**Javascript Basics**

**Removing Decimal Places**

When dividing numbers we often find ourselves with some decimal. However, many times we don't want a decimal on our resulting number.  JavaScript gives us quite a few ways to remove the decimal and get back to a whole integer.

**Math.round(num)**

This rounds a number like we are used to.  If the decimal is .5 or up it will round up to the next number, where if the number is .49 or below it will round down:

var num1 = Math.round(2.5) //num1 is 3

var num2 = Math.round(2.4) //num2 is 2

var num3 = Math.round(-2.9) //num3 is -3

var num4 = Math.round(-2.1) //num4 is -2

**Math.floor(num)**

This always rounds a number down:

var num1 = Math.floor(2.5) //num1 is 2

var num2 = Math.floor(2.4) //num2 is 2

var num3 = Math.floor(-2.9) //num3 is -3

var num4 = Math.floor(-2.1) //num4 is -3

**Math.ceil(num)**

This always rounds a number up:

var num1 = Math.ceil(2.5) //num1 is 3

var num2 = Math.ceil(2.4) //num2 is 3

var num3 = Math.ceil(-2.9) //num3 is -2

var num4 = Math.ceil(-2.1) //num4 is -2

**Math.trunc(num)**

This always truncates the number, chopping off the decimal regardless of what the number is:

var num1 = Math.trunc(2.5) //num1 is 2

var num2 = Math.trunc(2.4) //num2 is 2

var num3 = Math.trunc(-2.9) //num3 is -2

var num4 = Math.trunc(-2.1) //num4 is -2

**More on Objects**

When creating an Object, we might not create it correctly the first time. Let's make another Object that represents a Dojo:

var dojo = {}; // creates an empty object

dojo = {

name: 'Coding Dojo', // property can store a string

number\_of\_students: 25, // property can store a number

instructors: ['Andrew', 'Michael', 'Jay'], // property can store arrays

location: { // property can even store another object!

city: 'San Jose',

state: 'CA',

zipcode: 95112

}

} // access object properties with dot (.) notation

console.log(dojo.name, dojo.number\_of\_students, dojo.instructors, dojo.location);

console.log(dojo["name"]); // or bracket [] notation (note: specify key in quotes)

After creating a new object we might need to add a new key with a new value. Not only that we might need to update the information our Object holds. Let's add a new key to our Dojo Object called snacks where we have an Array of snacks we have at our Dojo.

dojo.snacks = ['Fig Bars', 'Bagels', 'Popcorn', 'Apples'];

Now that we have added a new key with a value to our object, let's update some of the previous keys with new values. Each time we use the same key, we reassign the value for that key.

dojo.number\_of\_students = 40; // we can reassign any of the properties

dojo.location.city = "Bellevue";

dojo.location.state = "Washington";

dojo.location.zipcode = "unknown";

If we were to console.log our object this is what we would see:

{

name: 'Coding Dojo',

number\_of\_students: 40,

instructors: ['Andrew', 'Michael', 'Jay'],

location: {

city: 'Bellevue',

state: 'Washington',

zipcode: 'unknown'

},

snacks: ['Fig Bars', 'Bagels', 'Popcorn', 'Apples']

}

**Arrays of Objects**

It is very common to see arrays filled with objects, where the objects all have the keys but with different values. Let's look at Mike's favorite collection of donuts.

var glazedDonuts = [

{

frosting: 'glazed',

style: 'cake',

type: 'old fashioned',

age: '45',

time: 'minutes'

},

{

frosting: 'glazed',

style: 'yeast raised',

type: 'regular',

age: '5',

time: 'minutes'

},

{

frosting: 'glazed',

style: 'yeast raised',

type: 'jelly filled',

age: '1',

time: 'seconds'

}

];

You could then go to the donut owner and ask something like: Can I buy the 1st donut on the rack if it has been out of the oven for fewer than 25 minutes? The code conversation for that would be:

var purchase;

//You

if((glazedDonuts[0].age < 25 && glazedDonuts[0].time == 'minutes') || glazedDonuts[0].time == 'seconds'){

//shop owner

purchase = glazedDonuts[0];

}

else {

console.log('sorry it has been out a bit longer');

}

As you can note, we are accessing an Array of glazedDonuts where each index represents an object. That object we accessed has keys we can print off and take a look at the details of.

Let's say Mike decided to break open the bank and purchase as many donuts as he can. Let's loop through all of the available donuts and see how many Mike can buy.

var numPurchase = 0;

for (var donut in glazedDonuts){

console.log(glazedDonuts[donut].type);

if((glazedDonuts[donut].age < 25 && glazedDonuts[donut].time == 'minutes') || glazedDonuts[donut].time == 'seconds'){

numPurchase++;

}

else {

console.log('not this donut...');

}

}

console.log(numPurchase);

Looks like Mike can buy 2 donuts!