* “let” and “const”
  + Replaces “var” variable declaration
  + “let” is used for variable values
  + “const” is used for constant values (i.e. will never change)
* Arrow Functions
  + Arrow functions are a different way of creating functions in JS.
  + Old Syntax:   
    function myFunc (…) {  
     …  
    }  
      
    Arrow Syntax:  
    const myFnc = (…) => {  
     …  
    }  
      
    note: arrow functions can be initialized using “const” or “let”
  + Advantage of syntax is that is resolves issues with the “this” keyword that occasionally arises with the old syntax
* Exports and Imports (Modules)
  + Example:  
      
    person.js utility.js app.js   
      
    const person = { export const clean = () => {…} import person from ‘.person.js’  
     name: ‘Grant’; import prs from ‘./person.js’  
    } export const baseData = 10;  
     import { baseData } from ‘./utility.js’  
    export default person import { clean } from ‘./utility.js’
  + Default exports: import constName from ‘filename’  
    Named exports: import {x1 as Y1, x2 as Y, …} from ‘filename’
* Classes
  + Like java class blueprints for objects: contains properties and methods
    - Properties are like variables attached to your class/object
    - Methods are like functions attached to your class/object
  + Syntax:  
      
    Definition Instantiation Inheritance   
      
    class Person { const myPerson = new person() class Person extends Master  
     name = ‘Grant’ myPerson.call()  
     call = () => {…} consolelog(myPerson.name)  
    }
  + Properties in next gen JS
    - Old way:
      * constructor () {  
         this.myProperty = ‘value’;  
        }
    - New way:
      * myProperty = ‘value’
  + Methods in next gen JS
    - Old way:
      * myMethod (…) { … }
    - New way:
      * myMethod = (…) => { … }
* Spread & Rest Operators: : “…”
  + Denoted by three periods “…”
  + Spread
    - Used to split up array elements or object properties
      * const newArray = […oldArray, 1, 2] (pulls out elements in oldArray and puts them in newArray)
      * const newObject = {  
         …oldObject,   
         newProp: 5   
        }  
          
        adds all the attributes from oldObject to newObject
  + Rest
    - Used to merge a list of function arguments into an array
      * Function sortArgs(…args) {  
         return args.sort()  
        }
* De-structuring
  + Allows you to easily extract single array elements or object properties and store them in variables
  + Array De-structuring:
    - [a, b] = [‘hello’, ‘Max’]
      * console.log(a) // hello
      * console.log(b) // Max
  + Object De-structuring
    - {name} = {name: ‘Max’, age: 28}
      * console.log(name) // Max
      * console.log(age) // undefined, only identifier in curly brackets is accessible
* Reference and Primitive Types Refresher
  + Primitive types: numbers, strings, booleans
  + Reference types: objects, arrays, etc
* Array functions Refresher
  + Filter, sort, map, etc there are a ton of built in methods
  + Just look at documentation when needed