JAVASCRIPT

COMMENTS IN JAVASCRIPT

// this is a comment

/\* this is a comment \*/

VARIABLES

- used to store and manage data

You can increase the value inside a variable in these examples:

var \_trainWhistles = 3

trainWhistles++ ------------- increases the variables value by 1

trainWhistles-- ------------- decreases the variables value by 1

trainWhistles += 3 ------------- adds to the variables value by any amount

trainWhistles \*= 2 ------------- multiplies to the variables value by any amount

Math.random() - If we declare a variable and make it equal to Math.random(), that variable will equal a number between 0 and 1.

DATA TYPES

Number - quantities, you can do math with them

Strings - this is how javascript stores and processes flat text. These are sequences of letters, spaces and numbers in quotation markss

Boolean - is either true or false

Special Characters in Strings

\t ---------------- tab stop

\” ---------------- quotaion marks

\\ ---------------- backslash

\n ---------------- makes a new line

We can find the length of string with the .length property

“roses”.length equals to 5

We can find a character at a specific index using the .charAt() method

var sentence = “this is it”;

sentence.charAt(3) is the letter s because strings have a 0 index

Substrings

"hello". substring(0, 2);

Since text could be quite long, you can use a backslash (\) at the end of each line to make your string "wrap" to the next line:

var text = "Hey, how are you \

doing? My name is Emily."

JAVASCRIPT FUNCTIONS

alert() - sends a message to the user in a small pop-up window

confirm() - asks the users for consent to move forward with an action which is true or false

prompt() - sends a message and retrieves an entry from the user

the typeof operator is used to identify the type of value inside a variable of expression

typeof true is equal to “boolean”

var name = “wendell”

typeof(name) is equal to “string”

return - gives the programmer back the value that comes out of the function. So the function runs, and when the return keyword is used, the function will immediately stop running and return he value. The you can put the result of the function to a variable.

function functionName ( parameter01, parameter02 ) {

\* here goes your code \*

return

}

Var functionName = function () {

\* here goes your code \*

return

}

return - gives us the result of what we did in the function

- it can be used anywhere in the function to stop the function’s work

COMMON OPERATORS IN JAVASCRIPT

Addition -------------------- +

Subtraction ---------------- -

Multiplication ------------- \*

Division --------------------- /

Modulus ------------------- %

- returns the remainder after division

List of comparison operators:

> ---------------------------------------- Greater than

< ---------------------------------------- Less than

<= ---------------------------------------- Less than or equal to

>=  ---------------------------------------- Greater than or equal to

===  ---------------------------------------- Equal to

!==  ----------------------------------------- Not equal to

COMPLEX CONDITIONALS

&& - and

- returns true if both values are true

|| - or

- returns true if either value is true

WHILE-LOOP

- runs its code as long as its Boolean expression is true

while ( \*some expression is true\* ) {

\*do this code\*

}

Ex: to make a loop that prints the numbers 1-5 in ascending order

var number = 1;

while ( number <= 5) {

console.log(number);

number++

};

FOR-LOOP

- almost the same as while-loop but has a different way of producing the same looping behavior

for( \*start with this\* ; \*loop if this expression is true\* ; \*do this after each lopp\* ) {

\* in each loop, do this code!\*

};

For ( var trainNumber = 1; trainNumber <= 10; trainNumber++ ) {

console.log( “train number ” + trainNumber + “ is operational.” );

};

IF AND ELSE STATEMENTS

If ( \*some condition is true\* ) {

\*do this code!\*

} else {

\*otherwise, do this code instead!\*

}

var totalTrain = 12

var workingTrain = 8

for ( train = 1 ; train <= totalTrain ; train++ ) {

if ( train <= workingTrain ) {

console.log("train number " + train + " is working");

} else if ( train === 10 ) {

console.log("train number " + train + " is a special snowflake! damn you train number " + train + "!");

} else {

console.log("train number " + train + " is not working");

}

};

DO WHILE LOOP

var number = 1

do {

alert(“number” + number);

number++;

