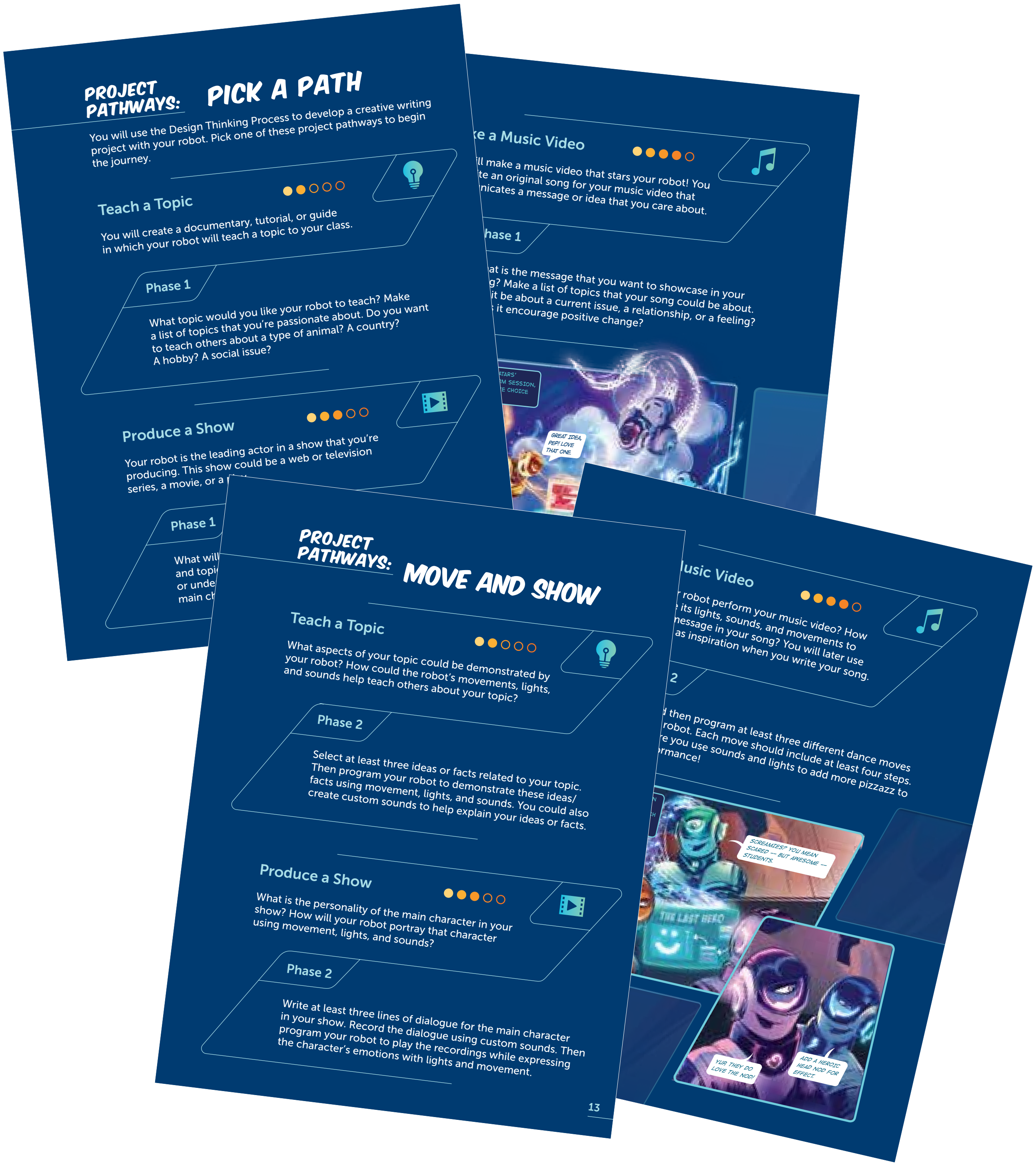
6

For each activity in the **Activity Menu**, you will find hints and suggested solutions in this guide (pages 30–70). These can help you support and coach students as they work on the activities.

Project Pathways

After getting more comfortable with new coding concepts and robot capabilities, students apply them to one of three **Project Pathways**, which guide students through the development of a long-term project with the robot. Each of the **Project Pathways** is broken down into a series of project phases.

Phases 1 and 2 of **Project Pathways**, for example, involve selecting a project path and then exploring how to use the robot's movements, lights, and sounds as part of their project.



PROJECT PATHWAYS: PICK A PATH

You will use the Design Thinking Process to develop a creative writing project with your robot. Pick one of these project pathways to begin the journey.

