JS Review

JS: Core concepts

- **Variables**: how to declare variables, assign, re-assign, and local vs. global scope.
- **Data types**: numbers, booleans, strings, arrays, and objects.
- **Functions**: how to group code into functions, pass arguments to them, and return values from them.
- **Conditionals**: how to use if/else statements and logical expressions.
- **Loops**: how to use while and for loops to repeat code.

JS: Variables and Data Types

```
var myAge = 29;

var myName = "Pamela";

var isSheCool = true;
```

...what other data types are there?

JS: Functions

```
var calculateFoodNeeded = function(numDays) {
  return numDays * 3;
};

var makeFunnyName = function(firstName, lastName) {
  return "Mister " + firstName + "Mc" + lastName + "Pants";
};
```

JS: Conditionals

```
var movieIsActionFlick = true;

var movieCost = 0;


if (movieIsActionFlick === true && movieCost < 1) {

    console.log('Okay fine Ill watch it');

}


var movieHasBradPitt = true;

var movieHasJohnnyDepp = false;


if (movieHasBradPitt === true || movieHasJohnnyDepp === true) {

    console.log('Ill DEFINITELY watch it');

}


if (movieHasBradPitt) {

    console.log('Def watch it');

} else if (movieCost === 0) {

    console.log('Free, might as well');

} else if (movieIsActionFlick) {

    console.log('Nah I dont like action flicks');

} else {

    console.log('I cant decide!');

}
```

JS: Loops

```
var countdown = 10;
while (countdown > 0) {
    console.log(countdown);
    countdown--;
}

var countdown = 10;
while (countdown > 0) {
    if (countdown > 1) {
        console.log(countdown + '...');
    } else {
        console.log(countdown + '!');
    }
    countdown--;
}

for (var i = 10; i > 0; i--) {
    console.log(i);
}
```

JS: Arrays

```
var children = ['Oliver', 'Pamela', 'Hunter'];


console.log('My dad has ' + children.length + ' children');

console.log('His first kid was ' + children[0]);


children.push('Alexis');

console.log('His fourth kid was ' + children[3]);


for (var i = 0; i < children.length; i++) {

    console.log('Kid #' + (i+ 1) + ' : ' + children[i]);

}
```

JS: Objects

```
var myCrazyCat = {

  name: "Angel",

  age: 3,

  likes: ["rubber bands", "boxes", "4am petting sessions"],

  fur: {colors: ["orange", "white"], pattern: "striped"}

};
```

JS: Many Environments

JS can be used inside many environments for many use cases:

- **Browser**: To make webpages interactive.
- **ProcessingJS**: To make drawings and animations.
- **NodeJS**: To make servers that render webpages and store data.
- **JohnnyFive**: To control robots and arduinos.
- **Photoshop**: To write scripts to automate image manipulation.

Each environment comes with its own set of relevant functionality and globals.

JS in ProcessingJS

In this environment, there are many functions dedicated to drawing and animation:

- **Shapes**: like rect(), ellipse(), and line()
- **Colors**: like fill(), stroke(), and background()
- **Text**: like text() and textSize()
- **Events**: like draw() and mousePressed()
- **Math**: like random() and dist()

JS in the Browser

In this environment, the functions are all for making web pages interactive, like:

- `document.getElementById("main")`
- `document.body.innerHTML += "<img src='cat.gif'>";`
- `window.setInterval(moveImage, 1000);`
- `window.addEventListener("scroll", loadMorePics);`