**JavaScript**

**Lesson 1. Introduction to JavaScript**

To discover the length of your name write your name within quotes. Then write a period (full stop) and the word **length** like this:  
"yourName".**length**

The // sign is for **comments**.

// This is a comment that the computer will ignore.

**Data types**  
a. numbers (*e.g.* 4.3, 134)

var myAge = 30;  
b. strings (*e.g.* "dogs go woof!", "JavaScript expert")

var myName = "Leng";  
c. booleans (*e.g.* false, 5 > 4)

var isOdd = true;

These boxes can be used on websites to confirm things with users.

**confirm**("I am ready to go.");

You can ask for input with a **prompt**.

prompt("What is your name?");

console.log() will take whatever is inside the parentheses and *log* it to the *console* below your code—that's why it's called console.log()!

This is commonly called **printing out**.

**List of comparison operators**:

* > Greater than
* < Less than
* <= Less than or equal to
* >= Greater than or equal to
* === Equal to
* !== **Not** equal to

**if**( "myName".length >= 7 ) {

console.log("You have a long name!");

} **else** {

console.log("You have a short name!");

}

17 **%** 5 evaluates to 2 // **modulo**

"wonderful day".**substring**(3,7); // => “derf”

**Lesson 2. Functions**

// This is what a function looks like:

var divideByThree = **function** (number) {

var val = number / 3;

console.log(val);

};

divideByThree(6); // we call the function!

var greeting = function (name) {

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

};

greeting("Vitalii");

var timesTwo = function(number) {

return number \* 2;

};

var newNumber = timesTwo(8);

console.log(newNumber);

**Lesson 3. ‘For’ loops in JavaScript**

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

console.log(counter);

} // The computer print out 1 to 10!

for (var i = 100; i > 0; i -= 5) {

console.log(i);

}

// Arrays are an awesome data structure!

var junk = ["Vi", "ta", 8, 12];

console.log(junk[0]); // => “Vi”

**Lesson 4. “While” loops in JavaScript**

var coinFace = Math.floor(Math.random() \* 2);

while(coinFace === 0){

console.log("Heads! Flipping again...");

var coinFace = Math.floor(Math.random() \* 2);

} // The loop do while condition is true

// It’s ideal when you don't know how many times you'll have to execute that loop

var loopCondition = false;

do {

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

} while (loopCondition); // The loop is doing one time and then check the condition

**Lesson 5. Control flow**

if (*/\* Some condition \*/*) {

*// Do something*

} else if (*/\* Some other condition \*/*) {

*// Do something else*

} else { *// Otherwise*

*// Do a third thing*

}

It checks to see if that thing *is not* a number

isNaN('berry'); // => true

isNaN(NaN); // => true

isNaN(undefined); // => true

isNaN(42); // => false

isNaN(“42”); // => false

var lunch = prompt("What do you want for lunch?","Type your lunch choice here");

**switch** allows you to preset a number of options (called cases), then check an expression to see if it matches any of them. If there's a match, the program will perform the action for the matching case; if there's no match, it can execute a default option.

var lunch = prompt("What do you want for lunch?","Type your lunch choice here");

switch(lunch){

case 'sandwich':

console.log("Sure thing! One sandwich, coming up.");

break;

case 'soup':

console.log("Got it! Tomato's my favorite.");

break;

case 'salad':

console.log("Sounds good! How about a caesar salad?");

break;

case 'pie':

console.log("Pie's not a meal!");

break;

default:

console.log("Huh! I'm not sure what " + lunch + " is. How does a sandwich sound?");

}

**Logical operators**

 JavaScript has three: **and** (&&), **or** (||), and **not** (!). Look Ruby, it’s the same.

**Lesson 6. Data structures**

var languages = ["HTML", "CSS", "JavaScript", "Python", "Ruby"];

console.log(languages.length); // => 5

var newArray = [[true, true, false],[1,2,3],["yes", "no", "don't know"]]

var jagged = [[123,454,45],[45],12,45];

**Objects**

var phonebookEntry = {};

phonebookEntry.name = 'Oxnard Montalvo';

phonebookEntry.number = '(555) 555-5555';

phonebookEntry.phone = function() {

console.log('Calling ' + this.name + ' at ' + this.number + '...');

};

phonebookEntry.phone();

var me = {

name: 'Vital',

age: 23

}; // Literal notation

var me = new Object(); // constructor using

me.name = 'Vital';

me.age = 23;

// We can add as many elements of any type as we, even objects.

var myObject = {

name: 'Eduardo',

type: 'Most excellent',

// Add your code here!

interests: ['programming','ingineering']

};

**Lesson 7. Objects I**

var bob = {

name: "Bob Smith",

age: 30

};

// here we save Bob's information

var name1 = bob.name;

var age1 = bob.age;

// Take a look at our next example object, a dog

var dog = {

species: "greyhound",

weight: 60,

age: 4

};

var species = dog["species"];

A **method** is just like a *function* associated with an object.