Teach a Topic ●●○○○

You will create a documentary, tutorial, or guide in which your robot will teach a topic to your class.

Phase 1

What topic would you like your robot to teach? Make a list of topics that you're passionate about. Do you want to teach others about a type of animal? A country? A hobby? A social issue?

Produce a Show ●●●○○

Your robot is the leading actor in a show that you're producing. This show could be a web or television series, a movie, or a f

Phase 1

What will
and topi
or unde
main ch

ke a Music Video ●●●●○

ll make a music video that stars your robot! You
te an original song for your music video that
nicates a message or idea that you care about.

hase 1

at is the message that you want to showcase in your
g? Make a list of topics that your song could be about.
it be about a current issue, a relationship, or a feeling?
it encourage positive change?

PROJECT PATHWAYS: MOVE AND SHOW

Teach a Topic ●●○○○

What aspects of your topic could be demonstrated by your robot? How could the robot's movements, lights, and sounds help teach others about your topic?

Phase 2

Select at least three ideas or facts related to your topic. Then program your robot to demonstrate these ideas/facts using movement, lights, and sounds. You could also create custom sounds to help explain your ideas or facts.

Produce a Show ●●●○○

What is the personality of the main character in your show? How will your robot portray that character using movement, lights, and sounds?

Phase 2

Write at least three lines of dialogue for the main character in your show. Record the dialogue using custom sounds. Then program your robot to play the recordings while expressing the character's emotions with lights and movement.

13

usic Video ●●●●○

r robot perform your music video? How
its lights, sounds, and movements to
message in your song? You will later use
as inspiration when you write your song.

2

d then program at least three different dance moves
robot. Each move should include at least four steps.
re you use sounds and lights to add more pizzazz to
rmance!

SCREAMIES? YOU MEAN SCARED — BUT AWESOME — STUDENTS.
THE LAST HERO
YUR THEY DO LOVE THE NOD!
ADD A HEROIC HEAD NOD FOR EFFECT.

Each phase in the **Project Pathways** encourages students to use the **Design Thinking Process** to ideate and iterate on their project. Each phase also includes graph pages with prompts to help students plan out, test, record, and reflect on the phases of their project.

DATE / /

WHICH PROJECT DID YOU PICK?

☐ Teach a Topic
☐ Produce a Show
☐ Make a Music Video

IDEATE

What topic will your project be about? What do you care about? List your ideas here. Think outside the box!

9

ur list of ideas and choose the best one. Describe and/or
ne idea in more detail here. Why did you pick this idea?
c reasons in your explanation.

RESEARCH

Research your topic by exploring books, videos, and/or websites about that topic. Write down your observations.

11

REFLECT

rn? Did you find anything particularly interesting or
could you incorporate these ideas into your project?

Notebook Tips & Tricks

Here are several tips and tricks that you can share with your students as they work in their **Design Process Notebooks:**

Determine team roles: Swap roles with your teammates for each challenge. Team roles include lead programmer, robot wrangler, and documentarian.

Plan your path: Use the graph pages in your **Notebook** to draw out the path or movements you want your robot to make. Use those drawings to help you plan out the program you will create in the *Cue* app. You can also test possible solutions by getting up and walking the path or imitating the movements you want Cue to make.

Mark your key spots: Use masking or painter's tape to mark your robot's starting spot and the location of any obstacles/objects that are part of your activity, challenge, or project (in case they move).

Go back to start: Always put your robot back at the starting spot before playing a program again.

Going backwards: If you want Cue to move backwards, use negative numbers when programming Cue's movements.

Think in centimeters: Your robot moves in centimeters. Use the ruler on the back of your **Notebook** to measure out distances. You can also refer to the **References & Resources** section to findouthowtoconvertinchestocentimeters.

Check off the steps: Use the blank paper in your notebook to list of each step you want the robot do as part of your program. You can then check off each item on the list once your robot completes that task.

Help your robots hear you: If the classroom is noisy, use the **when clap heard** event instead of the **when voice heard** event. You can also ask the teacher for permission to try out your program with Cue outside or in the hallway.

