BSD Education Workbook

COMPUTATIONAL THINKING

bsd | education

www.bsd.education

TABLE OF CONTENTS

bsd | education

Who Are We?

BSD stands for **B**uild **S**omething **D**ifferent.

We are an international team (Hong Kong, USA and Thailand) of social entrepreneurs, technologists and educators dedicated to empowering the kids of today with the tools for tomorrow. Our mission is to give every student access to technology education.

We partner with schools to build curated technology curriculum that fit into all subjects (e.g. History, Math, Science). Schools and teachers we work with have access to BSD's online learning platform, content and curriculum, as well as in-person or virtual training for all experience levels. This gives teachers the confidence, content and community they need to start embedding technology in their classroom today. Our goal is to get every student using the latest technology – including coding, design and robotics - to solve real world problems.

BSD wants every student to feel confident addressing today's problems with tomorrow's technology and feel empowered to "Build Something Different".

Our Mission

BSD is on a mission to make the learning and understanding of technology and design accessible to all students and teachers. Students that complete BSD projects will learn to **C.A.R.E.** about the future.

They will be:

CURIOUS
Always seeking to learn

ADAPTABLE
Never afraid to try something new

RESILIENT
Willing to start again and learn from challenges

EMPATHETIC
Thoughtful about how their technology impacts the world

We help students develop the right mindset to use technology to solve difficult problems. Students will have the confidence, empathy and skills they need to shape the future.

What Is This Workbook?

First, what is Computational Thinking? The way humans think of ideas are complex, intuitive, and can't always be explained. Computational thinking is the mindset of solving problems from the ground up:

Decomposing a problem into its smallest individual parts,
Recognizing the common patterns in those pieces,
Abstracting your ultimate goal so that it relates to those pieces, then
Creating an algorithm that puts the pieces into a reliable and reusable approach for solving that problem.

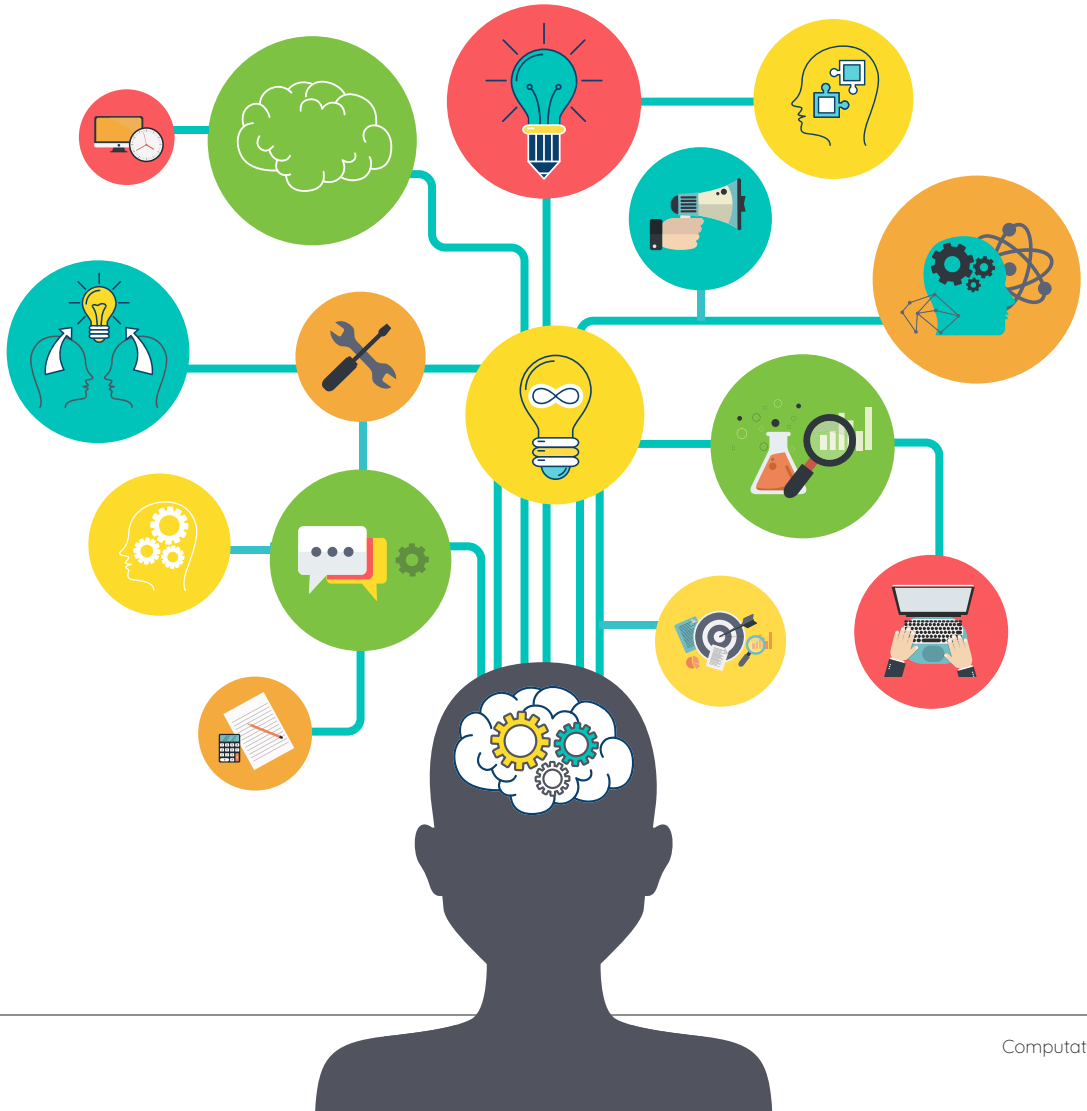
This process is how computers are built and run programs, and is increasingly vital in today's technical world. So, how can we prepare the next generation for the world of computers that we live in? Through this workbook, students will become familiar with the concept of computational thinking and will develop the skills required for success in the modern world. In this workbook, students will learn about the 4 main components of computational thinking and how to apply them in any situation, technical or not.

