**Ch. 9 - Javascript**

JS is interpreted language. It runs line by line.

JS is saved on your console between runs. You don’t need to repeat it on every run

Java is compiled language. It compiles the entire code first before running.

Function(“message”); <end

alert(“hello”);

**Data types:**

String: string A = “string”

Can be concatenated using a +

Numbers: 123

no keyword needed.

Boolean: boolean A = true or false

var Name = Data;

variable of name Name is equal to Data

variable name cannot begin with a number, be just a keyword or contain a space.

Only symbols that you can use are $ or \_

Camel-casing: lowerCaseThenUpperCase

**Functions - Strings:**

alert(message)

popup with message

prompt(message)

popup with message prompting a response

stringName.length

# of characters of string

First character is at position 0

stringName.slice(firstIndex, lastIndex)

returns string of firstIndex (inclusive) to lastIndex (exclusive)

toUpperCase()

changes all of the characters of a string to upper case.

toLowerCase() does the opposite.

**Functions – Numbers:**

+ - \* / basic arithmetic functions

% returns the remainder

8 % 3 = 2

Increment and Decrement

X = 5; X = X + 1; -or- X++; X = 5; X = X -1; -or- X--;

X += 2; X -= 2; X \*= 2; X /= 2;

Math.floor(number)

Truncates the number / rounds it down.

Math.round(number)

Rounds a number to closest whole integer

Math.pow(number1, number2)

Raises number1 to number2

Math.random() – generates random number from 0 (inclusive) to 1 (exclusive)

**Functions:**

Creating a function:

function functionName(arguement){

/\*function code

argument can be used here\*/

return answer;

}

Calling a function:

functionName(); //returns answer

if, if else, else (conditional) statements – control’s the flow of your code

comparators such as === or > or >= or < or <= allows use to compare two values

!== “not equal to”

=== checks if data type and value are the same

== checks only if value is the same. (number 1 is == to string 1)

Comparators can be combined with && (and) or || (or)

Arrays: var arrayName = [arrayElement1, arrayElement2, … ]

arrayName[#] – Retrieves element at position #

Indexes are counted beginning with 0

arrayName.length gives number of elements in array

arrayName.includes(item) – searches array for value item

arrayName.push(value) – adds value to end of array

arrayName.pop() – removes last element of array

while (statement) loop: loops some code until the statement is no longer true

Used while something is in a certain state.

Code inside must ensure the statement become false at some point to avoid infinite loops

while (i >= 0){

code;

increment/change code;

}

for (start statement; end statement; incrementation/change) loop: loops some code with more specifications than a while loop until the end statement is no longer true.

Used to iterate a set amount of times.

for (var i = 0; i < 5; i++){

code;

}

target.addEventLister(“eventType”, listener);

Will call listener on target when eventType is reached

**You can use anonymous or named functions in addEventListener parameters.**

If your callback function has a parameter, you can call it using an anonymous function

target.addEventListener(“eventType”, function(){

myFuntion(argument);});

addEventListener is a higher order function as it takes functions as parameters.

listener is a callback function as it gets called back once an eventType is reached.

If you specify a parameter in your callback function, the event will be returned.

**Anonymous function:**

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

drumArray[i].addEventListener("click", function (){

alert("hey");

});}

**Lambda function:**

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

drumArray[i].addEventListener("click", handleClick);

}

function handleClick() {

alert("hey");

}

**Higher order functions can take functions as inputs.**

function add(num1, num2){

return num1 + num2;

}

function multiply(num1, num2){

return num1\*num2;

}

function calculator(num1, num2, operator){

return operator(num1, num2);

}

**Playing audio in JS:**

var audioName = new Audio(“audioFile.mp3”);

audioName.play();

**Objects:**

var objectName = {

property1: value1,

property2: value2,

property3: value3

method1: function(){

method1 code;

}};

Can have number, “string” or boolean values for properties

methods are functions that are associated with objects

objectName.property1;

calls property 1 of objectName

objectName.method1();

calls method1 of objectName

**Constructor function:**

function ConstructorName (parameter1, parameter2, parameter3){

this.property1 = parameter1;

this.property2 = parameter2;

this.property3 = parameter3;

this.method1 = function(){

method1 code

};

}

Var constructorObject1 = new ConstructorName(“value1”, value2, [value3.1, value3.2, …];

This creates constructorObject1 of type constructorName with the given parameters and attached methods

Constructor function names should be capitalized.

**Case, Switch, Break:**

switch (buttonInnerHTML) {

case "l":

var crash = new Audio("sounds/crash.mp3");

crash.play();

break;

default:

console.log(buttonInnerHTML);

break;

}

Works as a sort-of if, else if, else statement.

Checks the switch parameter, if it matches to the case value, executes the case code.

Break; discontinues the remaining switch code from running.

Default section is if there are no matching cases to the switch parameter.