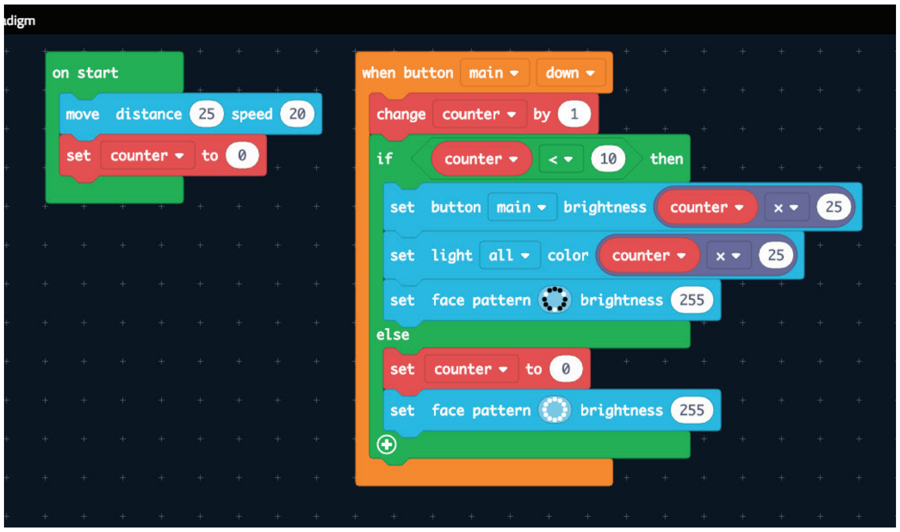
Up the challenge: When you've finished an activity from the **Activity Menu**, challenge yourself by adding more to your program or trying out another activity.

Programming Paradigms

The *Cue* app allows students to program Cue using two different paradigms:

Code with Block & JavaScript

In the **Code with Block & JavaScript** section, students begin with block-based programming, similar to what is used in the *Blockly* app for Dash and Dot or *Code.org's* **CS Fundamentals** course. With these programs, the Cue robot executes commands starting from the top and then moving down.

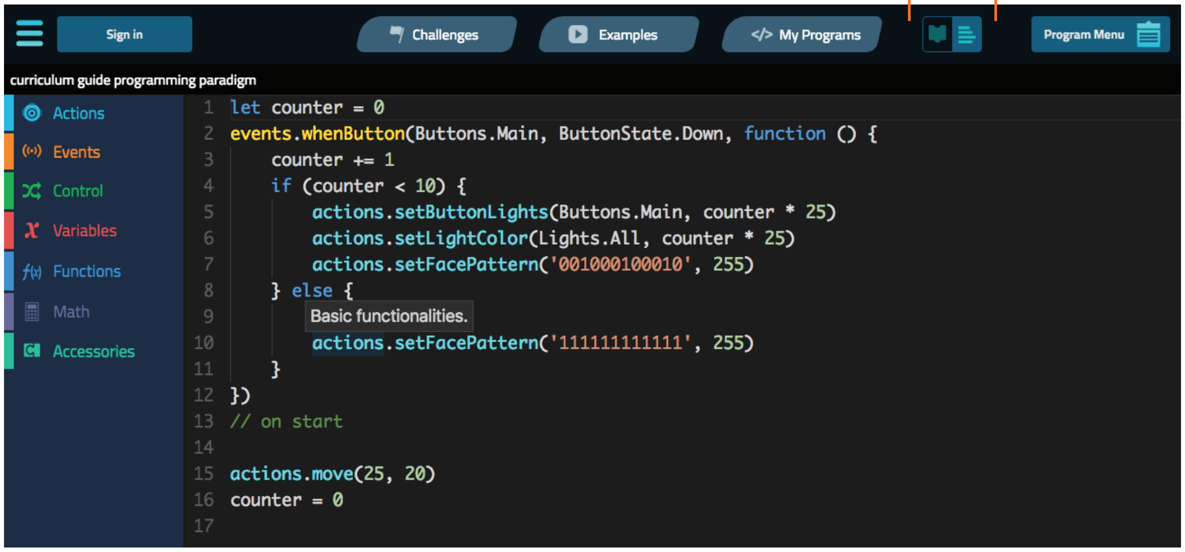


```
on start
    move distance 25 speed 20
    set counter to 0

when button main down
    change counter by 1
    if counter < 10 then
        set button main brightness counter × 25
        set light all color counter × 25
        set face pattern brightness 255
    else
        set counter to 0
        set face pattern brightness 255
```

With the touch of a button, students can then toggle to translate their program into JavaScript. JavaScript programs are also executed by starting from the top and then moving down. They enable students to unlock additional capabilities for the robot, such as inputting specific RGB colors for the robot's lights or creating more complex functions with parameters.