How To Use This Workbook

This workbook contains 7 Computational Thinking activities that largely do not rely on computers or the internet. Some lessons will reference web-games or videos as supplementary material, and some may require particular craft supplies. When reading the workbook overview, the estimated preparation time for each lesson is given for the instructor. Lessons are also marked if they have notable requirements:

- computers,
- special supplies,
- student teamwork, or
- access to the outdoors

When preparing for each lesson, note these indicators, and pay attention to the supplies list for each activity. Most lessons will require standard classroom craft supplies such as paper, colored pencils, scissors, tape, glue, markers, etc. Many lessons include external resources for the instructor's reference inside each lesson plan. This workbook can be used in classrooms by teachers or at home with parental guidance. If you are a parent using this workbook, follow along with the activities in the workbook with your child by leading them through each lesson.



BEGINNER

Recommended for Grades 3-6

ACTIVITY 1: 20 QUESTIONS

Schedule

HOUR	AGENDA	DURATION
1	20 Questions Game	40 mins.
	Sharing: 1 thing each student has learned today	15 mins.

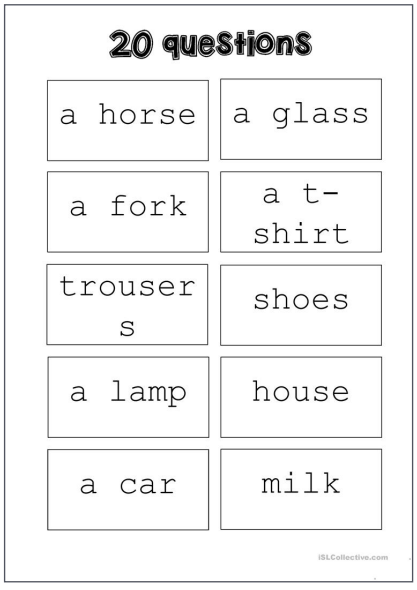
20 Questions-Game (40 minutes)

Learning Objectives:

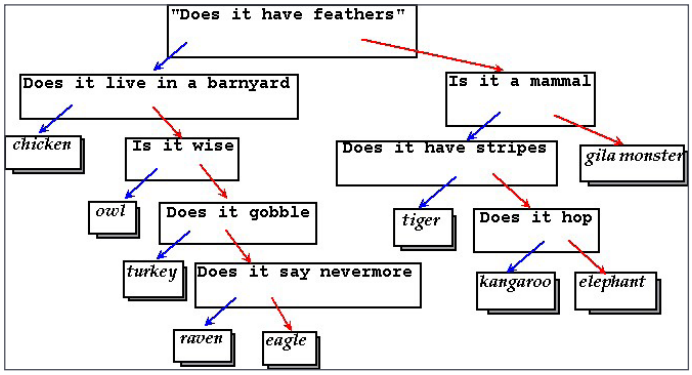
- Asking broad questions and narrowing possible answers.
- Understand how one observation may influence another component.

