“ ”

//prints string between quotes

“ “.length

//prints length of string between quotes

Confirm(“ “)

//prompts user to confirm information in quotes

Prompt(“ “)

//prompts for information based off quotes

Alert(“ “)

//alerts user with the message in the quotes

Console.log( )

//console logs information

varName();

//runs a variable (so if there is a console.log inside it will log that to the console)

String()

//lists a string as an input when said string is a var

.toUpperCase()

//changes var string to uppercase letters

.charAt(#)

//finds the character at that number in the string

> Greater than

< Less than

<= Less than or equal to

>= Greater than or equal to

== Equal to

!= **Not** equal to

//list of comparison operators

if ( condition)

{ code to run if true}

else

{

Code to run if false

}

//if/else statement

"some word".substring(x, y)

//substring, selects letters “x” through “y” where 0 is the first letter

var varName = data;

**Example**:  
a. var myName = "Leng";  
b. var myAge = 30;  
c. var isOdd = true;

//setting a variable

varName = data;

//changes a set variable

var functionName = function(parameters) {

// code code code;

};

//function syntax

var greeting = function (name) {

console.log("Great to see you," + " " + name);

};

//function example

greeting("Matt")

//calling a function, we **call** the function by just typing the function's name, and putting a parameter value inside parentheses after it

var perimeterBox =function (length, width){

return length + length + width + width;

};

//multi parameter function

var globalVar = "hello";

var foo = function() {

console.log(globalVar); // prints "hello"

}

//global variable (var applicable anywhere)

var bar = function() {

var localVar = "howdy";

}

console.log(localVar); // error

//local variable (var applicable only inside a function)

Math.random()

//random number generator (0.00 through 1.00)

Math.floor()

//rounds to a whole number

for (var i = 1; i < 11; i = i + 1) {

/\* your code here \*/;

}

//for loop syntax (great for doing the same task over and over when you know ahead of time how many times you'll have to repeat the loop)

for (var i = 5; i < 11; i = i + =12){

console.log(i);

}

//for loop example (starts at 5, goes to 11, counts by 1)

while(condition){

// Do something!

}

//while loop syntax (when you have to loop, but you don't know ahead of time how many times you'll need to loop)

loopCondition = false;

do {console.log("I'm gonna stop looping 'cause my condition is " + String(loopCondition) + "!");

} while (loopCondition);

//do/while loop (loops at least once)

var arrayName = [data, data, data];

//array syntax

var twoDimensional = [[1, 1], [1, 1]];

//two dimensional array

junkData[3]

//call an array

var cities = ["Melbourne", "Amman", "Helsinki", "NYC", "Tin"];

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

console.log("I would like to visit " + cities[i]);

}

//for loop listing everything in an array

newArray = [];

newArray.push('hello');

newArray[0]; // equals 'hello'

//push method (functions associated with objects)

switch (/\*Some expression\*/) {

case 'option1':

// Do something

break;

case 'option2':

// Do something else

break;

case 'option3':

// Do a third thing

break;

default:

// Do yet another thing

}

//switch statement

var color = prompt("What's your favorite primary color?","Type your favorite color here");

switch(color) {

case 'red':

console.log("Red's a good color!");

break;

case 'blue':

console.log("That's my favorite color, too!");

break;

//Add your case here!

case 'yellow':

console.log("Yellow is the color of pee.");

break;

default:

console.log("I don't think that's a primary color!");

}

//switch statement example

**and** (&&), **or** (||), and **not** (!)

//logical operators

true && true; // => **true**

true && false; // => *false*

false && true; // => *false*

false && false; // => *false*

//and logical operator rules

true || true; // => *true*

true || false; // => *true*

false || true; // => *true*

false || false; // => **false**

//or logical operator rules

!true; // => false

!false; // => true

//not logical operator rules

for (var i = start; i < end; i++) {

console.log(family[i].name);}

//for loop syntax where console logs the names in the family array

while(condition){

// Do something!