Blocks to Text Toggle

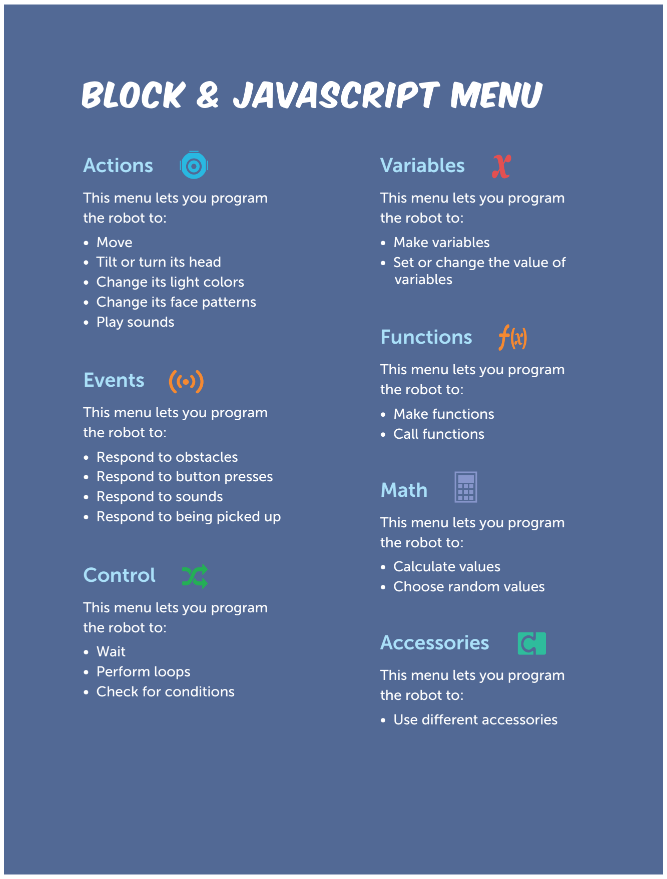


```
let counter = 0
events.whenButton(Buttons.Main, ButtonState.Down, function () {
    counter += 1
    if (counter < 10) {
        actions.setButtonLights(Buttons.Main, counter * 25)
        actions.setLightColor(Lights.All, counter * 25)
        actions.setFacePattern('001000100010', 255)
    } else {

        actions.setFacePattern('111111111111', 255)
    }
})
// on start

actions.move(25, 20)
counter = 0
```

Whether they use blocks or JavaScript, students can access a menu of different commands.



BLOCK & JAVASCRIPT MENU

Actions

This menu lets you program the robot to:

- Move
- Tilt or turn its head
- Change its light colors
- Change its face patterns
- Play sounds

Events

This menu lets you program the robot to:

- Respond to obstacles
- Respond to button presses
- Respond to sounds
- Respond to being picked up

Control

This menu lets you program the robot to:

- Wait
- Perform loops
- Check for conditions

Variables

This menu lets you program the robot to:

- Make variables
- Set or change the value of variables

Functions

This menu lets you program the robot to:

- Make functions
- Call functions

Math

This menu lets you program the robot to:

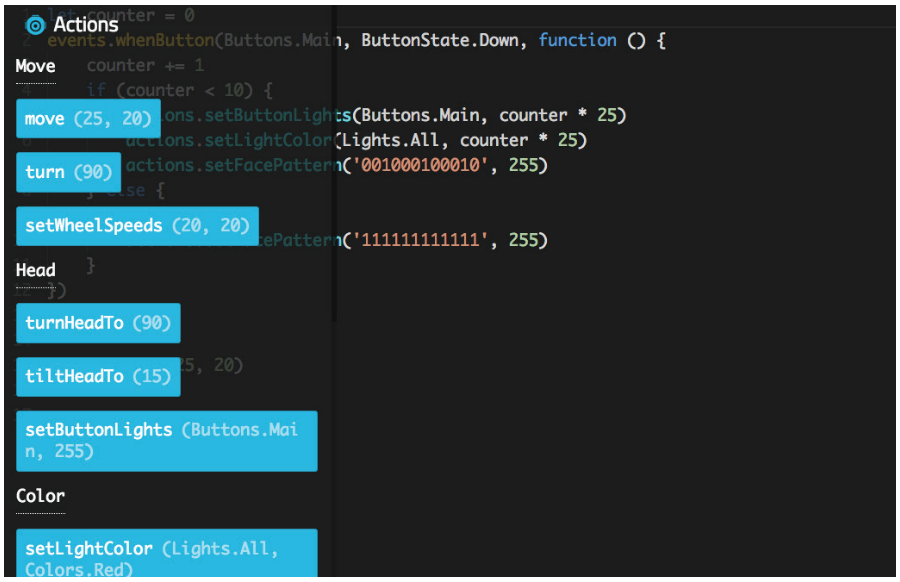
- Calculate values
- Choose random values

Accessories

This menu lets you program the robot to:

- Use different accessories

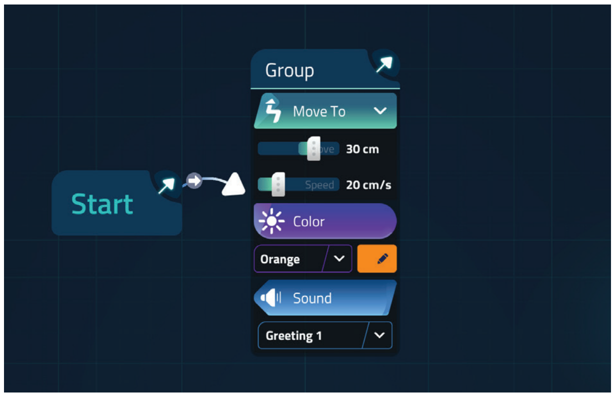
This menu can greatly help students when they begin programming in JavaScript. Students can use the menu to add a snippet of JavaScript code to their program and then adjust the parameters without worrying about syntax. Syntax issues, such as typos or not having the correct number of parentheses, are one of biggest stumbling blocks for students when they begin programming using text-based languages.



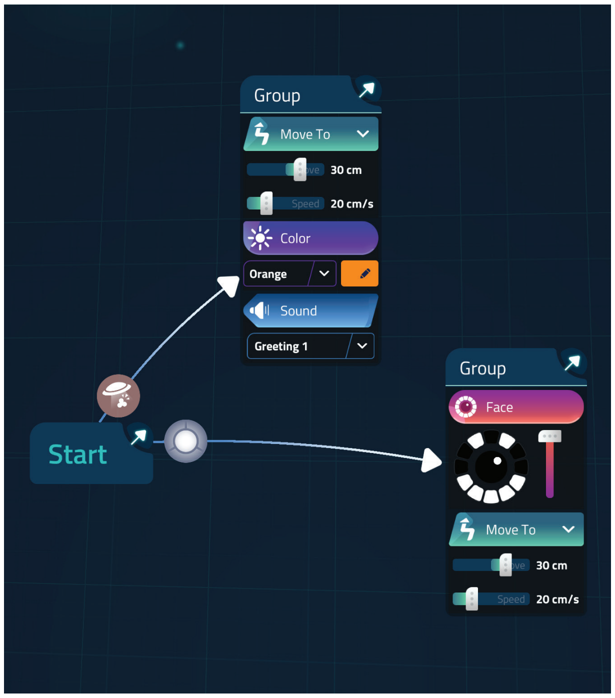
```
let counter = 0
events.whenButton(Buttons.Main, ButtonState.Down, function () {
    counter += 1
    if (counter < 10) {
        ons.setButtonLights(Buttons.Main, counter * 25)
        actions.setLightColor(Lights.All, counter * 25)
        actions.setFacePattern('001000100010', 255)
    } else {
        ePattern('111111111111', 255)
    }
})
5, 20)
```

Code with Wonder

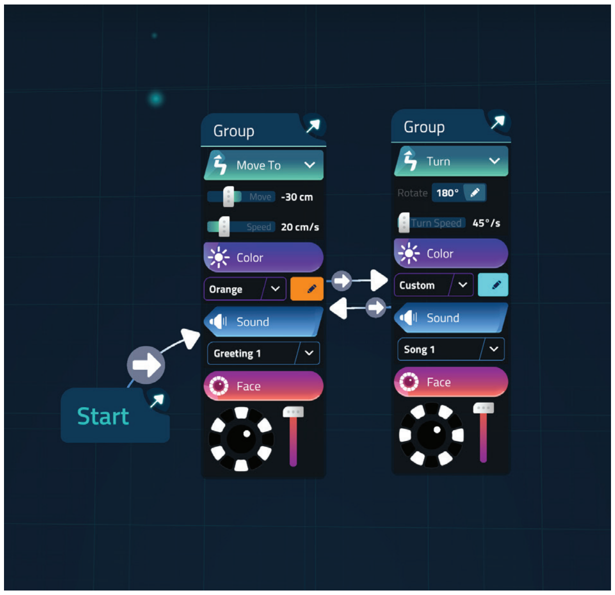
The **Code with Wonder** section provides a different programming paradigm for students to explore in the *Cue* app. Similar to the *Wonder* app for Dash and Dot, **Wonder** provides a revolutionary, flow-based programming language where students can arrange their code into patterns that are similar to flowcharts or decision trees.