Preparation:

- Sit all students together.



https://en.islcollective.com/english-esl-worksheets/vocabulary/storytelling/20-questions-game/61896



https://www2.cs.duke.edu/courses/compsci201/fall13/wordpress/index.html%3Fp=383.html

Game Rules:

1. Teacher should think of a person, place, or thing for the students to guess.
2. Students take turns asking YES or NO questions only.
3. Teachers should draw a flowchart, as seen above, on the whiteboard so that students can easily visualise the process.
4. Encourage students to ask broader questions, like “Is it a person?” rather than “Is it Abraham Lincoln?” or “Can it fly?” rather than “Is it a bird?”
5. After 20 questions have been asked, the final student must guess your person, place, or thing.
6. Allow students to take the place of the teacher and think of a person, place or thing to be guessed.
7. Explain to the students how this applies to the way programmers troubleshoot a problem. Starting with broad questions that confirm or eliminate basic truths, players get more specific as they go forward. Get too specific too early and you won’t find the answer.

Sharing: 1 thing each student has learned (15 minutes)

Learning Objectives:

- To facilitate reflection on what has been learned

Directions:

1. Sit all students in a circle.
2. Go around the circle and ask students to share one thing they have learned today.

Link answers back to programming. E.g student:”I learned that you need to ask big questions first”, teacher:”Right! That’s exactly what programmers do when solving a problem. They find the main, big area where there is a problem before they look at it in more detail.”

ACTIVITY 2: CATEGORY GAME

Schedule

HOUR	AGENDA	DURATION
1	Category Game	35 mins.
	Link to problem solving	5 mins.
	Wind-down game - Human Knot	15 mins.

Category Game (35 minutes)

Learning Objectives:

- Computational Thinking: Categorizing and grouping similar objects by their properties.

Required Supplies:

- Loose leaf paper for each student
- Pencils / pens

Preparation:

- You will need a list of 30-50 random objects (baseball, tree, beach, dog, lemon aid, etc).
- Have students write 3-10 objects, depending on the size of the class.
- When they're done, each student should write their list on the board.
- All students should have a separate piece of paper to write down their answers.

Game Rules:

1. Make the list of usable objects visible for all students to see (e.g. put objects in a word document in large font and project this onto the classroom screen. Alternatively objects could be printed out and fixed to the walls or tables).
2. Assign a category that at least 5 of these objects could fit into such as "animal"
3. The first student to write down 5 applicable items from the list that fit into this category wins a point.
4. Repeat this for different categories and record the points for the students on a whiteboard.

Explanation (5 minutes)

Learning Objective:

- Students should understand how the task they just carried out can help them with problem solving.

Directions:

1. Explain to the students how categorizing a problem can help it to be solved more effectively and efficiently.
2. Explain how the categorization can help you to know ways to tackle the problem. A similar problem may have been done before and this knowledge can be brought to solving this new problem.

Wind-down game: Human Knot (15 minutes)

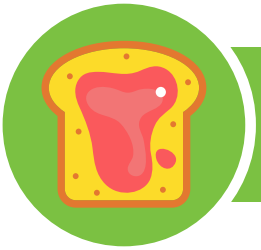
Preparation Video: https://www.youtube.com/watch?v=2OtLJMYftQo

Learning Objectives:

- To encourage all students to work together
- To teach students problem solving and communication skills

Directions:

1. Form students into a circle.
2. Each student holds the hand of two other students, creating a tangled knot of arms.
3. Students work together to untangle without letting go.



ACTIVITY 3: PROGRAMMING PEOPLE

Schedule

HOUR	AGENDA	DURATION
1	Programming People	55 mins.

Programming People (55 min)

Learning Objectives:

- Understanding computer logic by taking everyday actions and breaking them down into the simplest steps possible. Students will have to learn specificity to avoid miscommunication with the computer (teacher) following out the instructions.

Required Supplies:

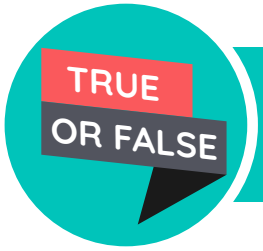
- Whiteboard or Paper/Pencils
- Food to make a PB&J sandwich

Directions:

1. Use this activity to teach the concept of how computers follow precise instructions. Computers and robots take things very literally, and can do what they are told, and nothing more.
2. Have the campers provide the teacher with instructions on how to complete a task (i.e. make a PB&J sandwich). If the campers says to move your hand, maybe let go of jar of peanut butter but still move your hand, or move the wrong hand. Similar exercises:
   a. Programming a Peanut Butter and Jelly Sandwich
   b. Programming a friend
   c. Robot Hands
3. Explain to students how computers are like this - they need specific instructions to function properly.

INTERMEDIATE

Recommended for Grades 7-9



ACTIVITY 4: BOOLEAN CONDITIONS

Schedule

HOUR	AGENDA	DURATION
1	Boolean Conditions	50 mins.
	Recap	5 mins.

Game: Green Glass Doors (50 minutes)

Learning Objectives:

- Learn about true/false (boolean) conditions.
- Use inductive reasoning to formulate a hypothesis.
- Practice debugging and troubleshooting skills by creating test cases to confirm or reject a hypothesis.