// here is bob again, with his usual properties

var bob = new Object();

bob.name = "Bob Smith";

bob.age = 30;

// this time we have added a method, setAge

bob.setAge = function (newAge){

bob.age = newAge;

// this.age = newAge;

};

// here we set bob's age to 40

bob.setAge(40);

// bob's feeling old. Use our method to set bob's age to 20

bob.setAge(20);

// here we define our method using "this", before we even introduce susan

var setAge = function (newAge) {

this.age = newAge;

};

// make susan here, and first give her an age of 25

var susan = new Object();

susan.age = 25;

susan.setAge = setAge;

// here, update Susan's age to 35 using the method

susan.setAge(35);

var square = new Object();

square.sideLength = 6;

square.calcPerimeter = function() {

return this.sideLength \* 4;

};

square.calcArea = function() {

return this.sideLength\*this.sideLength;

}; // help us define an area method here

var p = square.calcPerimeter();

var a = square.calcArea();

function Person(name,age) {

this.name = name;

this.age = age;

}

// Let's make bob and susan again, using our constructor

var bob = new Person("Bob Smith", 30);

var susan = new Person("Susan Jordan", 25);

// help us make george, whose name is "George Washington" and age is 275

var george = new Person("George Washington", 275);

function Rectangle(height, width) {

this.height = height;

this.width = width;

this.calcArea = function() {

return this.height \* this.width;

};

// put our perimeter function here!

this.calcPerimeter = function() {

return this.height \* 2 + 2 \* this.width;

};

}

var rex = new Rectangle(7,3);

var area = rex.calcArea();

var perimeter = rex.calcPerimeter();

// **Arrays of Objects**

// Our person constructor

function Person (name, age) {

this.name = name;

this.age = age;

}

// Now we can make an array of people

var family = new Array();

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

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

// loop through our new array

for (var i = 0; i <=3; i++) {

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

};

// We can make a function which takes persons as arguments

// This one computes the difference in ages between two people

var ageDifference = function(person1, person2) {

return person1.age - person2.age;

}

var alice = new Person("Alice", 30);

var billy = new Person("Billy", 25);

// get the difference in age between alice and billy using our function

var diff = ageDifference(alice, billy);

function Circle (radius) {

this.radius = radius;

this.area = function () {

return Math.PI \* this.radius \* this.radius;

};

// define a perimeter method here

this.perimeter = function () {

return Math.PI \* this.radius \* 2;

};

};

**Lesson 8. Objects II**

var james = {

job: "programmer",

married: false,

speak: function(text) {

console.log("Hello, I am feeling " + text);

}

};

james.speak("great");

james.speak("just okay");

var anObj = { job: "I'm an object!" };

var aNumber = 42;

var aString = "I'm a string!";

console.log(typeof anObj); // should print "object"

console.log(typeof aNumber); // should print "number"

console.log(typeof aString); // should print "string"

Method **hasOwnProperty** lets us know if an object has a particular property.

console.log( myObj.hasOwnProperty('name') ); // should print true

console.log( myObj.hasOwnProperty('nickname') ); // should print false

var dog = {

species: "bulldog",

age: 3,

color: brown

};

// To print out all elements, we can use a for/in loop, like this:

for(var property in dog) {

console.log(property);

}

When you make a constructor, you are in fact defining a new **class**.

function Person(name,age) {

this.name = name;

this.age = age;

}

// Let's make bob again, using our constructor

var bob = new Person("Bob Smith", 30);

var susan = new Person("Susan Jordan", 35);

So we know that a class will have certain properties and methods, but what keeps track of what a given class can or can't do? What a class has or doesn't have? That is the job of the **prototype**.

JavaScript automatically defines the prototype for class with a constructor. For example, our Dog constructor ensures that the Dogprototype has a breedproperty. Remember, the Dogprototype keeps track of whatDog has, doesn't have, can, or can't do.

function Dog (breed) {

this.breed = breed;

}

// here we make buddy and teach him how to bark

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

buddy.bark = function() {

console.log("Woof");

};

buddy.bark();

// here we make snoopy

var snoopy = new Dog("Beagle");

// we need you to teach snoopy how to bark here

snoopy.bark();

// this causes an error, because snoopy doesn't know how to bark!

function Dog (breed) {

this.breed = breed;

};

// here we make buddy and teach him how to bark

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

Dog.prototype.bark = function() {

console.log("Woof");

};

buddy.bark();

// here we make snoopy

var snoopy = new Dog("Beagle");

/// this time it works!

snoopy.bark();

// the original Animal class and sayName method

function Animal(name, numLegs) {

this.name = name;

this.numLegs = numLegs;

}

Animal.prototype.sayName = function() {

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

};

// define a Penguin class

function Penguin(name, numLegs) {

this.name = name;

this.numLegs = 2;

}

// set its prototype to be a new instance of Animal

Penguin.prototype = new Animal(); // **inheritance**

var penguin = new Penguin("Joe");

penguin.sayName(); // it works!

function Penguin(name) {

this.name = name;

this.numLegs = 2;

}

// create your Emperor class here and make it inherit from Penguin

function Emperor(name) {

this.name = name;

};

Emperor.prototype = new Penguin();

// create an "emperor" object and print the number of legs it has

var emperor = new Emperor("Ross");

console.log(emperor.numLegs); // => 2

function Person(first,last,age) {

this.firstname = first;

this.lastname = last;

this.age = age;

var bankBalance = 7500; //private variable

this.getBalance = function() {

return bankBalance;

};

}

var john = new Person("John", "Lebovsky", 50);

console.log(john.bankBalance); // => undefined

var myBalance = john.getBalance();

console.log(myBalance); // => 7500

function StudentReport() {

var grade1 = 4;

var grade2 = 2;

var grade3 = 1;

this.getGPA = function() {

return (grade1 + grade2 + grade3) / 3;

};

}

var myStudentReport = new StudentReport();

for(var x in myStudentReport) {

if(typeof myStudentReport[x] !== "function") {

console.log("Muahaha! " + myStudentReport[x]);

}

}

console.log("Your overall GPA is " + myStudentReport.getGPA());