**BASIC JAVASCRIPT**

* Best practice for naming things in JS is camelCase since JS is case-sensitive. lowercaseFirstWordCapitalizeFirstLetterOfEveryWordsAfterwards
* remainder operator % in JS is not the same as  "modulus" operator since it does not work with negative numbers
* Increment ++ operator; i++ same as i = i + 1
* Decrement operator --
* += operator: var = var + 5 is the same as var += 5;
* -=, \*= operator works same way
* Escape sequences for strings use a backslash \ then add code
  + \ backslash
  + n newline
  + r carriage return: return to the beginning of the current line without advancing downward
  + t tab
  + b backspace
  + f form feed: advance downward to the next "page"; commonly used as page separators, but now is also used as section separators
* concatenation operator: using + operator with strings to combine them
* equality operator == does uses “type coercion” to convert different data types during comparisons 3 == ‘3’ // true. Strict equality (===) doesn’t convert 3 === ‘3’ // false
* inequality operator (!=) ; uses type coercion; 3 != '3' // false
* strict inequality operator (!==); 3 !== '3' // true
* logical and operator (&&) returns true if and only if the operands to the left and right of it are true

if (num > 5 && num < 10) {

 return "Yes";

}

return "No";

* logical or operator (||) returns true if either of the operands is true

if (num > 10 || num < 5) {

 return "No";

}

return "Yes";

* if/else statements

if (num > 10) {

 return "Bigger than 10";

} else {

 return "10 or Less";

}

* if/else statements can be chained together for complex logic

if (*condition1*) {

*statement1*

} else if (*condition2*) {

*statement2*

} else if (*condition3*) {

*statement3*

. . .

} else {

*statementN*

}

* switch statement tests a value and can have many case statements which define various possible values. Statements are executed from the first matched case value until a break is encountered; can add the default statement which will be executed if no matching case statements are found. Think of it like the final else statement in an if/else chain.

switch(num) {

 case value1:

   statement1;

   break;

 case value2:

   statement2;

   break;

...

 case valueN:

   statementN;

   break;

}

 default:

   defaultStatement;

   break;

* If you have multiple inputs with the same output, you can represent them in a switch statement like this:

switch(val) {

 case 1:

 case 2:

 case 3:

   result = "1, 2, or 3";

   break;

 case 4:

   result = "4 alone";

}

ARRAYS

* .push() takes one or more parameters and "pushes" them onto the end of the array
* .pop() "pop" a value off of the end of an array. We can store this "popped off" value by assigning it to a variable. In other words, .pop() removes the last element from an array and returns that element.
* .shift() removes the first element instead of the last
* .unshift() adds the element at the beginning of the array

OBJECTS

* Objects are similar to **arrays**, except that instead of using indexes to access and modify their data, you access the data in objects through what are called **properties**.  Can also use numbers as properties or omit the quotes for single-word string properties. If your object has any non-string properties, JavaScript will automatically typecast them as strings.
* Var object = {

“property1”: value1 or “string 1”,

“property2”: value2 or “string 2”,

};

* two ways to access the properties of an object: dot notation (.) and bracket notation ([ ]), similar to an array.
  + Dot notation used when you know the name of the property you're trying to access ahead of time; object.propName
  + Bracket used to access a property which is stored as the value of a variable; or object property name has a space; object[“property name]
* When accessing number property such as var dogs = { 1: “test”;}
* Update or add property to object via: ourDog.name = "Happy Camper";
* Delete property via: delete ourDog.bark;
* Testing objects for property via: .hasOwnProperty() returns true or false if the property is found or not
* JavaScript Object Notation or JSON is a related data interchange format used to store data.