Required Supplies:

- Projector or white/blackboard w/ markers/chalk (optional)

Preparation:

1. Think of a condition that you can use to judge an object by. They should be questions that have binary yes/no answers. The classic condition for this game is to allow objects with two of the same letter next to each, hence Green Glass Doors. Here are examples of other rules:
    - Is it blue?
    - Can it fly?
    - Does it have four legs?
    - Does the word start and end with the same letter?
2. Differentiation - If working with advanced learners, we can introduce more complex conditions that use AND logic, such as:
    - Can it fly AND is it a machine?
    - Does it have four legs AND is a mammal?
3. (Optional) Create a table on the board or projector with 2 columns labeled CAN PASS and CAN'T PASS

Game Rules:

1. Introduce the idea of the imaginary "Green Glass Door". The door only allows certain things to pass through. Those things all share something in common.
2. Start the game by providing learners with an example of something that would pass through the door. Optionally, write that example in the CAN PASS column.
3. Learners then try to discover the condition by asking whether certain objects can pass (e.g. "can water pass?"). Optionally, keep track of the items using the table.
    a. If learners are having trouble, guide them by asking what things in the CAN PASS list have in common.
    b. Teachable moment - When a learner guesses the condition incorrectly, point them to an example in the CAN'T PASS column eliminates their guess. For example, if "bird" is in the CAN'T PASS column and a learner guesses the condition is "only flying things can pass", point out "bird" didn't pass even though it flies.
    c. Objects that might but won't always/usually fulfill a condition do NOT pass (e.g. a car can be blue but not all cars are blue, whereas blueberries are always blue).
4. Once the condition is solved, you may start again with a different rule, or challenge students to create their own. Allow a student to think of a condition and have his/her peers guess the condition through repeating the game above.

Example
Condition: "Can it fly?

CAN PASS	CAN'T PASS
Eagle Helicopter Paper Airplane	Paper Rock Chicken

Recap: (5 minutes)

Directions:

1. Sit students in a circle and ask them what they think Boolean conditions are from the game they just played.

Explain to the students that Boolean conditions are conditions with a yes/no answer only.

ACTIVITY 5: PASSWORDS

Schedule

HOUR	AGENDA	DURATION
1	Password guessing game	35 mins.
	Phishing video and discussion	20 mins

Password guessing game (35 minutes)

Learning Objectives:

- Utilize algorithmic/computational thinking to maximize efficiency.
- Apply probability and statistics to a real world problem.
- Demonstrate the importance of password security.

Required Supplies:

- (Optional) Projector or white/blackboard w/ markers or chalk

Preparation:

- Prepare the projector or board so that students can see.

Game Rules:

1. Start by thinking of a secret password with only one digit from 0-9. Ask the learners to take turns guessing the password. Record guesses in a single column.
    a. Repeat guesses are not recorded.
2. Once the class has discovered the password, record the number of guesses it took to arrive at the correct answer.
3. Now think of a password with 2 numerical digits. This time, when a learner's guess contains a correct digit and position, put a checkmark next to that guess. Don't reveil which digit they guessed correctly.
    a. For example, if the password is 15 and the guess is 12, they would get one check-mark, but 21 would get no checkmarks.
4. Record the number of guesses it takes for learners to find the second password.
5. Repeat this process for 3 and 4 digit passwords. Make sure to put a checkmark for each digit the learners get correct in a guess.

6. You can have a learner come up and do their own password for the class to guess. Record the number of guesses each time.
    a. Encourage algorithmic thinking by having them find the “checkmarked” digits through an ordered process of elimination.
7. Point out how the longer a password was, the more guesses it took for learners to find it.

Example
Password: 1234
Guesses:
1376 - x x x
1374 - x x -
1254 - - x -
1234 - - - -

Watch: Video on Phishing and discuss (20 minutes)

Directions:

1. Write the following questions on the whiteboard and ask students to think about these questions as they watch the video:
    - What is ‘phishing’?
    - What does this mean for us when we create passwords?
    - Should we use the same password for everything?
    - Is that safe?
    - What should we do if we experience ‘phishing’?
2. Play the video: https://www.youtube.com/watch?v=9TRR6lHviQc
3. Ask the students to form a circle and have students discuss the video they just watched.
4. Explain the answers to the question as you go.

ACTIVITY 6: HTML CLASSES

Schedule

HOUR	AGENDA	DURATION
1	HTML and CSS classes	20 mins.
	Matisse cutouts	30 mins.
	Divs in websites	5 mins.

Understanding HTML/CSS Classes (20 minutes)

Learning Objectives:

- Students should understand what 'classes' are in HTML and CSS from the classification of their own attributes.