}

//while loop syntax

function Person(name,age) {

this.name = name;

this.age = age;

}

//custom constructor

var family = new Array();

family[0] = new Person("alice", 40);

family[1] = new Person("bob", 42);

family[2] = new Person("michelle", 8);

// add the last family member, "timmy", who is 6 years old

family[3] = new Person("timmy", 6);

//array

var Name = { };

//literal notation object syntax

var spencer = {

age: 22,

country: "United States"

};

//literal notation object example

Var Name= newObject();

Name.newName=data

//constructor notation object syntax

var spencer2 = new Object();

spencer2.age = 22;

spencer2.country = "United States";

//constructor notation object example

ObjectName.PropertyName

//dot notation

ObjectName["PropertyName"]

//Bracket notation

var someObject = {someProperty: someValue};

console.log( typeof someObject );

//typeof tells you what someObject is

console.log( myObj.hasOwnProperty('name') );

//hasOwnProperty prints out if the object has the property ‘name’

var suitcase = {

shirt: "Hawaiian"

};

if (suitcase.shorts){console.log("shorts")}

else {suitcase.shorts = "plad"}

console.log(suitcase.hasOwnProperty("shorts"));

//if checks to see if there is a “shorts” property, which there isn’t, so else adds one and then hasOwnProperty confirms there is a “shorts” property

for(var property in dog) {

console.log(property);

}

//prints out all properties in an object (“property” can be any word you like)

var prototypeType = typeof Object.prototype;

console.log(prototypeType);

var hasOwn = Object.prototype.hasOwnProperty("hasOwnProperty");

console.log(hasOwn);

//checks to see what type prototypeType is and then sees if it has it’s own property

dog.species = dog["species"] = "bulldog";

And if we say:

var x = "species";

then

dog[x] = "bulldog";

//by assigning the property name to a variable, we can then use the variable name in bracket notation to get the property's value

for (var x in nyc) {console.log(nyc[x])}

//example of how to find all the values for the properties in object “nyc”

for(var x in obj) {

executeSomething();

}

//for-in loop

var languages = {

english: "Hello!",

french: "Bonjour!",

notALanguage: 4,

spanish: "Hola!"

};

for(var x in languages) {

if(typeof languages[x] === "string"){

console.log(languages[x]);

}

}

//prints only strings

function Circle(radius){

this.radius = radius;

}

//Circle is the class of this object (also a prototype?)

snoopy.bark = function(){

console.log("Woof")};

//adding the bark method to the var of snoopy in the class of dog

function Dog (breed) {

this.breed = breed;

};

var buddy = new Dog("golden Retriever");

Dog.prototype.bark = function() {

console.log("Woof");

};

buddy.bark();

var snoopy = new Dog("Beagle");

snoopy.bark();

// the prototype lets both dogs bark without having to add the method

function Person(name,age) {

this.name = name;

this.age = age;

}

var printPersonName = function (p) {

console.log(p.name);

};

var me = new Person("Matt Greenlaw", 26);

printPersonName(me);

// a complete object with the class of Person and the function of printPersonName which will print the name of anyone in that calss (printPersonName can be anything)

function Animal(name, numLegs){

this.name = name;

this.numLegs = numLegs;

}

Animal.prototype.sayName = function() {

console.log("Hi my name is " + this.name);

};

var penguin = new Animal("Captain Cook", 2);

penguin.sayName();

//prints the name property for any Animal after Hi my name is

function Animal(name, numLegs) {

this.name = name;

this.numLegs = numLegs;

}

Animal.prototype.sayName = function() {

console.log("Hi my name is " + this.name);

};

function Penguin(name){

this.name = name

this.numLegs=2

}

Penguin.prototype = new Animal();

var penguin = new Penguin ("sam");

penguin.sayName();

// Penguin inherits properties and methods from Animal by using prototype=. In this case, the sayName method from Animal is used by Penguin without having to write out Penguin’s own sayName method

function Penguin(name) {

this.name = name;

this.numLegs = 2;

}

function Emperor (name){

this.name = name;

}

Emperor.prototype = new Penguin();

var newemp = new Emperor ("frank");

console.log(tim.numLegs);

//frank the emperor is prototyped into a penguin with 2 legs

console.log( myEmperor.hasOwnProperty("name") );

//checks to see if myEmperor has a name property

function Person(first,last,age) {

this.firstName = first;

this.lastName = last;

this.age = age;

}

var john = new Person('John','Smith',30);

var myFirst = john.firstName;

var myLast = john.lastName;

var myAge = john.age;