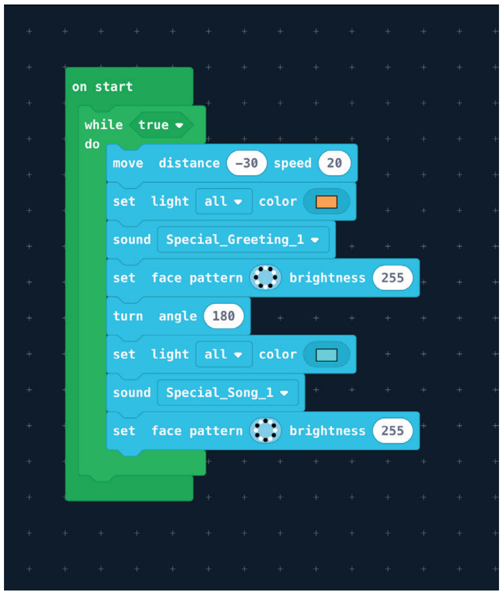


Students first begin by creating **groups**. Each group can contain multiple **actions** that the robot can perform at once. For example, in the program above, when the robot goes to the first group, it moves forward, changes colors, and plays a sound—all at the same time! This type of programming is known as **parallel computing**, since the robot can execute multiple commands at once.



Students can connect the groups they create by using transitions, or **events**, for the robot. These events can include whether the robot's button is pushed or whether it senses an obstacle. When the program starts, the robot moves from one group of actions to another based on whether or not these event conditions are met. In the program above, if the robot's button is pressed, its lights will change to orange. If the robot detects an obstacle, it will move forward. This method of programming is a great way to teach **conditionals**, as the decision tree illustrates how conditions help the robot make decisions on what to do next!





```
on start
  while true ▾
  do
    move distance -30 speed 20
    set light all ▾ color 🟧
    sound Special_Greeting_1 ▾
    set face pattern ⁘ brightness 255
    turn angle 180
    set light all ▾ color 🟦
    sound Special_Song_1 ▾
    set face pattern ⁘ brightness 255
```

Choosing Paradigms

For many occupations, it is essential for people to be able to assess a problem or challenge and then select a strategy that will best help them meet their objectives.

For example, engineers who build apps must make similar decisions. There are a wide variety of programming languages available (e.g., CSS, JavaScript, Python), and thus, engineers must choose which language will best help them achieve their objectives based on what they are trying to accomplish in the app.

In the **Design Process Notebook**, all activities and projects can be completed using either **Block**, **JavaScript**, or **Wonder-**based coding in the *Cue* app. However, some activities are easier to complete with **Block** & **JavaScript**, and some with **Wonder**. We encourage you to let students choose which programming paradigm they prefer to use for each activity and project. By providing students with a choice, you help them develop the ability to assess which paradigms work best for completing specific activities and projects.

The example above shows programs for the same activity completed in both **Block-** and **Wonder-**based coding. The activity requires students to program a dance for Cue. It is easier to program a dance with **Wonder** because the robot can perform multiple actions at once, whereas the robot can only perform one action at a time in using blocks.

Student groups can compare and contrast their experiences completing the same activity using **Block**, **JavaScript**, or **Wonder**. Encourage them to answer questions such as:

- Which activities are easier to complete with **Wonder**, and why?
- Which activities are easier to complete with **Block** & **JavaScript**, and why?

Assessment Strategies

Wonder Portfolios

We recommend providing an authentic assessment by having students create a **Wonder Portfolio**. In their portfolio, students can include programs they've created for different projects and activities. They can also include planning and reflection entries from their **Design Process Notebooks** and videos or documentation of their **Project Pathway** presentations.

CSTA, ISTE, NGSS, and Common Core standards all stress the importance of having students reflect on their learning process. When students reflect on their work, they're able to:

- Identify their strengths and weaknesses.
- Assess and learn from their mistakes.
- Improve their work through iterative design thinking.
- Identify any resources, guidance, or support they need.
- Create an informed plan for their next projects.

With these portfolios, students can also share the programs they've created and demonstrate what they've learned to their friends and family!

Wonder Portfolios can include:

Design Process Notebook Entries

Students can use their **Design Process Notebooks** to document their planning, results, and reflection while completing activities, challenges, and projects.

Programs

After students complete each activity or project phase, have them take a screenshot of their code. If needed, demonstrate for your class how to take screenshots with your classroom's tablets or devices. For activities selected from the **Activity Menus**, you can compare your students' programs to the suggested solutions we've provided in this guide.

Videos

Students can also take videos of Cue while running their programs. In this way, they can showcase any custom sounds or light patterns they've created. Additionally, the videos will allow you to assess whether their programs completely meet activity or project phase objectives.

