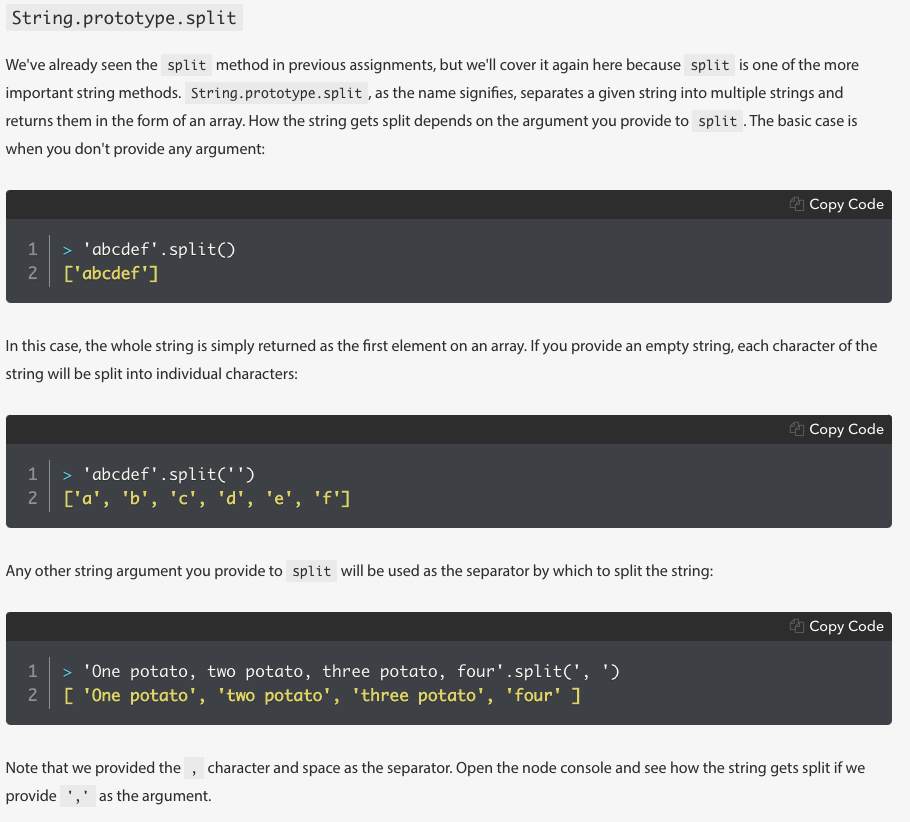
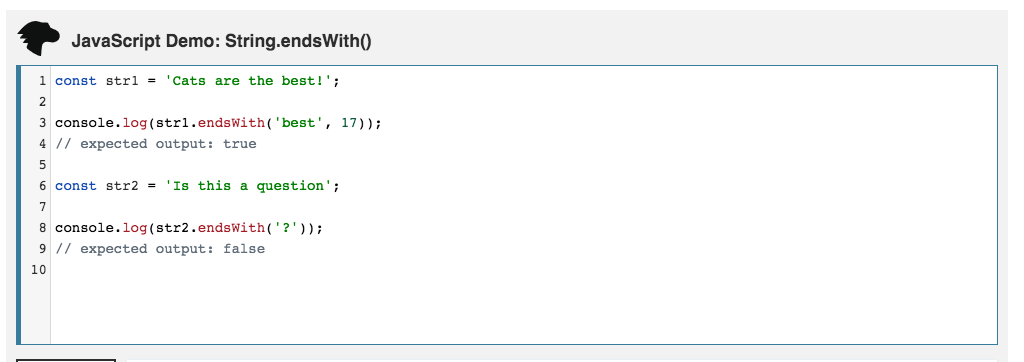
Topics of interest for JS 101 written exam:

* declarations, initialization, assignment, and re-assignment
  + Declarations: A variable declaration is a statement that asks the JavaScript engine to reserve space for a variable with a particular name.
  + Initialization: the assignment of an initial value to a data object or variable
  + Assignment: act of setting a variable value to a variable name, this returns the value on the right side of the statement
  + Reassignment:
* variable scope, especially how variables interact with function definitions and blocks
* variable scope determines where it is available in the program
  + variables with let or const keywords have block scope
  + a block is a related set of Javascript statements and expressions between a pair of opening and closing curly braces
* primitive values, objects, and type coercions
  + primitive values :
    - numbers
    - strings
    - undefined
    - null
    - Boolean
  + Objects: arrays and objects
  + Explicit type Coercion:
    - The person creating the program is the one who is deciding what to coerce
      * Number(), toString
  + Implicit type coercion
    - The engine is the one who is choosing what to coerce and how
      * Ie 4 + ‘34’
* object properties
  + store a collection of key-value pairs,
    - each item in the collection has a name that we call the key, and an associated value
      * keys can also be called properties
    - can access values in an object using 2 different ways:
      * dot notation: person.name
      * bracket notation: person[‘name’]
    - If you want to remove something from an existing object you can use the ‘delete’ keyword
      * This operation will return ‘true’ unless it cannot delete the property
* mutability vs. immutability vs. const
  + primitive values are always immutable
    - also can be considered atomic
  + Objects are almost always considered immutable
    - Although sometimes they are not
  + Object and primitive values are the data and functions that you use in your program. The rest is neither primitive or an object
    - Variables and other identifiers such as function names
    - Statements such as if, return, try, while, and break
    - Keywords, such as new, function, let, const, and class
    - Comments,
    - Anything else that is neither data nor a function
* loose and strict equality
  + loose equality will look for things that may equal each other, and may even coerce two values to find equality, ie ‘4’ == 4, this would evaluate as true
  + where as strict equality has to have data type and value be the same for it to be considered true, it has to be an identical 1:1 match for it to evaluate true
* passing arguments into and return values out of functions
  + arguments that are passed into the function are essentially values from outside the scope of the function brought into the function to be assessed and used.
  + In the definition of the function,. The names between parenthesis are called parameters. The **arguments** are the values of those parameters
  + Function names and parameters are both considered variable names in JS. Parameters in particular are local variables.
  + Return Values:
    - All functions return something unless they raise an exception, even if they don’t execute a return statement
    - If you don’t specify a return value for the function it will return undefined
    - Functions that always return true or false are called predicates
* working with Strings
  + useful functions to know
    - .concat()
      * Takes whatever word it is attached to and adds that together to the argument that is passed into it
    - .includes()
      * Checks to see if the word it is acting on includes that argument that is passed to it, there is a second argument if you want to specifcy at what index the first argument should occur at
    - .split()
      * 
    - .trim()
      * Removes whitespace from both ends of the strung its called on
        + Also removes any number of space characters as well like \n or \t
    - .toUpperCase(), .toLowerCase()
    - .charAt
      * Takes an index and return what character is at that index
    - String.charCodeAt()
      * Similar to char at but return Unicode code point or character code of the character at that index
      * If you don’t provide an index it assumes zero which return 97
      * String.charCodeAt() : is a Static methods a method. Meaning that it cant be called directly on to a string
    - .endsWith()
      * 
  + .beginsWith()
  + .repeat(),
    - Repeats whatever it is attached to for the amount of times that is passed in the argument
* working with Arrays, especially the iteration methods (forEach, map, filter, and find)
  + Graphical user interface, text, application, email

    Description automatically generated
* working with Objects; accessing keys and values of an Object as arrays
  + Object.Keys() takes an object and returns an array with the keys stored inside
  + Object.values() takes an object and returns an array with the keys stored
  + Object.entries() takes an object and retruns the key value pairs as nested arrays
  + Object.assign() takes two or more objects and combines them together
    - This mutates the first object listed as an argument
* arrays are objects

When you need to choose between an object or an array to store some data, ask yourself a few questions:

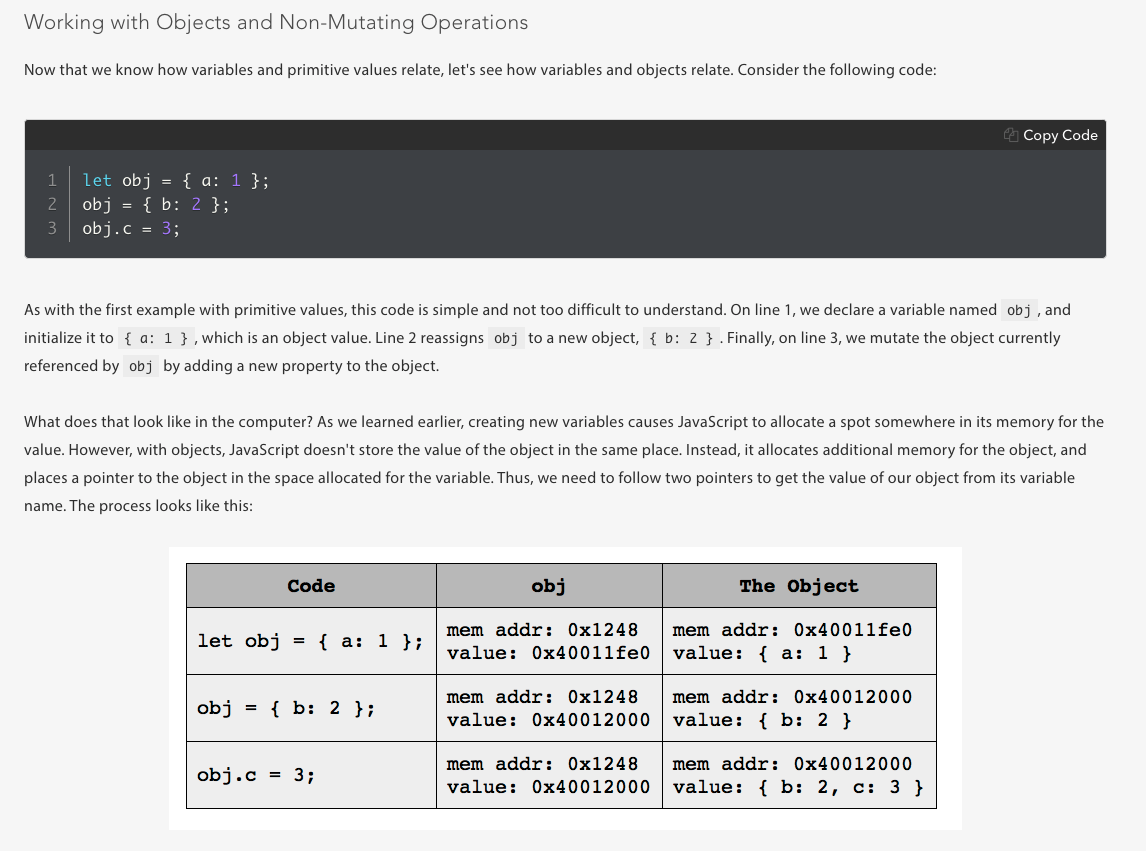
* Do the individual values have names or labels? If yes, use an object. If the data doesn't have a natural label, an array should suffice.
* Does order matter? If yes, use an array.
* Do I need a *stack* or *queue* structure? Arrays are good at mimicking simple "last-in-first-out" stacks and "first-in-first-out" queues.

As you grow as a developer, your familiarity with these data structures may affect which one you choose to solve a problem. Practice and experiment with each to find out which data structure works best in which situations.

