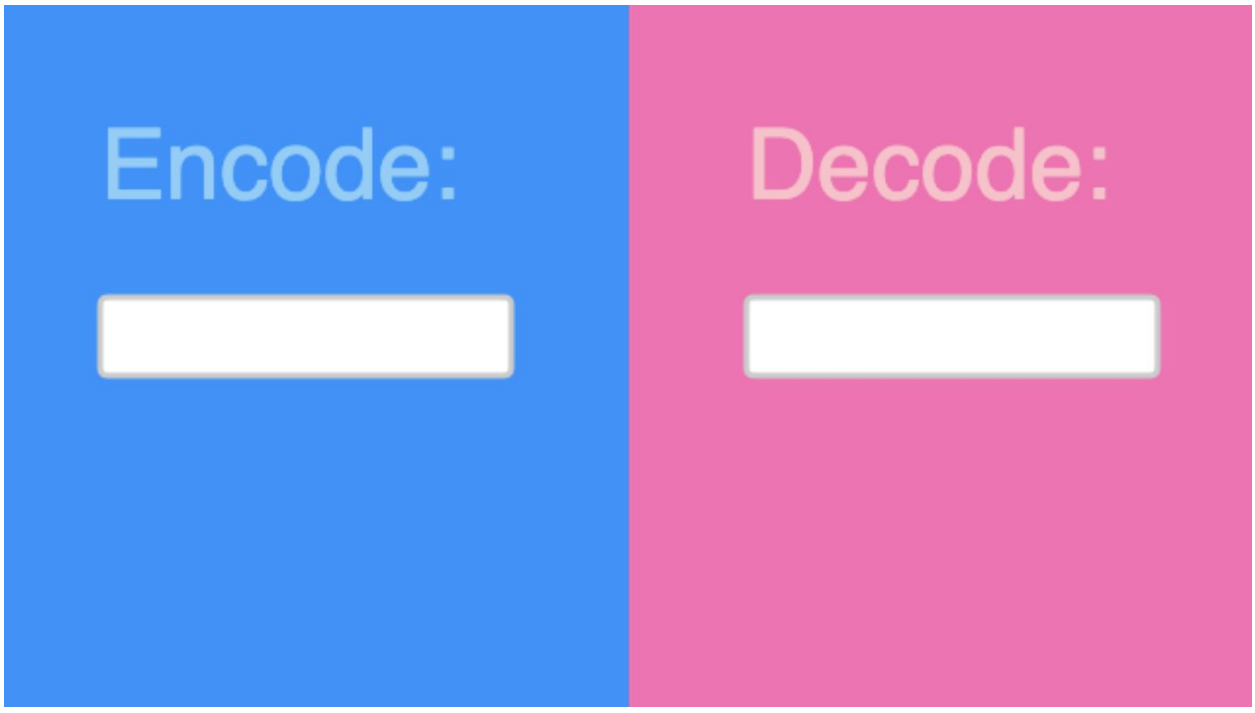
Vidcode Lesson Plans
www.vidcode.com
info@vidcode.com



Super-Secret Coded Message

Big Idea: Creating your own functions with parameters can make your code easier to reuse!

What You Are Making
End Project = a secret encoder where a friend can type in something and something cool happens. We suggest you come in with ideas about security and communication with friends.

Standards

CSTA 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.

CSTA 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.

CSTA 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

CSTA 2-AP-17 Systematically test and refine programs using a range of test cases.

CSTA 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

AP CS P Standards

EK 5.3.1I Strings and string operations, including concatenation and some form of a substring, are common in many programs.

EK 5.3.1D Procedures have names and may have parameters and return values.

EK 5.3.1E Parameterization can generalize a specific solution.

EK 4.1.1F Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.

EK 5.3.1G Parameters provide different values as input to procedures when they are called in a program.

Time

- 45-60 minutes
- 10 minutes background
- 20-35 minutes coding
- 10 minutes sharing
- 5 minutes reflection

Background (10 minutes)

Review:

- Strings
- Event Listeners
- Functions

You know that when you call certain functions, you can specify what arguments it takes by typing them in the parentheses. When you create your own functions you can decide what arguments your function will take by defining **parameters**.

If you define **parameters** for your functions, you can enter any value to the function when you call it as long as the value is the correct **data type** (string, number, array, etc.)

Creating your own functions with arguments make your code more reusable. Instead of creating a function like this:

```
function greeting(){
  myText.message = "Hello, " + myInput1.message;
}
```

You can write it like this:

```
function greeting(name){
  myText.message = "Hello, " + name;
}
```

Since you set a parameter, the second function could be used with an argument like "Emily" or `myInput1.message` or `myInput2.message`. This means the code is more reusable!

When you look up functions in the Vidcode reference, it actually lists their parameters! That's why the code doesn't work right away when you copy and paste it. You have to put values in as arguments for the code to work!

Another important thing you will do in this activity is use event listeners on shapes. You can have an event listener that only responds when you click on a specific shape. This means you can make your code do different things when a user clicks on different parts of the video. These can be helpful when taking values from two text inputs!

Code Challenge (20-35 minutes)

Students will create code that takes a word and converts it into numbers and code that takes numbers and converts them into a word. Students can use the functions they create to transmit messages in a secret code!

Sample Solution

```
// creates the two rectangles
var left_rect = rect(0, 0, movie.width*0.5, movie.height, "dodgerblue",
"clear");
var right_rect = rect(movie.width*0.5, 0, movie.width*0.5, movie.height,
"hotpink", "clear");

// creates two inputs and two labels, colors and positions each of them
var my_text1 = text("Encode:", 50, 50);
my_text1.color = "lightskyblue";
my_text1.size = 50;
```

```
var my_text2 = text("Decode:", 380, 50);
my_text2.color = "pink";
my_text2.size = 50;

var my_input1 = textInput(50, 150);
my_input1.size = 30;
my_input1.color = "steelblue";

var my_input2 = textInput(380, 150);
my_input2.size = 30;
my_input2.color = "fuchsia";

// the encoding function
function numberfy(word){
  // notice: 'changeA' uses 'upped' with a replace method, and then,
  // 'changeO' uses 'changeA', etc
  if (word != ""){
    var upped = word.toUpperCase();
    var changeA = upped.replace(/A/g, "1");
    var changeO = changeA.replace(/O/g, "2")
    var changeE = changeO.replace(/E/g, "3");
    var changeL = changeE.replace(/I/g, "4");
    var changeH = changeL.replace(/U/g, "5");
    my_input1.message = changeH;
  }
}

// the decoding function
function wordify(num){
  if (num != ""){
    var upped = num.toUpperCase();
    var change1 = upped.replace(/1/g, "A");
    var change2 = change1.replace(/2/g, "O")
    var change3 = change2.replace(/3/g, "E");
    var change4 = change3.replace(/4/g, "I");
    var change5 = change4.replace(/5/g, "U");
    my_input2.message = change5;
  }
}

// decodes the left side message with numberfy
left_rect.whenMouseClicked = function(){
  numberfy(my_input1.message);
```

```
}

// encodes the right side message with wordify
right_rect.whenMouseClicked = function(){
  wordify(my_input2.message);
}
```

Sharing (10 minutes)

Share your project with your classmates. See if you can figure out how your classmates' code works without looking at the JavaScript!

Reflection (5 minutes)

Computers use can perform hundreds of tasks in a matter of milliseconds. How could passing values through hundreds of steps in the process of encoding be useful for encryption?