After completing all of the phases, students will share their project with the class and highlight their **Design Thinking Process** in their **Project Pathway** presentation. Students can also include video and/or documentation of these presentations in their **Wonder Portfolios**.

Digital Portfolio Platforms

Digital portfolio platforms can help students assemble the different elements of their **Wonder Portfolios**. With these platforms they can:

- Share screenshots of their code.
- Take videos of their robots in action.
- Record results and reflect on their coding experiences.
- Provide feedback to one another.
- Share their **Wonder Portfolios** with friends and family.

Here are a few digital portfolio platforms that pair well with our curriculum:

Seesaw

- *Seesaw* offers various account sign-in options that provide easy access for all K–12 students, including those without email addresses.
- *Seesaw* also includes features like voice recording and mark-up tools for students to reflect on their work.

Google Classroom

- *Google Classroom* provides a great way for student groups to collaborate on a journal together. Multiple students can work on a document and provide feedback at the same time.
- *Google Classroom* allows students to organize their entries and multimedia files into folders. These folders can then be shared with specific students, groups, teachers, and family members.

Evaluation

To evaluate **Wonder Portfolios**, you can use:

Our Rubrics

Share the rubrics with students before they begin coding so that they are aware of your learning and performance expectations. Then use the rubrics to evaluate each student's portfolio. We think that life skills, such as communication and collaboration, are just as important as coding skills. Thus, we have incorporated assessment criteria for both academic and life skills into our rubric system. We also encourage you to adapt the rubrics to meet your students' needs.

Peer Review

Peer review is an important practice that is commonly used in diverse work environments. Your students can practice peer review with Cue, too. They can use our rubrics to assess each other's programs and portfolios while providing constructive feedback.

We recommend that you review best feedback practices with your class. These strategies include:

- Give one compliment and one suggestion.
- Be kind and considerate when giving feedback.
- Use concrete details and examples when giving suggestions.
- Explain why you like or don't like something about your classmate's project or program.
- Listen and write down the feedback you receive.
- Spend time to think how you can incorporate the feedback into your next program.

Implementation Strategies

We designed our **Applied Robotics Curriculum** and **Design Process Notebooks** to accommodate different classroom structures. This way, you can easily adapt the curriculum to meet the specific needs of your class.

Differentiated Instruction with Small Groups

If your students have a wide range of coding abilities, you can form groups based on their coding levels.

Have students who need more guidance or support complete only one activity per **Activity Menu**. They can also skip the later phases of the project, since these focus more on revision work. For their presentations, they can share one of the programs they created for the earlier project phases.

Have more advanced students complete multiple activities per **Activity Menu**. Encourage them to complete the bonus portion of each activity, as well. These students can complete multiple **Project Pathways** or spend more time refining the details of their presentations.

By providing these options, student groups can advance at their own pace, while you facilitate and provide support as needed with our lesson plan content.

Makerspace Stations

Our **Applied Robotics Curriculum** is also perfect for makerspaces. Set up a **Wonder Workshop** station in your makerspace with Cue robots, tablets, **Design Process Notebooks** (1 per student), a **Sketch Kit**, a **Wonder Whiteboard Mat**, and any additional materials that students may need (e.g., masking tape).

You can use our lesson plans to introduce new coding concepts or strategies at the beginning of each class and then let students rotate through the **Wonder Workshop** stations. Students can then progress through the **Notebook**'s activities and projects at their own pace.

Design Your Own Challenge

Another fun extension activity is to have students design their own activity menus. Students can attempt to complete each other's activities and give feedback to one another. This helps students practice their instructional writing skills and explore coding concepts in a different way.

Best Practices

Our **Applied Robotics Curriculum** works best in student-centered coding environments. These environments establish a classroom culture where students take control of their own learning! Here are a few suggestions for creating a student-centered coding environment in your own classroom:

Learn with Your Students

Technology is constantly changing and adapting to new platforms. You shouldn't feel as though you need to know everything about coding and robotics before you introduce them to the class. Don't be afraid to explore Cue along with your students. As you learn new coding concepts together, you'll be able to model good learning practices, such as finding answers and solutions through experimentation and trial and error.

Growth Mindset

In coding, students progress more quickly and effectively when they learn from their own mistakes. If students are afraid to fail, they are less likely to discover innovative ideas and designs. Encourage your students to try out different strategies as they work on the activities and projects in their **Design Process Notebooks**. Remind them to see their mistakes as progress and growth, and encourage them to "fail fast and fail forward!"

Collaboration

Cue is best explored in student pairs. By working in pairs, students get to practice 21st-century skills, such as collaboration, communication, and cooperation. These essential life skills are required and constantly practiced in modern-day work environments.