* understand the concepts of *pass-by-reference* and *pass-by-value*
  + Pass by Value:
    - Means that when you use a variable to pass an argument to a function, the function cant do anything that sets the orginal variable to a different value
  + Pass by reference:
    - Essentially pointing to a refernce in he computer that holds a certain value, this refence can be changed. Essentially when an operation within the function mutates its argument, it affects the original object
  + In JS, Pass by value and pass by reference are only used in regards to calling and returning functions, not assignments
* variables as pointers
  + working with primitive values

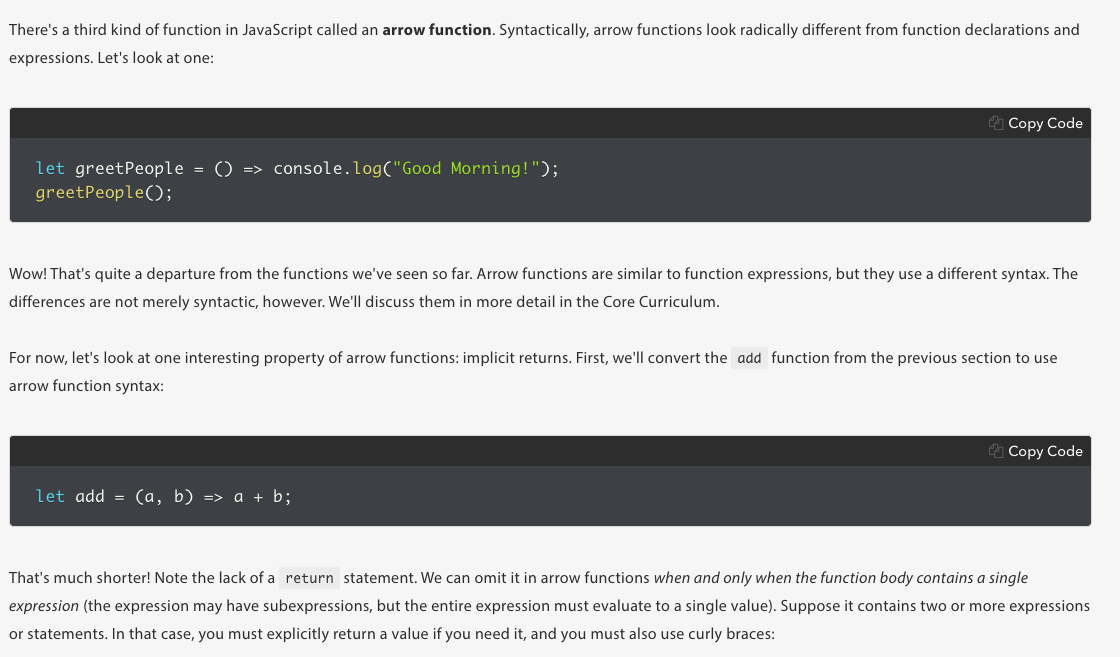
Table

Description automatically generated

* + Working with objects
* console.log vs. return
  + console.log is a static method that outputs onto the console what ever argument that you provide
  + a return value returns a specific value from the function
* truthiness vs. Boolean
  + truthiness is a way that a value can be considered while being evaluated in a Boolean setting.Graphical user interface, text, application, email

    Description automatically generated
* function definition and invocation
  + function invocation: Programmers often talk about function **invocation** and **invoking** functions. The terms are synonymous with "call" and "calling." You *invoke a function* or write a *function invocation*. We use these terms as well.
  + Function definition: the specific paramters and and methods used within the function that will be used when it is invoked.
* function declarations, function expressions, and arrow functions
  + function declaration:
    - Graphical user interface, text, application

      Description automatically generated
  + Function expression:
    - You cannot invoke a function expression before it appears in your programGraphical user interface, text, application

      Description automatically generated
  + Arrow functions:
    - 
* implicit return value of function invocations
  + undefined
* first-class functions
  + you can assign them to variables, pass them as arguments to other functions, and return them from a function call
  + all JS functions are first class functions
* side-effects
* Most functions should return a useful value or they should have a side effect, but not both.
  + Graphical user interface, text

    Description automatically generated
* naming conventions (legal vs idiomatic)
  + Table

    Description automatically generated
  + Legal (non idiomatic) Graphical user interface, text, application, table

    Description automatically generated
* be able to explain what a function does without talking about its implementation; that is, document a function's use and purpose. (See below.)