Directions:

- Have students raise their hands to show if certain criteria apply to them (e.g. do you have a dog? Do you play baseball? Do you like carrots?). Explain to the students how this shows that we can group people by categories.
- Next, have them do separate actions to represent multiple classes. For instance, raise your right hand if you've traveled out of the country, stand up if you have a cat, and lift your left leg if you have blue eyes. This shows how an individual can be represented by multiple attributes.
- Finally, link this concept to how headings and boxes on a computer can be grouped and styled.

Matisse Cutouts (30 minutes)

Required Supplies:

- Multiple sheets of coloured paper
- Scissors (at least one per 2 students)
- Glue (at least one per 2 students)

Directions:

- Ask the students to use the colored paper provided to cut out various shapes. (e.g. squares, rectangles, triangles, circles all of different sizes)
- Ask the students to stack and glue the shapes together.

- Explain that a ‘div’ is the term used in HTML for a box.
- Explain to the students how stacking and gluing the shapes together mimics the structure and positioning of nested s in HTML. (A ‘nested div’ is a ‘div’ inside another ‘div’. Almost like a rectangle inside another rectangle, for example.)

Divs in Websites (5 minutes)

Directions:

1. Explain to students how you can see one shape inside another on a webpage. In the code behind it all this would be one div inside another, i.e. a ‘nested div’.

ADVANCED

Recommended for Grades 10-12

ACTIVITY 7: BUBBLE-SORT ALGORITHM

Schedule

HOUR	AGENDA	DURATION
1	Sorting Game	20 mins.
	Introduction to bubble-sort and algorithms	20 mins
	Sorting Game with the application of bubble-sort	15 mins.

Sorting game (20 minutes)

Learning Objectives:

- Students will be able to understand what an algorithm is.
- Students will be able to understand how to apply a bubble sort algorithm.

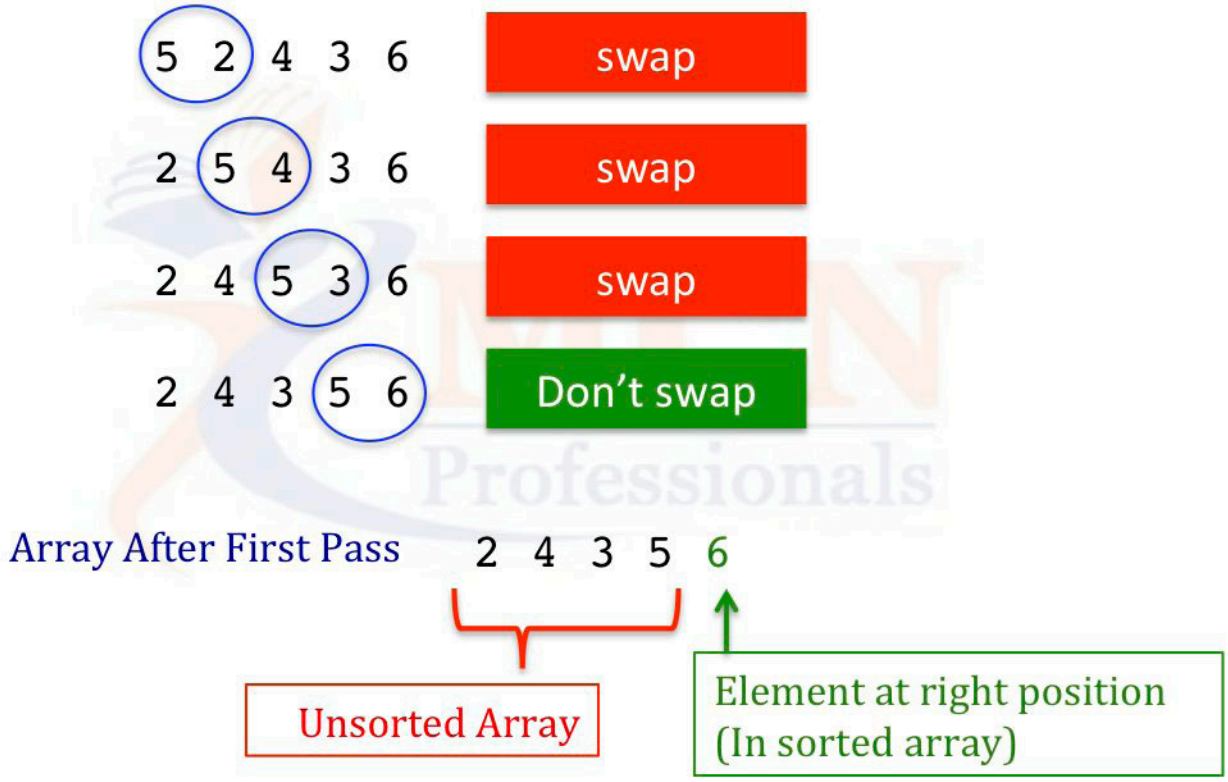
Required Supplies:

- Stopwatch

Game Rules:

1. Choose a criterion for students to be sorted by. This can be anything from height, first/middle/last name, birthdate, to shoe size.
2. Have students line up in order on their own. Time how long it takes from them to do so.
3. Have students brainstorm ways in which they can shorten the amount of time it takes for them to sort themselves. Write these ideas on the board.
4. Try again with a different criteria.