Encourage students to share the tablets and robots. Have them establish and rotate through roles such as:

- lead programmer: holds the tablet and integrates group member ideas to create the program in the *Cue* app
- robot wrangler: resets robot starting locations, sets up obstacles, and performs any needed measurements
- documentarian: records plans, results, and reflections in the **Design Process Notebook** and takes photos or videos of robots

When students work together while coding, they're able to help each other identify mistakes and develop creative solutions!

Troubleshooting

You and your students are bound to run into some complications when using new technology. Help students help themselves by reviewing a few troubleshooting strategies in advance:

If your program is not running correctly . . .

- make sure the Cue robot is turned on.
- make sure the Cue robot is connected to the app by tapping the icon at the top right of the screen.
- check the name on your robot. Make sure it's the same as the name of the robot connected to your app.
- make sure your blocks are connected to the **Start** block (unless they are event handlers).
- check to see if your program is already running. You need to stop the program before you can begin editing it.
- try restarting the app.

If the Cue robot is not connecting or is disconnecting . . .

- turn off the robot and turn it on again. Then reconnect the robot to the app by tapping the icon at the top right of the screen.
- check the battery level of the robot by tapping your robot's icon at the top right of the screen. You will then see a battery icon next to your robot avatar. If the battery is low, try charging your robot before connecting to it again.
- check the name on your robot. Make sure that other groups are not connected to your robot. If they are, have them disconnect before trying to connect your robot again.
- press play and then press stop to make the robot reset.
- back out of your program to the main menu and then revisit your program. This will help the program reset.

Three, then me!

- First, ask for help from three of your classmates. If you still need help, then ask the teacher.

Problem Solving and Debugging

Sometimes students can get stuck on a particular challenge or problem. They may have an error or "bug" in their code. They may also have only partially solved the problem or misunderstood the challenge.

To provide challenge-specific hints, refer to the **Solutions** section of this guide (pages 29–70). We also recommend that you review a few debugging strategies in advance to help students get unstuck on their own:

Break down the activity:

- What accessories or materials do you need for the activity?
- What is Cue supposed to do in the activity?
- Have you completed similar activities to this one?
- Focus on one step at a time.

Plan your code:

- Draw a picture or make a list of what you want Cue to do.
- Will you use **Block**, **JavaScript**, or **Wonder** to complete the activity?
- Which parts of the **Block**, **JavaScript**, or **Wonder** menu will you use to complete the activity?
- Are there any hints in the activity description that can help with your planning?
- Use tape to mark Cue's starting point.
- If there are obstacles in the challenge, use tape to mark each obstacle's location in case it gets moved.

Test your code:

- Does your code meet the objectives of your activity or project?
- If not, play your code again. Watch as the program goes through each line of code or action. Do you notice any mistakes?
- Are the lines or actions in your code in the correct order?
- Do you need to add more or remove any lines of code or actions?
- Are you starting Cue at the same position each time you run your program?
- Pretend to be Cue. Act out each line or action in your code and look for mistakes.

Improve your work:

- Ask another student or group to check your program and provide feedback.
- Is there an easier way to complete the activity? Can you use fewer lines of code?
- Would the activity be easier to complete in **Block**, **JavaScript**, or **Wonder**?
- Could you add more lights, sounds, or other customizations to your program?

You can find **Troubleshooting** and **Problem Solving and Debugging Handouts** in the **Appendix** of this guide (pages 73–74).

Wonder Workshop's **Applied Robotics Curriculum Guide** provides teachers of grades 6–8 with step-by-step, comprehensive guidance on bringing computer science to life in their classrooms with Cue robots. This guide includes lesson plans that introduce coding concepts with Cue robots, in-app content, student notebooks, and accessories. Before jumping into the exciting world of coding with hands-on robotics, teachers will also learn how to set up their classrooms to establish a student-centered culture that fosters collaboration, design thinking, growth mindset, and project-based learning.

To use the **Applied Robotics Curriculum Guide**, you'll need:

Cue robot

Cue app

Design Process Notebooks (Units 1, 2, and 3)

FREE Applied Robotics Lessons:
https://education.makewonder.com/curriculum/appliedrobotics

This guide is perfect for all educators—those who are just starting their coding journey and those who are looking to add Cue to their coding and robotics toolkit. For more information on how to get started, visit **education.makewonder.com**

Designed to meet with CSTA and ISTE standards.
Integrated with connections to *Code.org*'s **CS Discoveries** Course.

wonder workshop

Wonder Workshop, Inc.
1500 Fashion Island Blvd., #200
San Mateo, CA 94404, USA

Support:
help.makewonder.com
support@makewonder.com