} while ( number <= 10 );

SWITCH STATEMENT

var name = “name01”

switch() {

case “name01” :

alert(“name01”);

break;

case “name02” :

alert(“name02”);

break;

case “name03” :

alert(“name03”);

break;

default :

alert(“error”);

};

ARRAY

- a data structure with automatically indexed positions

- arrays are zero index which means they start at 0

var arrayName = [ “data1”, “data2”, “data3”, “data4”, “data5” ];

arrayName[4] gets “data5”

we can add a cell into the last position of the array by using the

push() function

arrayName.push(“data6”); adds “data6” in the last cell

arrays can also hold number and other arrays

var arrayOne = [ 1, 2, 3, 4 ];

var arrayTwo = [ 5, “six”, “seven”, 8 ];

var arrayCombination = [ arrayOne, arrayTwo ];

we can get things inside by calling it

arrayCombination[1][2] goes inside arrayTwo then gets the third cell of arrayTwo which is 3

OBJECTS

- are combination of variables to define one object

Standalone Variables

var color = “green” ;

var height = 5 ;

var weight = 180 ;

function functName () {}

var orc = { var color = “green” , var height = 5 , var weight = 180 , functName: function() {} };

orc.height;

orc.functName();

JAVASCRIPT HTML DOCUMENT OBJECT MODEL

THREE MAIN TYPES OF NODES IN HTML

Element Node

Attribute Node

Text Node

GRABRING AN ELEMENT BY ID, CLASS or TAG

document.getElementById(“IDname”)

document.getElementsByClassName(“ClassName”)[0] - notice that the elements is plural

document.getElementsByTagName(“span”)[0] - notice that the elements is plural

var variableName = document.getElementsByClassName(“ClassName”)

variableName[0]

document.querySelector(“.className”) - only gets the first element

document.querySelector(“#IDname”)

document.querySelectorAll(“div”) - gets all the elements

you can also add or remove classes

var classChange = document.getElementsByClassName(“ClassName”)[0]

classChange.className = “newClassName” - replace the current class

classChange.className = “” - remove the class

classChange.className = classChange.className + “ newClassName” - add a class

classChange.className = classChange.className.replace( “classToBeReplaced” , “newClass”)

another way to add or remove classes

classChange.classList.add(“newClass”)

classChange.classList.remove(“classToRemove”)

classChange.parentElement - to go up an element

classChange.children[0] - to go down an element

GETTING THE ATTRIBUTES OF AN ELEMENT

var this = document.getElementById(“IDname”)

this.attributes["data-name"].value;

after you got the ID or Class you can style it

document.getElementById(“IDname”).style.color = “blue”;

document.getElementByClassName(“ClassName”)[1].style.border = “2px solid black”;

document.getElementsByTagName(“span”)[1].style .fontStyle = “italic”; - 0 index starts at 0

CSS to Javascript Properties

color - color

border - border

margin - margin

background-color - backgroundColor

font-style - fontStyle

text-decoration - textDecoration

padding-left - padding-left

border-right - border-right

CHANGING THE CONTENT OF AN HTML

document.getElementsByClass(“ClassName”)[0].innerHTML = “new text here”

Changing an Image

document.getElementById(“IDname”).src = “new image.jpg”;

making a new element

var element = document.createElement(“p”);

var body = document.getElementById(“IDname”);

body.appendChild(element);

or

body.removeChild(element);

var text = document.createTextNode(“Some Text Here ”);

element.appendChild(text);

ADDING AN EVENT

var classToAdd = document.getElementsByClassName(“ClassName”)[0]

classToAdd.addEventListener( “click” , functionName );

function functionName () {

if ( true ) {

classToAdd.classList.add(“newClass”);

} else {

classToAdd.classList.remove(“newClass”);

};

};

Mouse Events

click - single click

dblclick - double click

contextmenu - right click

mouseenter / mouseover - hover mouse

mouseleave / mouseout - mouse leaves element

mouseup - occurs when user releases mouse button over an element