Explain: Bubble sorting (20 minutes)



https://www.ritambhara.in/optimized-bubble-sort-algorithm/

Directions:

1. Write down " 5, 10, 32, 16, 21" on the board. Explain that this list will be sorted into descending order (i.e. the numbers should be getting smaller and the largest number should be on the far left)
2. Compare the first two numbers. "Which is bigger? - 10 is bigger so put it on the left." Rewrite the list underneath as: "10, 5, 32, 16, 21".
3. Compare 5 and 32. "Which is bigger? - 32 is bigger so put it on the left." Rewrite the list underneath as "10, 32, 5, 16, 21".
4. Compare 5 and 16. "Which is bigger? - 16 is bigger so put it on the left." Rewrite the list underneath as: "10, 32, 16, 5, 21".
5. Compare 5 and 21. "Which is bigger? - 21 is bigger so put it on the left." Rewrite the list underneath as: "10, 32, 16, 21, 5". Explain how the smallest number, 5, is on the very right and is the only number completely sorted so far.
6. Perform the next pass-through by repeating this process. After the 2nd pass-through 10 should be in the right position.
7. Continue until the list is completely sorted (i.e. "32, 21, 16, 10, 5").
8. Finally, explain how what we just did is called an "algorithm" and it is a set of steps that we follow to solve a problem.
9. Explain to students how this is what computers do to sort a list of BIG numbers. This set of steps would make the sorting process much easier and quicker. However, today we have just learned the process as applied to a short list of numbers so that you understand how it is done.

Task: Sorting by height using bubble sort method (15 minutes)

Learning Objectives:

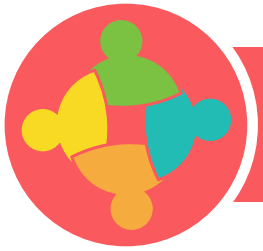
- Students will be able to visually apply a bubble sort algorithm.

Preparation:

- (Optional) An overview of common sorting algorithms: https://brilliant.org/wiki/sorting-algorithms/

Directions:

1. Choose 5 students from the class and make them line-up randomly.
2. Perform the bubble-sort method to sort the students in descending order of height. Ask the other students not in the chosen 5 to tell you whether two students should switch or not at each decision.
3. Keep going through the line until you have gone through it 4 times (i.e. 4 pass-throughs), asking each pair of students if a switch needs to be made. At the end of this the 5 students should be in the correct height order.
4. If there is extra time then choose another 5 students and try sorting in ascending order. Note that the prompt question will be “Who is shorter?” - the shorter student should move to the left.



ACTIVITY 8: TEAM CODING

Schedule

HOUR	AGENDA	DURATION
1	Team Coding	55 mins.

Team Coding (55 minutes)

Learning Objectives:

- Reinforce coding syntax through a competitive and fun game.
- Trusting team members to acquire the correct syntax.
- Quickly and accurately collaborating to create working code.
- Troubleshooting errors in their code.

Required Supplies:

- Large cardboard cards
- Markers
- Tape
- Magnets
- An area to place the cards

Preparation:

- Write out multiple of each code card you will need for the prompts you will be asking.
  - Things like quotation marks need two separate cards.
- You will need cards such as “var”, “x”, “=”, and “number”.
- Make a set for each team.
- Have a large enough area for the two teams to place their answers down.

Game Rules:

1. Two teams of students each start with their stack of code cards.
2. Once a prompt is given, the two teams must race against the other team to find all the coding elements they need, and arrange them in the most convenient way for the teacher to see.
   a. (i.e. holding each card, arranging them on the ground, hanging them on the wall, taping them on the wall, make them magnetic to go on the whiteboard)
2. The first team to finish wins a point for that round.
3. Reset cards and await next prompt.
   a. Or even build the next prompt off of what they already have.
4. Prompts could be “Create a button” and the teacher explains that we are looking for the opening and closing button tags. Another prompt could be “Create a paragraph”. Students are then looking for the opening and closing paragraph tags.
5. Keep score of the team points on the whiteboard.
6. Try again with a different criteria.

ACTIVITY 9: DESIGN MULTI-PAGE WEBSITE

Schedule

HOUR	AGENDA	DURATION
1	Research Multi-page websites online	20 mins.
	Design your own Multi-page Website	35 mins.

Research (20 minutes)

Learning Objectives:

- Students will learn about the function of different pages on a website.

