Lesson 1

Add-ons

Pre-Lesson Ideas:

* Use printed grid paper and have students draw and label a sketch of what they want to draw.
* Students can only use the shapes provided so make sure to remind them of their constraints.
* Then, require them to get approval of their hand drawn sketches, before they can start programming.

Post-Lesson Ideas:

Reflection Questions

1) When your code did not work, what were steps you took to fix it?

Possible Answers: Look for spelling mistakes/ Check for missing semicolons/ Ask a friend to look over your code

2) Did anything unexpected happen? What did you think would happen?

Possible Answers: My rectangle was not drawn in the place I wanted it to. I thought the x and y coordinates were describing the center of the rectangle, not the top left-hand corner.

Further Development

* Have students take their project and re-create it in advance sketch. This will help them understand the math that goes into creating shapes.
* After, you can challenge students to add even more details to their sketch in advanced. They will be able to see how much easier it is to add details, with a larger scale grid. (many young students can get intimidated by the large numbers)

Lesson 1

Set Up

The Transportation Project:

* Teach students visual programming logic by having them use the beginning grid to design a transportation device using basic JavaScript Commands.

Project Goal:

1) Generally, what should the project look like?

A simple sketch (maybe with color) of a type of transportation device. This could be a car, skateboard, animal, etc…

2) What skill(s) are being learned/ practiced?

Navigating the coordinate grid

Understanding and building geometric shapes

Basic shape commands and syntax in JavaScript

Basic programming logic

3) What concept are students gaining insight on?

Communicating with the computer to translate an idea into a product with code.

How geometrics are formed and can be manipulated on a coordinate grid.

Programming/ Math Vocabulary:

- **Syntax** - The spelling and grammar of a programming language
- **Sequence** - The order of events, commands that are executed by a computer exactly as they are written.
- **Coordinate Grid** - A two-dimensional surface by two intersecting and perpendicular number lines on which points are plotted and located by their x and y coordinates.
- **X-coordinate -** The horizontal distance to a point. (from the origin of the graph)
- **Y-coordinate -** The vertical distance to a point. (from the origin of the graph)
- **Width/Height -** Width describes the linear extent/measurement of a shape along the x-

Lesson 1

Outline

Introduction to Topic:

"Today we are going to be making our own transportation device using basic shapes. We will be drawing each shape on the **coordinate grid** (define). If we want to tell the computer how to draw a point, we have to give an **x and y coordinate** (define)."

Go through as many of the commands listed below as you want. We suggest writing them on the board, so the students have a reference. Younger students can create their device with just points.

"It is important that we write our code in the correct **sequence** (define). The computer draws each shape in the order that it is written, from top to bottom. If your code does not work at first, check your **syntax** (define) before asking for help. "

Project Breakdown:

1) Plan out your drawing (use graph paper and label each shape)

2) Program basic shapes

3) Problem solve and trouble shoot errors

4) Add color (optional)

Example Projects/ Basic Source Code:

```
point(x, y);

line(x1, y1, x2, y2); // connects two points to make a line

rect(x, y, w, h); // x and y coordinate indicate the top left hand corner of the rectangle

ellipse(x, y, w, h); // x and y coordinate indicate the center of the ellipse. The width and height represent the length from one end of the circle to the other.

triangle(x1, y1, x2, y2, x3, y3); // connects three points

quad(x1, y1, x2, y2, x3, y3, x4, y4); //connects four points

fill(r, g, b); // percent of red, green, and blue from 0-255
```

Lesson 1

Troubleshooting

Common Mistakes and Confusions:

1) Misspelling Commands

It is really easy for students to make silly mistakes, when they are working with a subject completely unfamiliar to them.

2) Forgetting a comma or semi-colon

Learning JavaScript syntax is like learning a new language; in time these mistakes will go away.

3) Shortening commands based on assumptions

4) Writing code in the wrong order, so shapes that are supposed to be on top end up behind larger shapes

Remind students that the computer draws the code written at the top of the screen first.

FAQ's:

1) Are the commands case sensitive?

Yes! All are lowercase except when you are taught otherwise.

2) How many numbers do I need to put in the commands?

It depends on what command you are using. Think about what information you need to relay to the computer to create the shape most efficiently.

Point - 2 (for x and y) / Line, Ellipse, Rectangle - 4/ Triangle - 6 / Quadrilateral - 8

3) Why do I need a semi-colon?

It is the computer's period. It is how the computer knows that you are done with that command and are moving onto the next.