//Person shares it’s properties with john when we call them

function Person(first,last,age) {

this.firstname = first;

this.lastname = last;

this.age = age;

var bankBalance = 7500;

}

var john = new Person ("tim", "john", 26);

var myFirst = john.firstName;

var myLast = john.lastName;

var myAge = john.age;

var mybankBalance = john.bankBalance;

console.log(john.bankBalance);

//Person keeps it’s bankBalance private

function Person(first,last,age) {

this.firstname = first;

this.lastname = last;

this.age = age;

var bankBalance = 7500;

this.getBalance = function() {

return bankBalance;

};

}

var john = new Person('John','Smith',30);

console.log(john.bankBalance);

var myBalance = john.getBalance();

console.log(myBalance);

//getBalance finds the private bankBalance (finding a private method through a public method

function Person(first,last,age) {

this.firstname = first;

this.lastname = last;

this.age = age;

var bankBalance = 7500;

var returnBalance = function() {

return bankBalance;

};

this.askTeller = function (){return returnBalance};

}

var john = new Person('John','Smith',30);

console.log(john.returnBalance);

var myBalanceMethod = john.askTeller();

var myBalance = myBalanceMethod();

console.log(myBalance);

//The this.askTeller method uses the returnBalance method

var myVariable = otherVariable.function(argument);

//passing an argument

function Person(first,last,age) {

this.firstname = first;

this.lastname = last;

this.age = age;

var bankBalance = 7500;

this.askTeller = function(pass) {

if (pass == 1234) return bankBalance;

else return "Wrong password.";

};

}

var john = new Person('John','Smith',30);

var myBalance= john.askTeller(1234);

//Asking for a password

a += b

//add b to a

var cashRegister = {

total:0,

add: function(itemCost){

this.total += itemCost;

}

};

cashRegister.add(0.98);

cashRegister.add(1.23);

cashRegister.add(4.99);

cashRegister.add(0.45);

console.log('Your bill is '+cashRegister.total);

//manually adding items into the cashRegister by calling the add method

var cashRegister = {

total:0,

add: function(itemCost){

this.total += itemCost;

},

scan: function(item) {

switch (item) {

case "eggs":

this.add(0.98);

break;

case "milk":

this.add(1.23);

break;

case "magazine":

this.add(4.99);

break;

case "chocolate":

this.add(0.45);

}

return true;

}

};

cashRegister.scan("eggs");

cashRegister.scan("eggs");

cashRegister.scan("magazine");

cashRegister.scan("magazine");

cashRegister.scan("magazine");

console.log('Your bill is '+cashRegister.total);

//giving properties values and then adding up the values by using the property name

var cashRegister = {

total:0,

add: function(itemCost){

this.total += itemCost;

},

scan: function(item, quantity) {

switch (item) {

case "eggs": this.add(0.98\* quantity); break;

case "milk": this.add(1.23\* quantity); break;

case "magazine": this.add(4.99\* quantity); break;

case "chocolate": this.add(0.45\* quantity); break;

}

}

};

cashRegister.scan("eggs", 4);

cashRegister.scan("milk", 4);

cashRegister.scan("magazine", 4);

cashRegister.scan("chocolate", 4);

console.log('Your bill is '+cashRegister.total);

//listing quantities of items and adding their values

var cashRegister = {

total:0,

lastTransactionAmount: 0,

//Dont forget to add your property

add: function(itemCost) {

this.total += itemCost;

this.lastTransactionAmount = itemCost;

},

scan: function(item,quantity) {

switch (item) {

case "eggs": this.add(0.98 \* quantity); break;

case "milk": this.add(1.23 \* quantity); break;

case "magazine": this.add(4.99 \* quantity); break;

case "chocolate": this.add(0.45 \* quantity); break;

}

return false;

},

voidLastTransaction: function() {

this.total -= this.lastTransactionAmount;

},

};

cashRegister.scan('eggs',1);

cashRegister.scan('milk',1);

cashRegister.scan('magazine',1);

cashRegister.scan('chocolate',4);

cashRegister.voidLastTransaction();

cashRegister.scan('chocolate',3);

console.log('Your bill is '+cashRegister.total);

//deletes the last transaction and adds in the new scan