Directions:

- Students should look at various websites online and learn the different structures possible.
- Tell students to look at restaurant websites for small and local businesses.
- Tell students to look at local community service websites like for animal shelters or volunteer organizations.

Design (35 minutes)

Learning Objectives:

- Students should be able to design the elements needed for a Multi-page website using a planning guide.

Directions:

- Give each student a blank sheet of paper and a copy of the Multi-page Website planning document (below).
- Students should draw out a possible layout for a Multipage Website they would like to create and write down what would go where. Here is a link to an example website that has 4 pages: https://app.bsd.education/share/o/64ja2d4r/
- Follow along with the Multi-page Website planning document to plan out all of the necessary elements needed to make a Multi-page Website.

Multi-Page Website Planning Document
(https://bsd.education/wp-content/uploads/2022/03/Multi-Page-Website-Planning-Document.pdf)

NEXT STEPS

Now that your class has learned more about computational thinking, you are ready to start your first coding project with BSD-START -- a free 4-hour course that ends with students coding their own multi-page websites.

Teachers can create their accounts at bsd.education/bsd-start and add their students once they sign in.

bsd.education/bsd-start

Learn more about how BSD Education can help integrate computational thinking, coding and other digital skills into your classroom at

www.bsd.education

HTML Cheat Sheet

Hyper Text Markup Language (HTML) is a coding language used to build websites. Specifically, HTML's job is to label and organize content such as headings, paragraphs, lists and images, so that the web browser (e.g. Chrome, Firefox, etc.) knows how the page should look.

Key Vocabulary

Element

Each individual piece of content on a web page is an element (text, images, links, etc).

```
<h1>This is my page heading</h1>
```

A text element example.

Tag

Elements are made up of tags. Typically, tags come in pairs: a start tag and an end tag, noted by a forward slash (/).

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<p>Paragraph</p>
<img src="path/to/image.jpg" />
```

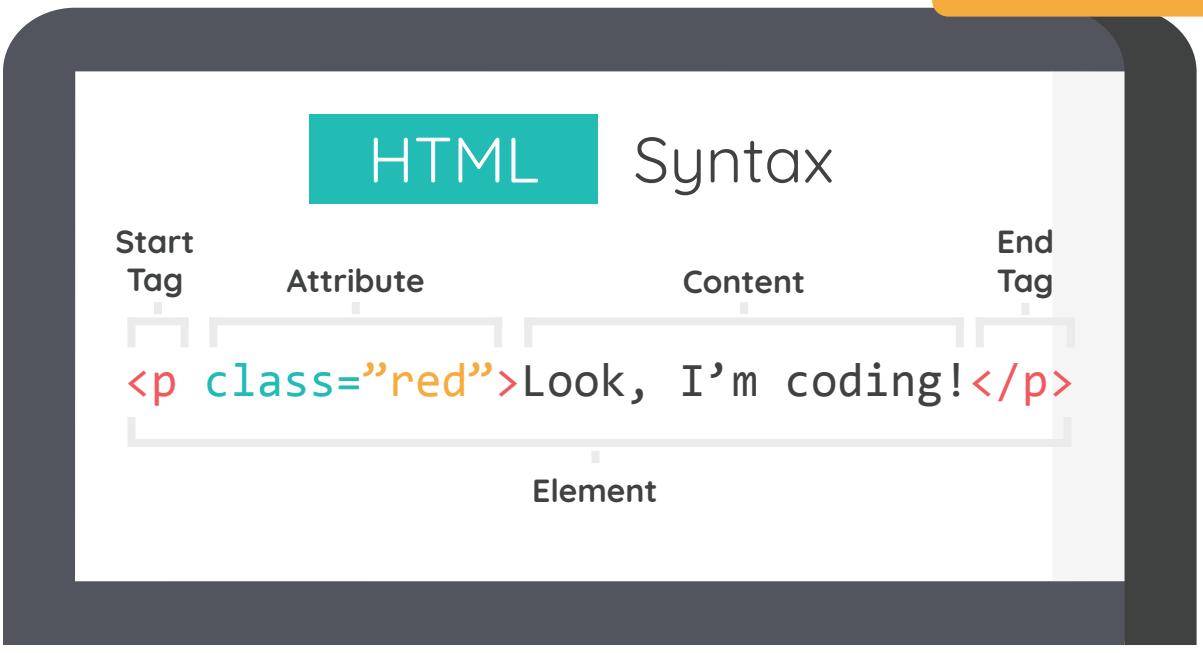
Common Tags

Attribute

Attributes provide additional information about elements. Images and links require attributes to work properly.

```
<a href="www.google.com">Look it up!</a>
```

Example of attribute linking to website.



HTML Cheat Sheet

Common HTML Elements

Headings

- Headings are the only HTML elements that use numbers.
  <h1> should be used only once per page.

```
<h1>My large heading</h1>
<h6>My small heading</h6>
```

Headings come in 6 different sizes.

Paragraph

Try using a
 tag to add line breaks within paragraphs.

```
<p>The fox jumps over the lazy dog.</p>
```

Don't forget your end tag (</p>) when adding a paragraph.

Image

The image element is a single tag.
The src (source) attribute is required.

```
<img src="my_image.jpg" />
```

Remember to include quotation marks, they are required for attribute values.

(Link) Anchor

Anchor tags can be used to link text or images.
The href (hypertext reference) attribute is required.

```
<a href="www.example.com">Click me</a>
```

The end tag signals when to stop the link.

(Unordered) List

Any number of <li> (list items) can be nested within a list.
Try <ol> for an

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

Keep in mind that nested elements are indented.

Div

Div tags are useful for grouping and organizing page content.

```
<div>Web page content</div>
```

The div tag is known as a Division tag.

CSS Cheat Sheet

Cascading Style Sheets (CSS) is a coding language that pairs with HTML. It works by defining a series of rules for how the HTML should look (colors, spacing, etc).
CSS is helpful for establishing the layout and personality of a website.

Key Vocabulary

Rule

Rules define how HTML content will appear. Rules are made up of two parts: (1) selectors and (2) properties.

Note the use of brackets for a rule.

```
h1 {
  color: blue;
}
```

Selector

The selector specifies which HTML element should be styled (e.g. img or ul). Note that selectors do not include angle brackets (< >).

This selector is styling a heading.

```
h1 {
  color: blue;
}
```

Property

Properties describe how that element should look and behave (e.g. color or font size).

This property defines the color.

```
h1 {
  color: blue;
}
```

Value

Each property needs a value. Different properties require specific types of values, from numbers to font names.

```
h1 {
  color: blue;
}
```

Note the use of a semi-colon after the specified value.

CSS Syntax

Rule

Selector

```
h1 {
  color: blue;
}
```

Property

Value

CSS Cheat Sheet

Common CSS Properties

color

Defines the color of text, borders and list bullets

Remember the selector needs to be specified before the color.

```
color: purple;
```

background-color

Defines background c
Try background-image

background-image: href("image.jpg");

```
background-color: plum;
```

font-family

Use quotes for specific fonts

Don't forget quotes around the font.

```
font-family: "Impact";
```

font-size

A popular unit for font-size is "em" (e.g. font-size: 1em;)

The "px" after the number stands for pixel.

```
font-size: 24px;
```

border

Note the three values for border: (1) width,

Width Style Color

```
border: 1px solid green;
```

margin

The space between HTML elements

```
margin: 10px;
```

These three properties together are known as the "box model."

padding

The space between HTML content and the border

```
padding: 10px;
```

JavaScript Cheat Sheet

JavaScript (JS) can be combined with HTML and CSS to bring websites to life! JavaScript is a versatile programming language that can be used for animation, dynamic apps, interactive games and more.

Key Vocabulary

Variable

A variable is like a container, used to store data.

Variables must be identified with unique names.

```
var food = “cake”;
```

Function

Functions define repeat actions that can be performed on demand.

This function returns the sum of a and b.

```
function sum(a, b) {
  return a + b;
}
```

Conditional

Conditionals (if... then...) are used to determine which action to perform depending on some condition.

```
if( thisHurts == true ) {
  alert( “Ouch!” );
}
```

“if” executes a block of code when a specific condition is true.

(for) Loop

Loops allow the same action to be executed many times.

```
for ( var i=0; i<3; i++ ) {
  alert( “Hip, hip, hooray!” );
}
```

The “for loop” is shown here.

JavaScript Syntax

Function

```
function myFunction() {
  var dogName = “Ruffles”;
}
```

Variable

JavaScript Cheat Sheet

Common JS Data Types

Number

Standard math operators work in JavaScript (+, -, *, /)

```
1 or 21.1 or -6
or 0 or 42e3
```

Very large or small numbers can be shorted using scientific notation.

String

Strings must be surrounded by quotation marks

```
“cat” or “Michael Jordan”
```

Strings are used for storing and manipulating text.

Boolean

Booleans are binary, either true or false

```
true or false
```

Booleans can only be true or false.

Useful JS Methods

alert

Triggers a pop-up message

```
alert( “Hello, world.” );
```

prompt

Triggers a pop-up with a text input field

```
prompt( “What is
your name?” );
```

confirm

Triggers a pop-up with “OK” and “Cancel” options
Returns true or false

```
confirm( “Continue” );
```

Users will have to click “OK” to proceed when these three pop-ups occur.

LET'S BUILD SOMETHING
DIFFERENT TOGETHER

www.bsd.education